AO 120 (Rev. 08/10)

TO:	REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK
Mail Stop 8 Director of the U.S. Patent and Trademark Office P.O. Box 1450 Alexandria, VA 22313-1450	

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Western District of Texas, Waco Division on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO.	DATE FILED	U.S. DISTRICT COURT
6:22-cv-00241	3/4/2022	Western District of Texas, Waco Division

PLAINTIFF	DEFENDANT
Webroot, Inc. and OpenText Inc.	CrowdStrike, Inc. and CrowdStrike Holdings, Inc.

	PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1			
2			
3			
4			
5			

In the above---entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED 9/7/2022

INCLUDED BY ☑ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading

	PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1	11,409,869	8/9/2022	Webroot, Inc.
2	8,719,932	5/6/2014	Webroot, Inc.
3	8,201,243	6/12/2012	Webroot, Inc.
4	8,181,244	5/15/2012	Webroot, Inc.
5	8,856,505	10/7/2014	Webroot, Inc.
6	8,763,123	6/24/2014	Webroot, Inc.

In the above---entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1---Upon initiation of action, mail this copy to Director Copy 3---Upon termination of action, mail this copy to Director
Copy 2---Upon filing document adding patent(s), mail this copy to Director Copy 4---Case file copy

AO 120 (Rev. 08/10)

TO:	REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK
Mail Stop 8 Director of the U.S. Patent and Trademark Office P.O. Box 1450 Alexandria, VA 22313-1450	

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Western District of Texas, Waco Division on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO.	DATE FILED	U.S. DISTRICT COURT
6:22-cv-00239	3/4/2022	Western District of Texas, Waco Division

PLAINTIFF	DEFENDANT
Webroot, Inc. and OpenText Inc.	TrendMicro, Inc.

	PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1			
2			
3			
4			
5			

In the above---entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY
8/31/2022	☑ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading

	PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1	11,409,869	8/9/2022	Webroot, Inc.
2	8,719,932	5/6/2014	Webroot, Inc.
3	8,201,243	6/12/2012	Webroot, Inc.
4	8,181,244	5/15/2012	Webroot, Inc.
5	8,856,505	10/7/2014	Webroot, Inc.

In the above---entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1---Upon initiation of action, mail this copy to Director Copy 3---Upon termination of action, mail this copy to Director
Copy 2---Upon filing document adding patent(s), mail this copy to Director Copy 4---Case file copy

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
13/460,655	04/30/2012	Jerome L. Schneider	04584.0011USC1

CONFIRMATION NO. 9502

POWER OF ATTORNEY NOTICE

23552
MERCHANT & GOULD P.C.
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903

OC000000116329961

Date Mailed: 04/20/2020

NOTICE REGARDING CHANGE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 04/14/2020.

- The Power of Attorney to you in this application has been revoked by the assignee who has intervened as provided by 37 CFR 3.71. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

Questions about the contents of this notice and the requirements it sets forth should be directed to the Office of Data Management, Application Assistance Unit, at **(571) 272-4000** or **(571) 272-4200** or **1-888-786-0101**.

/tmwilliams/

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
13/460,655	04/30/2012	Jerome L. Schneider	04584.0011USC1

CONFIRMATION NO. 9502

109422
Sprinkle IP Law Group/OPEN
1301 W. 25th Street, Suite 408
Austin, TX 78705

POA ACCEPTANCE LETTER

OC000000116329982

Date Mailed: 04/20/2020

NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 04/14/2020.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

Questions about the contents of this notice and the requirements it sets forth should be directed to the Office of Data Management, Application Assistance Unit, at **(571) 272-4000** or **(571) 272-4200** or **1-888-786-0101**.

/tmwilliams/

Power of Attorney to Transact Business with the United States Patent and Trademark Office

I hereby revoke all previous Powers of Attorney given in the application(s) identified in the attached Transmittal for Power of Attorney to One or More Registered Practitioners ("Transmittal Form").

I hereby appoint practitioners associated with the following Customer Number associated with Sprinkle IP Law Group, PC :

109422

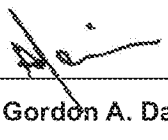
as my/our attorney(s) or agent(s) and to transact all business before the United States Patent and Trademark Office (USPTO) in connection with any and all patents and patent applications that are assigned to **Webroot Inc.** (the "assignee's patents/patent applications") in which they shall appear, any and all assignee's patents/patent applications associated with the above-referenced customer number, and any and all patents and patent applications referenced in the attached Transmittal Form.

Please recognize or change the correspondence address for the patents and patent applications referenced in the attached Transmittal Form to the address associated with the above-referenced customer number.

The undersigned is:

- ☐ Inventor or Joint Inventor
- ☒ Assignee or Person to Whom the Inventor is Under an Obligation to Assign
  *the undersigned has authority to act on behalf of the Assignee or Person to Whom the Inventor is Under an Obligation to Assign.
- ☐ Legal Representative of a Deceased or Legally Incapacitated Inventor
- ☐ Person Who Otherwise Shows Sufficient Proprietary Interest (e.g., a petition under 37 CFR 1.46(b)(2) was granted in the application or is currently being filed in this document)

The Transmittal Form may be completed and executed by one of the practitioners associated with the above-referenced customer number.

SIGNATURE OF APPLICANT			
Signature:		**Date:**	January 29, 2020
Name:	**Gordon A. Davies**		
Title:	Director		
Company:	**Webroot Inc.**		
I am authorized to act on behalf of the assignee.			

Electronic Acknowledgement Receipt

EFS ID:	39153891
Application Number:	13460655
International Application Number:	
Confirmation Number:	9502
Title of Invention:	Malware Management Through Kernel Detection During a Boot Sequence
First Named Inventor/Applicant Name:	Jerome L. Schneider
Customer Number:	23552
Filer:	John L. Adair/keunah cheun
Filer Authorized By:	John L. Adair
Attorney Docket Number:	04584.0011USC1
Receipt Date:	14-APR-2020
Filing Date:	30-APR-2012
Time Stamp:	14:23:42
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Power of Attorney	WEBROOT_POA_012820.pdf	42303 460c019d47e52952aad549f0d335131a49b6cfef	no	1

Warnings:

Information:	
Total Files Size (in bytes):	42303

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Large Entity Declaration

PATENT
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Number: 13/460,655 **Current Date** December 15, 2017

Filing Date: April 30, 2012

Patent Number: 8,856,505

Issue Date: October 7, 2014

CHANGE OF ENTITY STATUS PURSUANT TO 37 C.F.R. §1.27 (g)(2)

Commissioner for Patents
Mail Stop M Correspondence
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This communication hereby notifies the United States Patent and Trademark Office that small entity status is no longer applicable for the above-identified patent.

COMPANY or FIRM NAME AND ADDRESS:

Webroot Inc.
385 Interlocken Crescent, Suite 800
Broomfield, Colorado 80021

Respectfully submitted,

Signature
Ryan Howe
Printed Name
General Counsel
Title

OR
Reg. # if US Attorney

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	ISSUE DATE	PATENT NO.	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/460,655	10/07/2014	8856505	6890-45-CON	9502

23552 7590 09/17/2014

MERCHANT & GOULD PC
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903

ISSUE NOTIFICATION

The projected patent number and issue date are specified above.

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment is 24 day(s). Any patent to issue from the above-identified application will include an indication of the adjustment on the front page.

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Application Assistance Unit (AAU) of the Office of Data Management (ODM) at (571)-272-4200.

APPLICANT(s) (Please see PAIR WEB site http://pair.uspto.gov for additional applicants):

Jerome L. Schneider, Boulder, CO;

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation, and commercialization of new technologies. The USA offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to encourage and facilitate business investment. To learn more about why the USA is the best country in the world to develop technology, manufacture products, and grow your business, visit SelectUSA.gov.

Trend Micro, Inc.
EX1002-9

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	105	kernel AND rootkit	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 12:54
S2	3	S1 AND "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 12:55
S3	16	S1 AND @pd < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 12:56
S4	17	kernel AND rootkit SAME signature	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 13:45
S5	68	kernel AND rootkit AND signature	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 13:45
S6	13	S5 AND @pd < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 13:46
S7	36	S5 AND @ad < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 16:04
S8	0	driver SAME kernel SAME (detector	US-PGPUB;	ADJ	ON	2009/02/03

		OR monitor OR inspec$5) AND rootkey	USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			19:09
S9	690	driver SAME kernel SAME (detector OR monitor OR inspec$5)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/03 19:09
S10	523	S9 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/03 19:10
S11	34	(device driver) AND rootkit	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 11:11
S12	8	S11 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 11:11
S13	8	S11 AND @rlad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:10
S14	3	kernel NEAR2 detector SAME driver AND (malicious OR rootkit OR infect$3) NEAR2 driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:32
S15	4	kernel level detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:38
S16	5	kernel detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	ADJ	ON	2009/02/04 12:38

			IBM_TDB			
S17	2	(detector NEAR3 driver) AND kernel AND (malicious OR infect$4) driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:39
S18	3	(detector) AND kernel AND (malicious OR infect$4) driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:40
S19	4	detector AND kernel AND (malicious OR infect$4) (driver OR dll)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:41
S20	35	detector AND kernel AND ((malicious OR infect$4) (driver OR dll) OR rootkit)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:46
S21	20	S20 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:46
S22	145	(kernel level driver) OR (detector NEAR2 driver) AND kernel	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:05
S23	105	S22 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:05
S24	145	(kernel level driver) OR (detector NEAR2 driver) AND kernel AND detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:08
S25	105	S24 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS;	ADJ	ON	2009/02/04 13:08

			EPO; JPO; DERWENT; IBM_TDB			
S26	56	((kernel level driver) OR (detector NEAR2 driver)) AND kernel AND detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:10
S27	50	S26 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:10
S28	2	S26 AND @ad < "20051004" AND (malware OR virus OR rootkit OR worm)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:14
S29	94	kernel level driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:49
S30	0	("20070055711")".pn"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 14:41
S31	2	("20070055711").pn.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 14:41
S32	1	("20070055711").pn.	US-PGPUB; USPAT	ADJ	ON	2009/02/04 16:44
S33	1	("20070055711").pn. AND signature	US-PGPUB; USPAT	ADJ	ON	2009/02/04 17:49
S34	1	("20040049693").pn.	US-PGPUB; USPAT	ADJ	ON	2009/02/05 09:18
S35	1	("7398389").pn.	US-PGPUB; USPAT	ADJ	ON	2009/02/05 09:20
S36	14844	(380/276,277,278,284,286 OR 713/182,183,184,185,176 OR 726/28,30,33 OR 714/22-30).CCLS.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/05 11:30

S37	38	paul NEAR gassoway	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/13 15:28
S38	24	kernel AND (rootkit OR malware OR virus OR worm OR pestware) AND boot SAME (initialize OR start OR bootup) AND kernel WITH monitor	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:02
S39	14	S38 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:02
S40	83	kernel AND (rootkit OR malware OR virus OR worm OR pestware) AND boot$3 SAME (initializ$5 OR start$3 OR bootup) AND kernel WITH monitor$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:09
S41	55	S40 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:09
S42	762	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR "20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:39
S43	58	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:39

		"20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433").PN.				
S44	3	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR "20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433").PN. AND kernel AND boot$3 AND (malware OR virus OR worm OR pestware)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:41
S45	6	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR "20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433" OR "20020129277" OR "20030159070" OR "20040225877" OR "20050120242" OR "20050149726" OR "20050172115" OR "20050188272" OR "20050257266" OR "2006026576").PN. AND kernel AND boot$3 AND (malware OR virus OR worm OR pestware)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:46
S46	42	S40 AND @ad < "20060801" AND @pd < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:47
S47	37986112	S40 AND @ad < "20060801" OR @pd < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	ADJ	ON	2009/08/20 14:47

			IBM_TDB			
S48	55	S40 AND (@ad < "20060801" OR @pd < "20060801")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:47
S49	8349	S40 AND @ad < "20060801" AND (boot$3 WITH before) OR pre-boot$3 OR (prior WITH boot$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:50
S50	14	S40 AND @ad < "20060801" AND ((boot$3 WITH before) OR pre-boot$3 OR (prior WITH boot$3))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:51
S51	180	kernel AND boot AND Win32 AND (scan$4 OR monitor$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:09
S52	158	S51 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:10
S53	94	S51 AND @ad < "20060801" AND registry	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:11
S54	12	kernel AND boot AND Win32 AND (scan$4 OR monitor$3) WITH registry	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:13
S55	8	S54 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:14
S56	1	("20050125687").pn.	US-PGPUB; USPAT	ADJ	ON	2009/08/24 13:57
S57	1	("20050125687").pn. AND log	US-PGPUB;	ADJ	ON	2009/08/24

			USPAT			13:57
S58	18	kernel AND boot AND (change OR modify OR update) WITH (rule$2 OR polic$3) SAME (read WITH log)	US-PGPUB; USPAT	ADJ	ON	2009/08/24 14:28
S59	18	kernel AND boot AND (change OR modify OR update) WITH (rule$2 OR polic$3) SAME (read WITH log)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 14:32
S60	51	kernel AND boot AND (change OR modify OR update) WITH (rule$2 OR polic$3) AND (read WITH log)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 14:38
S61	26	S60 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 14:38
S62	1	("20030135791").pn.	US-PGPUB; USPAT	ADJ	ON	2009/08/25 12:03
S63	71	(713/164 OR 726/2,4,23-25) AND kernel AND (malware OR virus OR worm) AND boot$3	US-PGPUB; USPAT	ADJ	ON	2009/08/25 12:55
S64	7	jerome NEAR schneider	US-PGPUB; USPAT	ADJ	ON	2009/08/25 12:58
S65	331	(system OR device) driver SAME (malware OR pestware OR virus)	US-PGPUB; USPAT	ADJ	ON	2011/04/09 10:44
S66	209	S65 AND @AD < "20060801"	US-PGPUB; USPAT	ADJ	ON	2011/04/09 10:44
S67	85	S65 AND @AD < "20060801" AND kernel	US-PGPUB; USPAT	ADJ	ON	2011/04/09 10:45
S68	41	S65 AND @AD < "20060801" AND kernel AND boot	US-PGPUB; USPAT	ADJ	ON	2011/04/09 11:05
S69	1	native NEAR2 scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT	ADJ	ON	2011/04/09 11:59
S70	1	native NEAR2 scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2011/04/09 12:00
S71	433	scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2011/04/09 12:01
S72	16	boot With scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2011/04/09 12:20
S73	11	S72 AND @AD < "20060801"	US-PGPUB; USPAT	ADJ	ON	2011/04/09 12:20
S76	2	("8190868").PN.	US-PGPUB;	ADJ	ON	2013/10/04

			USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			16:03
S77	236	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus)	US-PGPUB; USPAT	ADJ	ON	2014/04/03 11:50
S78	53	S77 AND @ad < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 11:51
S79	53	S77 AND @ad < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 11:51
S80	9	kernel WITH (scanner OR monitor) WITH (rule OR policy) SAME (malware OR pestware OR antivirus OR anti NEAR virus)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:45
S81	0	S80 AND @ad < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:46
S82	31	kernel WITH (scanner OR monitor) WITH (rule OR policy) AND (malware OR pestware OR antivirus OR anti NEAR virus)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:47
S83	0	S82 AND @ad < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:47
S84	0	S82 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:49
S85	45	kernel WITH (scan$4 OR monitor$4) WITH (rule OR policy) AND (malware OR pestware OR antivirus OR anti	US-PGPUB; USPAT; USOCR;	ADJ	ON	2014/04/03 13:50

		NEAR virus)	FPRS; EPO; JPO; DERWENT; IBM_TDB			
S86	4	S85 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:51
S87	97	kernel WITH (scan$4 OR monitor$4) SAME (rule OR policy) AND (malware OR pestware OR antivirus OR anti NEAR virus)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:54
S88	13	S87 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:54
S89	9	S88 NOT S86	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:54
S90	114	kernel WITH (scan$4 OR monitor$4) SAME (rule OR policy) AND (malware OR pestware OR antivirus OR anti NEAR virus OR spyware)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:01
S91	25	S90 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:01
S92	21	S91 NOT S86	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:01
S93	12	S92 NOT S89	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:02
S94	185	kernel WITH (scan$4 OR monitor$4)	US-PGPUB;	ADJ	ON	2014/04/03

		SAME (rule OR policy) AND (malware OR pestware OR antivirus OR anti NEAR virus OR spyware OR malicious)	USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			14:05
S95	56	S94 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:05
S96	47	S95 NOT S89	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:05
S97	43	S96 NOT S86	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:06
S98	1	("20030159070").PN.	US-PGPUB; USPAT	ADJ	ON	2014/04/03 16:30
S99	236	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus)	US-PGPUB; USPAT	ADJ	ON	2014/04/17 15:43
S100	28	boot With scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2014/04/17 15:48
S101	1	("20120216027").PN.	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2014/04/17 15:52
S102	223	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus) AND (G06F$8).CPC.	US-PGPUB; USPAT	ADJ	ON	2014/04/17 15:53
S103	45	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus) AND (G06F21/56).CPC.	US-PGPUB; USPAT	ADJ	ON	2014/04/17 15:55
S104	41	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus) AND (G06F21/566).CPC.	US-PGPUB; USPAT	ADJ	ON	2014/04/17 15:55
S105	23	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus) AND (G06F21/554).CPC.	US-PGPUB; USPAT	ADJ	ON	2014/04/17 15:55

EAST Search History (Interference)

Ref	Hits	Search Query	DBs	Default	Plurals	Time

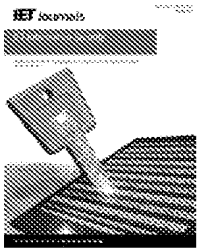
#				Operator		Stamp
S74	0	(713/2,164,193).CCLS. AND ((pestware AND boot AND sequence AND period AND drivers AND initialized AND kernel-level AND monitor AND boot AND managing AND related AND before AND native AND application AND loaded AND subsystem AND scanning AND scanner AND registry AND behavior AND rules) OR (launching AND operating AND system AND pestware AND management AND behavior AND rules AND portion AND compiled AND engine) OR (generating AND response AND monitoring AND record AND events AND pestware-related AND analyzing AND identify AND behavior AND rules AND prevent)).CLM.	USPAT; UPAD	ADJ	ON	2012/01/14 20:55
S75	0	(726/4,22-27).CCLS. AND ((pestware AND boot AND sequence AND period AND drivers AND initialized AND kernel-level AND monitor AND boot AND managing AND related AND before AND native AND application AND loaded AND subsystem AND scanning AND scanner AND registry AND behavior AND rules) OR (launching AND operating AND system AND pestware AND management AND behavior AND rules AND portion AND compiled AND engine) OR (generating AND response AND monitoring AND record AND events AND pestware-related AND analyzing AND identify AND behavior AND rules AND prevent)).CLM.	USPAT; UPAD	ADJ	ON	2012/01/14 20:56
S106	0	(713/2,193) AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:10
S107	0	(G06F21/56).CPC. AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:11
S108	0	(713/2,193).CCLS. AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:12
S109	0	(G06F21/566).CPC. AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:12

		native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.				
S110	0	(G06F21/554).CPC. AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:13
S111	0	(managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:13

7/ 30/ 2014 1:53:20 PM
C:\ Users\ glane\ Documents\ EAST\ Workspaces\ case_13460655.wsp

www.ietdl.org

Published in IET Information Security
Received on 8th November 2012
Revised on 1st February 2013
Accepted on 1st February 2013
doi: 10.1049/iet-ifs.2012.0343



ISSN 1751-8709

Approach for malware identification using dynamic behaviour and outcome triggering

Hao Bai[1], Chang-zhen Hu[1], Xiao-chuan Jing[2], Ning Li[2], Xiao-yin Wang[2]

[1]*School of Computer Science and Technology, Beijing Institute of Technology, Beijing, People's Republic of China*
[2]*China Aerospace Engineering Consultation Center, Beijing, People's Republic of China*
E-mail: howard.bai85@gmail.com

Abstract: Malware identification is the process of determining the maliciousness of a program, which is necessary for detecting malware variants. Although some techniques have been developed to confront the rapid expansion of malware, they are not efficient to recognise booming malware instances, and can be evaded by using obfuscation techniques. In this study, a novel dynamic malware identification approach is proposed. Concretely, this approach employs techniques that explore multiple execution paths and trigger malicious behaviours with resulting outcomes. To this end, a group of featured malicious behaviours and outcomes (MBOs) are primarily constructed, from which weights for malware family classification are derived. A virtual monitor is then developed to dynamically trigger MBOs by exploring multipath with suitable probing depths. Finally, triggered malicious behaviours are modelled with features recorded in MBOs to train a malware classifier which can identify unknown malware variants. The experimental results on test cases demonstrate the proposed approach is effective in identifying new variants of popular malware families. The comparison with latest malware identifiers shows that our approach achieves lower false positive rate and can recognise malware equipped with obfuscation techniques.

1 Introduction

Malware (malicious software) is a kind of unauthorised software that damages computer systems. Although many kinds of techniques have been developed to analyse and identify malware, such software still poses a serious threat to the public. The fundamental issue of identifying malware is to determine whether a piece of binary is malicious or not, which means it has to be aware that if a programme is semantically equivalent to a known malware sample [1]. This has been proved to be a critical problem, since malware is crafted by adversary so that counter-measures like obfuscation techniques are equipped to thwart analysis or detection [2]. What is worst is that making malware has turned into a sophisticated and profitable business, boosting the swelling of malware variants in the wild. Most of these variants are packed or obfuscated to hide their malicious features; some variants are not even recognised as valid executable programs.

Currently, two kinds of techniques are widely employed to identify malware: static analysis and dynamic analysis. During static analysis, a programme is analysed without actually being executed. In this process, a program's binary is preliminarily disassembled by a reverse engineering tool like IDA Pro [3], which occupies the control flow graph (CFG) of the programme [4]. Every individual function in the CFG is depicted in details with all the variables, registers, system calls and arguments [4]. Then, techniques like signature matching and semantics analysis can be utilised to determine the functionality or maliciousness of the binary. This process is very much like the one utilised in commercial anti-virus software. Static analysis is faster than dynamic analysis and is precise in detecting known malware instances as it scans the complete binary and extracts signatures to do pattern matching. However, obfuscation techniques such as polymorphism [5, 6] and metamorphism [7–10] have been demonstrated being effective in evading static identification [11]. The reason is that the static analyser can only extract a binary's semantics which may be tampered in the self-modifying binary after the obfuscation process. Another limitation of static approaches is that the malicious binary under analysis may not be the one that actually runs, so the syntactic signatures are missed in static resolution.

The limitation of static analysis motivates the use of its dynamic counterpart. Dynamic analysis is the process of analysing a programme during the run-time. As a result, run-time behaviours are used as evidence to determine the maliciousness of a program. In practice, a virtual machine or an emulator is usually employed to simulate the real system environment. A suspected binary is run in the virtual system to reveal its behaviours. In this process, all the actually executed instructions of the binary can be monitored, and thus the functionality of the programme can be drawn according to the actions exposed by the binary. Using the model constructed with malicious actions, behaviour-based malware detection technique is possible to identify unknown malware pieces and to model for multiple malware families [12]. Prior to static techniques, dynamic analysis has been proven more effective in classifying and

© The Institution of Engineering and Technology 2014

Trend Micro, Inc.
EX1002-23

identifying malware which is adversely designed to resist detection. Specifically, dynamic technique is immune to obfuscation and metamorphism, because only the binary that actually executes is analysed. Although, dynamic malware identification is still a challenging task, requiring one to reliably extract all the malicious behaviours and clearly understand the resulting outcomes or impacts on the system.

Some traditional dynamic approaches ignore the outcomes or consequences made by malicious behaviours, and thus they suffer relatively high false positive rates. Moreover, the traditional methods lack the ability to model for the classification of malware, resulting in ineffective detection when encountering unknown malware variants. The latest security threat analysis [13] reports that there were 1.8 billion detected malware instances over the course of 2011, compared with merely 403.8 million identified malware families, indicating that most malware instances are variants of finite malware families.

All the techniques discussed above demonstrate that, compared with static analysis, dynamic analysis is more practical in identifying obfuscated malware variants. This motivates us to focus on a more effective dynamic identification of malicious behavioural characteristics. Considering that malicious behaviours make impacts on the target execution environment [14], it is necessary to jointly monitor the outcomes triggered by malicious behaviours. Malicious outcomes include accessing suspect network terminal and tampering the system states like memory, register and files [15]. Hence, the accuracy of malware identification can be greatly improved, as suspect malicious behaviours can be verified by the observation of outcomes. Moreover, in order to recognise malware variants equipped with obfuscation techniques and detect previously unseen malware instances, it is necessary to represent malware in abstract classifications [12]. Our approach focuses on identifying Windows-based malware, because Windows series systems have become primary targets to malware writers.

In this study, we propose a dynamic malware identifier that consists of three key procedures: firstly, a group of featured malicious behaviours and outcomes (MBO) are established with representative malware samples that have typical malicious characteristics; then, malicious behaviours are triggered with their resulting outcomes from malware in a virtual monitor; and finally, the malicious feature matrix is constructed as the feed to train a modified supporting vector machine (SVM) which identifies unknown malware pieces. Fig. 1 illustrates the kernel architecture of the proposed approach.

In the first procedure, shown as path '1' in Fig. 1, a group of MBOs are constructed to stand for representative malware features. MBO consists of featured MBOs that have causal correlation. With an attribute to register the weight of malicious behaviours, MBO denotes the maliciousness and category of a target binary. In the multipath exploration procedure, malware binaries are analysed in a virtual monitor that takes two steps to explore the branches of execution paths and find a practical searching depth to trigger malicious behaviours with outcomes. Specifically, the processes of analysing training samples and testing samples are tagged with paths '2A' and '2B', respectively. In the last procedure, the malicious feature matrix is established with MBO and triggered behaviours. Separately, the processes of training malware classifier and identifying malware variants are tagged as paths '3A' and '3B'. For the

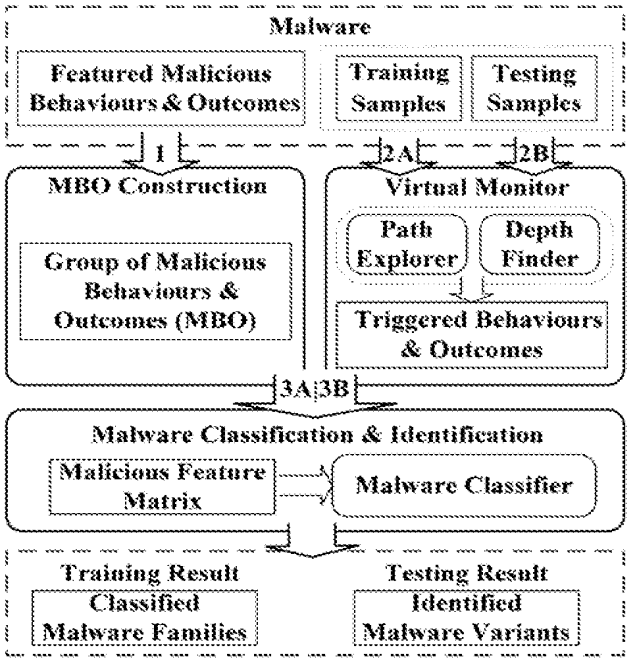


Fig. 1 *Kernel architecture of the proposed approach*

training process, the matrix is an abstracted model of a malware family and is fed to a modified SVM classifier [16]. The SVM is a powerful classifier in characterising small sample models, for which the malware issue is suitable because the quantity of malware samples gathered in the laboratory can hardly be large enough. The output of the training process is classified into malware families. Then, with the trained classifier, unknown malware variants can be identified.

The main contributions of this study are 3-fold: (i) the MBO provides a metric to determine the malicious degree of a binary and to weigh for malware classification. Traditional obfuscation techniques aimed at defeating semantic-based static signatures are ineffective, as MBO is built on run-time behavioural features; (ii) compared with the latest malware identifiers, the proposed approach focuses on dynamic analysis environment in which multipaths are explored and malicious behaviours are triggered to yield resulting outcomes within a suitable probing depth, meanwhile improving the accuracy of identification. The analysis is comprehensive, because our approach monitors suspicious user/native function calls as well as invalid modification of system space; (iii) the malicious feature matrix is built to model triggered malware features that can be processed by a modified multi-classification SVM [16]. All the featured malicious behaviours in MBO are abstracted as the materials for malware classification, so the malicious feature matrix can perform an accurate characterisation of all distinguished malware families. The experimental results demonstrate that the proposed approach is effective in triggering hidden malicious behaviours with appropriate system overhead. In the process of identifying multiple malware variants including obfuscated ones, our approach outperforms the static detectors and semantic-based dynamic approaches.

The remainder of the paper is organised as follows: Section 2 presents related work in the area of malware identification. The proposed approach including detailed techniques and

© The Institution of Engineering and Technology 2014

Trend Micro, Inc.
EX1002-24

algorithms is introduced in Section 3. Section 4 shows the experimental evaluation. Conclusions and future work are addressed in Section 5.

2 Related work

In recent years, there is a growing body of work in the area of malware analysis and identification. Researchers have developed multiple techniques to extract malware features, make model to distinguish malware families and recognise emerging malware variants. We observe that currently two major techniques are the most popular: static identification and dynamic identification.

Static identification: Kang *et al.* [17] propose an approach to reverse and recognise malware families using control-flow abstracting. Their system works by searching common characteristics in a malware family's binaries, and by using CFGs to derive functions with malicious features. Their method focuses on the abstraction of malware's semantic signature rather than on the binary code sequences, so the detection on malware can be scalable and fast. However, they cannot deal with obfuscated malware instances, because no dynamic features are extracted. Similarly, Cha *et al.* [18] implement a system using Bloom filter to filter out benign binaries and off-interest malicious features prior to the signature matching step. Although Cha's system can improve recognition performance as all the safe binary and identified malware pieces are not processed in the scanning phase, it relies on traditional static comparison to perform further identification. As a result, it is unable to detect packed or obfuscated malware variants.

Researchers who concentrate on static analysis have been aware of the obfuscation techniques used by malware writhers. Lakhotia *et al.* [19] present a method to statically identify malware's call instructions that are obfuscated by stack related instructions. Their method abstracts the stack instructions and models the abstracted stack graphs with instructions' addresses. Although the obfuscated call instructions can be detected, their method lacks the ability to tackle with malware variants which may be obfuscated with metamorphic techniques. Tabish *et al.* [20] propose a malware detection technique which employs data mining algorithms to identify malicious features in the block-wise level. Their method is robust to some common evasion techniques. Although it is still a challenging task for their method to detect manually obfuscated malware pieces which only take malicious actions with specific triggers.

Dynamic identification: To overcome the disadvantage of static approaches, Jiang *et al.* [21] design a virtual machine-based monitor to extract the semantic information of malware. Although this system can reconstruct details of malicious behaviours, it ignores the outcomes of malicious behaviours and does not use an appropriate data structure for modelling. To tackle these problems, Kolbitsch *et al.* [22] propose a malware detection approach that characterises malicious behaviours and extract the instruction slices responsible for such behaviours. However, obfuscated malware can still evade the detection, as the binary slice can be easily transformed to a variant form. Similar to the proposed approach, Anderson *et al.* [23] and Pleszkoch *et al.* [24] both introduce malware detection algorithms that model instruction traces of target binary. They construct graphs to represent instruction sequences sharing equivalent malicious behaviours. Theoretically, their methods can measure similarity between malicious graphs, and thus is sufficient to perform malware classification in the representation of (Hidden) Markov Model. However, their model needs huge amount of input data, whereas in real-world circumstance, the amount of collected malware samples is limited, probably affecting the building of (Hidden) Markov Model.

According to our investigation, few methods available can identify malware effectively, meanwhile promising low-false positive rate. Considering those static methods can be easily evaded by obfuscation techniques, the proposed approach extracts run-time behaviours as well as their outcomes to characterise malware features. Seeing that some dynamic techniques lack the ability to model or classify malware families, the proposed approach employs a suitable model to identify multiple malware variants. The detailed techniques of the proposed approach are presented in Section 3.

3 Proposed approach

3.1 MBO construction

In this subsection, a group of MBOs are built to define multiple malicious features which embody malicious behaviours and their correlated outcomes. MBOs in the group are denoted as $\{M_1, M_2, \ldots, M_v\}$, where v is a variable to represent the quantity of malicious features according to practical needs. In the proposed circumstance, M_1–M_6 are built to represent six categories of run-time malicious features: (i) abuse of suspected user API, (ii) abuse of suspected native API, (iii) occupation of INT 13H by force, (iv) writing boot sector, (v) invalid modification of EPROCESS structure and (vi) invalid modification of kernel input libraries. The advantage of MBO is that false positive/negative rates can be greatly reduced as alarm is raised only if malicious behaviours are detected along with their resulting outcomes. Equation (1) shows the structure of M_i

$$M_i = \{L_i(\text{beh},\ \text{out}),\ w_i\}, \quad (1 \le i \le 6) \tag{1}$$

The L_i(beh, out) is a logical expression containing malicious behaviours beh and outcomes out based on causal correlation. Equation (2) shows the structure of a single L_i(beh, out)

$$L_i(\text{beh}, \text{out}) = \left\{\text{beh}_1 \wedge \text{beh}_2 \wedge \cdots \wedge \text{beh}_{n_i} \Rightarrow \text{out}_1 \vee \text{out}_2 \vee \cdots \vee \text{out}_{n_j}\right\}, \quad (n_i \ge 1,\ n_j \ge 1) \tag{2}$$

In (2), all beh are correlated with logical 'and' relation, and all out are correlated with logical 'or' relation, which means strictly, at least one element of set out will be triggered, if and only if all beh are monitored during run-time.

In real-world circumstance, when some outcomes from the out set are triggered, some of the beh may not be monitored, whereas others may be detected for multiple times. Consequently, the Boolean association rules are not sufficient to judge the maliciousness of a binary, and we have to record the appearances of beh and out to calculate their weights which are denoted as w_i. Consequently, w_i indicates a binary's malicious degree on M_i.

© The Institution of Engineering and Technology 2014
IET Inf. Secur., 2014, Vol. 8, Iss. 2, pp. 140–151
doi: 10.1049/iet-ifs.2013.0318
Trend Micro, Inc.
EX1002-25

Table 1 Statistics of the occurrence of all the beh and out in MBOs for one binary instance

	ct_1^b	ct_2^b	–	$ct_{n_i}^b$	ct_1^o	ct_2^o	–	$ct_{n_j}^o$
M_1	3	2	–	0	1	0	–	0
M_2	0	3	–	7	0	3	–	0
–	–	–	–	–	–	…	–	–
M_6	5	2	–	9	0	0	–	2

During the analysis of a binary, in order to calculate w_i, the occurrence counts of detected beh and out from all MBOs are recorded in a table, where the count of beh is denoted as ct^b which is used to be the weight factor of w_i, and the count of out (represented by ct^o) must be at least 1 to testify the detected beh is really malicious. Table 1 shows an example of the statistics of the occurrence of all the beh and out in MBOs for one binary instance under analysis.

For the convenience of drawing the table, the numbers of beh and out (i.e. n_i and n_j) in all the MBOs are unified to align the table. In practice, every n_i of an MBO is decided by the analyst, and the numbers of beh are presented in Section 4.2. The calculation of w_i is formulated as a multivariate linear regression equation which is shown in (3)

$$w_i = a_0 \times \beta_0 + ct_1^b \times \beta_1 + ct_2^b \times \beta_2 + \cdots + ct_{n_i}^b \times \beta_{n_i} + \mu, \quad (1 \le i \le 6) \tag{3}$$

The sequence of ct^b is sorted according to the importance degree of beh, where important beh is sorted in front and beh of less importance is laid behind. If these beh are not logically related, they are sorted with numerical sequence. As the importance degrees of beh are explicit and beh are accordingly sorted, constant term a_0 and the mean of random error μ can be set to 0. As a result, the equation can be standardised as (4)

$$w_i = ct_1^b \times 10^{-1} + ct_2^b \times 10^{-2} + \cdots + ct_{n_i}^b \times 10^{-n_i}, \quad (1 \le i \le 6) \tag{4}$$

In the construction of MBOs, the beh of M_i are determined by expert analysis. In order to make the selected beh be of the most representation of malicious features, it is necessary to precisely understand the motivation of the malware pieces and distinguish between malicious behaviours and those of benign purposes. This is essential, because some legal binaries interact with the system kernel, and thus reveal suspect behaviours that may be mistakenly judged as malicious. For example, a legal binary can read process information from the kernel, write entries to another file or access the internet to download its updates. Therefore a white list is attached in MBO to filter out benign sources and obviate false alarm. The key insight here is that every single behaviour in beh does not warrant maliciousness, but together and with the corresponding outcomes are positive indicators of malware.

3.2 Triggering multipath behaviours

3.2.1 Virtual monitor: The proposed approach is equipped with a virtual monitor to dynamically analyse malware that commonly employs obfuscation to foil static analysis. To this end, the execution of malware is under surveillance in a QEMU-based [25] simulated PC environment in which, with the obtained information, we can observe how malicious behaviours are touched off and how particular outcomes are activated by triggered malicious behaviours.

The proposed approach utilises QEMU [25], an open-source system emulator, to develop the proposed virtual monitor. The primary task of the virtual monitor is to emulate a machine. Then, based on that, the monitor executes all the instructions of a target binary, and keeps watching the behaviours of the binary. As the monitor can act like a real machine to the programme under surveillance, it is difficult for the malware to realise that it is being monitored in a simulated environment. Another reason why the virtual monitor is proposed is that it protects the analysis environment from tampering by the malware as the analysis phase and the execution phase are separated.

During monitoring the malware, QEMU may encounter some unforseen instructions and may lead to termination. The reason is that, in QEMU, all the instructions are parsed by the 'translate.c' in which a 'switch-case' structure is in charge of identifying multiple instructions. Some specific instructions may not be included in the 'case' judgement list, resulting in unexpected termination. In the proposed work, when encountering a new instruction which cannot be identified by the 'switch-case', this instruction is parsed and added into the 'case' judgement list. The source code in 'translate.c' needs to be modified to amend the 'switch-case' structure. The proposed system will identify the newly added instruction when encountering it next time.

To facilitate an efficient monitoring process, the QEMU-based virtual monitor employs a technique called dynamic translation on the machine code that are interpreted as basic blocks. Formally, a basic block is defined as a sequence of instructions, where each instruction must execute before all the later ones [11]. The execution of a basic block ends with a jump instruction that is delineated as a 'trigger instruction' in the proposed work. Since the control-flow of malware consists of multiple branches, malicious behaviours are touched off only in some specific locations, where trigger instructions make the execution go down [26]. At binary level, all the trigger instructions fall into three categories: direct jump instructions, condition jump instructions referring to register CX (ECX) and condition jump instructions referring to register EFLAGS.

The process of handling a trigger instruction is described in Fig. 2. In the first step, when encountering a trigger instruction, the virtual monitor records the instruction length, its next-address and jump-to-address. Then, in the

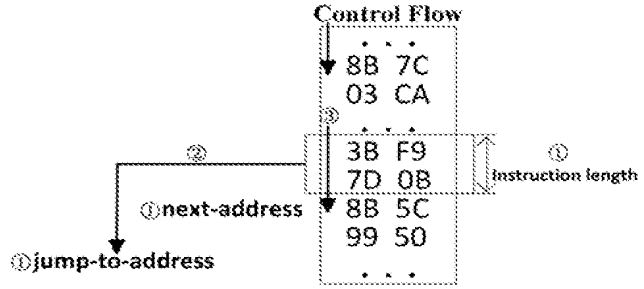


Fig. 2 *Process of handling a trigger instruction*

© The Institution of Engineering and Technology 2014

Trend Micro, Inc.
EX1002-26

second step, the monitor ignores the jump condition, directly moves control to jump-to-address and executes following instructions. Finally, when the execution in jump-to-address is finished, the monitor comes back to the trigger instruction, skips over it according to its length and executes instructions starting with next-address.

The proposed virtual monitor explores execution paths to an utmost expansion and controls the execution depths to a manageable level. During exploration of execution paths, the monitor disassembles the malware in binary level, so malicious behaviours are observed in basic blocks. The execution suspends until all derived paths are traversed in the on-going basic block. Meanwhile, the current CPU status and contents in memory and registers need to be saved and restored when execution comes back to the suspended instructions starting from the next-address.

Since QEMU is mounted in the Linux system, the fork() function is employed by the proposed system to traverse the derived paths. When two derived paths are encountered, the proposed system calls the fork() function and fork the two paths into two independent QEMU processes. Then, by saving the CPU status and other information, the proposed system separately runs these two forked QEMU virtual machines in which these two derived paths are executed, respectively.

3.2.2 Multipath exploration: To improve richness and speed of analysis, a breadth-first-based traversal algorithm is implemented, where a breadth-first tree is developed along execution paths with basic blocks as the vertices. The traversal starts at the first parsed basic block, and continues to visit all the vertices adjacent to their predecessors. The exploration is ergodic, as the proposed algorithm can visit all the reachable basic blocks. Since the breadth-first traversal can identify cycles on the course, the algorithm can avoid infinite loop during exploration.

Considering the traversal algorithm visit basic blocks level by level, it is necessary to set a threshold of the traversal depth in order to keep the exploration non-exhausted. The information entropy [27, 28] is applied to figure out the threshold, in the light of the application of entropy in information theory to measure the uncertainty of a collection of data items. In the proposed circumstance, information entropy [27, 28] can be utilised as a measure of the regularity of trigger instructions in basic blocks. Each set of basic blocks in which malicious behaviours are triggered represents a class. Consequently, the fewer basic blocks the set have, the smaller the entropy is. For M_i, depth_{M_i} is defined as the maximal depth to trigger its beh. In the multipath exploration, malware samples with identical features of M_i are employed to obtain the distribution of beh in basic blocks, where the maximum distribution indicates depth_{M_i} that is, $H(M_i)$, the entropy of M_i. The sparser the beh distribute, the bigger the depth_{M_i} is, implying more basic blocks are occupied. Contrarily, the depth_{M_i} is smaller when the beh are more centralised in the basic blocks.

Definition 1: C_{M_i} is a class consisting of basic blocks. For M_i whose items beh belong to C_{M_i}, the $H(M_i)$ relative to this C_{M_i} wise classification is defined in (5), where $P(\text{beh})$ is the probability of beh in M_i

$$H(M_i) = \sum_{\text{beh} \in C_{M_i}} P(\text{beh}) \log \frac{1}{P(\text{beh})}, \quad (1 \le i \le 6) \qquad (5)$$

The algorithm presented in Fig. 3 shows how to trigger malicious behaviours in an unknown malware piece. This algorithm works by constructing a breadth-first tree along the analysed CFG of a procedure. On lines 1–5, basic blocks in the CFG are indicated as vertices in the tree, and the first parsed basic block the root of tree; the triggered malicious behaviours in beh are recorded in a set mab which is initialised to empty; a counter dcount is employed to limit the traversal level to depth (i.e. depth_{M_i}); we maintain a set of queues to contain the adjacent vertices of their predecessor. Line 6 shows the root of the tree is the first to insert the queue, meanwhile tagged with 'visited'. The traversal process is reflected in lines 7–20: in the condition of the tree being not fully traversed and the upper limit of depth being not reached, if the mab is not full, the trigger progress continues with all unvisited vertices. When a malicious behaviour b is triggered and the set $\{b\}$ is not a subset of mab, as shown on line 12–15, mab is updated to contain b and the related vertex is tagged with 'visited'. After the traversal is completed on one level of a predecessor vertex, the value of dcount increases one, which is depicted on line 18. When the algorithm is finished, it implies that either all malicious behaviours in beh are triggered or the traversal limit depth_{M_i} is reached. The output of this algorithm is mab which is a proper subset of beh.

In the analysis of an unknown malware piece, the traversing level to trigger beh should be no more than depth_{M_i}. That is, the trigger progress continues until the exploration depth reaches depth_{M_i}, or every behaviour in beh is triggered. The setting of the threshold on traversal depth lies in the assumption that, in real-world analysis, trigger instructions of malware are mostly spotted in consecutive basic blocks. Although it is obvious that bigger depth_{M_i} comes with higher trigger rate, it comes with more time/system consuming. We present in the experiment that depth_{M_i} yielded by $H(M_i)$ functions properly with malware instances in the wild.

As the virtual monitor watches the malware tightly, once a malicious behaviour is triggered, the outcome it yields is also registered. Using the algorithm described in this subsection, we are in a position of recognising both malicious behaviours and their resulting outcomes. For instance, if a malware sample invokes a user level API to make a copy of it, the virtual monitor will detect that some behaviours are made to create an unauthorised file. As the outcome, a copy of this malware piece is stored in hardware. This impact will be solidly recorded, as the proposed virtual monitor emulates the whole system space, ensuring the modification on memory or other devices totally under surveillance.

3.3 Modelling malware for real-world detection

In real-world circumstance, unknown malware pieces are modelled and identified according to their dynamic features. The malicious feature matrix is built with MBO and the trigger results derived from identified malware instances that are split into m groups, where m is the quantity of malware families used for training. After the multipath exploring procedure, every instance comes with an established table introduced in Section 3.1, filled with MBOs captured in Section 3.2. Thus, m malicious feature matrices are built to delegate the training malware families, where for convenience, each family is set to contain k samples. The appearance of w_i defined in Section 3.1 is expanded to w_{ji}, where $1 \le j \le k$ and $1 \le i \le 6$. The structure

© The Institution of Engineering and Technology 2014
IET Inf. Secur., 2014, Vol. 8, Iss. 2, pp. 140–151
doi: 10.1049/iet-ifs.2013.0318
Trend Micro, Inc.
EX1002-27

Input: Control Flow Graph (CFG) of a procedure
Output: triggered *mab*

```
1.  vertices ← all basic blocks in the CFG
2.  root ← the first parsed basic block
3.  mab ← ∅
4.  dcount ← 0
5.  makeEmptyQueue(Q[i]) //i is initialised to 0
6.  insert(Q[i], root, VISITED)
7.  while ¬empty(Q[i]) and dcount ≤ depth do
8.    if mab ⊂ beh then
9.      makeEmptyQueue(Q[i + 1])
10.     for all v ∈ Q[i] do
11.       for all w ∈ UnvisitedAdjacent(v) do
12.         if trigger(b) ∈ beh and {b} ∉ mab then
13.           mab ← mab ∪ {b}
14.           insert(Q[i + 1], w, VISITED)
15.         end if
16.       end for
17.     end for
18.     dcount ← dcount + 1
19.   end if
20. end while
```

Fig. 3 *Algorithm to trigger malicious behaviours in an unknown malware piece*

of malicious feature matrix is defined as

$$\begin{pmatrix} \langle w_{11}, w_{12}, \ldots, w_{16} \rangle \\ \langle w_{21}, w_{22}, \ldots, w_{26} \rangle \\ \vdots \\ \langle w_{k1}, w_{k2}, \ldots, w_{k6} \rangle \end{pmatrix} \quad (6)$$

In the malicious feature matrix, every element is the result of (4), and each row denotes the result extracted from one malware instance and all six MBOs.

Malicious feature matrices are fed as input to train an SVM-based classifier which is used for identifying unknown malware variants. However, a significant drawback of SVM is that it is originally designed for two-classification. To address this problem, a structure of binary tree [16] is applied to make the classifier capable of classifying multiclass malware. For m malware families that are abstracted to m matrices, a non-linear mapping-based SVM is employed to map these m categories of malware to $m-1$ dimensional hyperplanes. Then, a kernel function is used to map a target binary into this high-dimension space. Consequently, the target binary can be determined malicious if it belongs to one of the hyperplanes, or determined benign if excluded by the space.

The classification process is analogous to building a binary tree: the recognition of each family begins at the root of the tree. The classifier firstly divides all the matrices into two subclasses, and then divides each subclass into two secondary classes, respectively. This process continues recursively down the tree until every leaf of the tree contains only one family of malware. Fig. 4 illustrates an example of the binary tree-based SVM [16] that solves a five-class malware classification problem, where each node of the tree makes a two-class decision using a basic SVM.

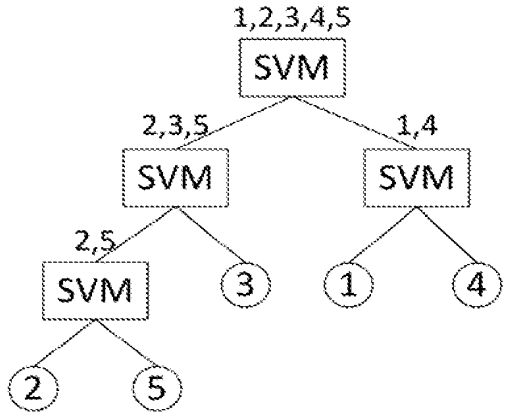


Fig. 4 *Illustration of the multiclass malware classification using a binary tree-based SVM [16]*

After the training phase, the resulting classifier can determine whether a suspicious binary is malicious or not, render the verdict of an unknown malware variant on its belonged family and precisely identify malware instances in the wild. More concretely, the target binary is fed as input to the trained classifier. Then level by level, the classifier computes the similarity of the binary with all the subclasses on the nodes. Finally, the target binary will be classified to one of the malware families or will be determined as benign programme if it does not belong to any of the malware families.

4 Experimental evaluation

4.1 Experimental configuration

In this section, experimental results of the proposed approach are presented on the malware targeting Windows systems. The Windows-based malware is chosen, because the Windows series operation systems are the most commonly used commercial systems. As a result, they have become attractive targets for adversaries who develop malware. In recent years, the Windows-based malware is keeping booming and has outnumbered malware targeting other platforms. In this experiment, we firstly employ 120 typical malware samples to verify the effectiveness of the proposed approach in triggering malicious behaviours within suitable exploration depth. Then, variants of the top-two malware families in 2011 [13] are utilised to demonstrate the mechanism of our detailed techniques. In the comparison with benchmark methods, 100 malware pieces in the wild are sampled to test the performance of the proposed approach in identifying unknown malware instances, where 50 malware instances are equipped with obfuscation techniques to hide their malicious features.

To protect the analysis environment from tampering by malware, our identifier is implemented as two parts: the virtual monitor and the classifier, each of which contains three utilities.

Utilities in the virtual monitor include:

© The Institution of Engineering and Technology 2014

Trend Micro, Inc.
EX1002-28

(1) Multipath execution.
(2) Triggering malicious behaviours.
(3) Recording MBOs.

Utilities in the classifier include:

(1) Building malicious feature matrix.
(2) Training the classifier.
(3) Identifying unknown malware instances.

To achieve a high utility and performance, two separate workloads responsible for the above two parts are implemented, where these two workloads can run on two CPUs or on two cores with the support of multi-cores CPU. Since the analysis is decoupled from the execution, the proposed approach allows more detection techniques to be added on the classification environment without fear of affecting the virtual monitor. Moreover, as the virtual monitor runs on an independent workload, the classifier will be secure when the virtual monitor is compromised by the malware.

All experiments are carried out on machines with Pentium Dual CPU of 2.20 GHz and 2G memory running the simulated/solid (virtual monitor/classifier) Windows XP system.

4.2 Determination of exploration depth

The exploration depth needs to be optimal as it is a key parameter of the proposed approach. For each M_i, its behavioural features are predetermined by expert analysis and thus the number of malicious behaviours beh in L_i(beh, out) is fixed. The depth_{M_i} is figured out as $H(M_i)$ in (5) with 60 typical malware samples as input, where every ten samples represent an MBO and contains all the beh of M_i. Table 2 shows relation between the number of beh and depth_{M_i}. Obviously, half of depth_{M_i} are greater than their corresponding numbers of beh. The reason is that malicious behaviours always hide deeply in the control flows of malware.

Then, the impact of the degree of exploration depth on the rate of triggered malicious behaviours in practice is studied. For each M_i, depths ranged from 1 to 10 are tested. The range of 1–10 is chosen, because it is observed that most malware pieces in the wild hide their malicious behaviours, and in most cases, the depths they apply are <10. To investigate each depth, another 60 malware pieces are fed as input of multipath exploration phase to obtain the numbers of actually triggered malicious behaviours. These 60 samples are evenly split into six groups represented as G_{1-6} corresponding to M_{1-6}, where each sample in a group at least embodies the beh of M_i. For instance, every sample in G_1 contains all the five beh of M_1. As a result, in G_{1-6}, the numbers of beh relative to M_{1-6} are 50 (5 × 10), 40 (4 × 10), 20 (2 × 10), 30 (3 × 10), 30 (3 × 10) and 30 (3 × 10). Since samples in the group G_i only ensures the number of beh of M_i, malicious behaviours recorded in other MBOs may not be detected frequently. The trigger result on each degree of depth can be denoted as a matrix. Fig. 5 takes the results on the degrees of exploration depths equalling 1, 3, 6 and 9 as examples.

In Fig. 5, the data on diagonals represents the numbers of triggered behaviours from samples in G_i relative with M_i. The rest of the data in these matrices delineate the numbers of triggered behaviours recorded in other MBOs, which are not associated with G_i. In Fig. 5*a*, the matrix indicates that rates of triggered behaviours in M_{1-6} are results of numbers of beh in G_{1-6} dividing items on the diagonal, which are 20% (10/50), 25% (10/40), 50% (10/20), 33% (10/30), 33% (10/30) and 33% (10/30). Most of the trigger rates are <50% as the exploration depth is too shallow to trigger malicious behaviours hiding in those obfuscated control flows. In Fig. 5*b*, the trigger rates of beh in M_{1-6} are 64% (32/50), 53% (21/40), 95% (19/20), 70% (21/30), 97% (29/30) and 57% (17/30). Since the exploration depth of 3 has reached some depth_{M_i}, the trigger rates of those beh in relative M_i are rapidly increasing. In Fig. 5*c*, the trigger rates of beh in M_{1-6} soar to 96% (48/50), 98% (39/40), 100% (20/20), 93% (28/30), 100% (30/30) and 100% (30/30), where the exploration depth of 6 satisfies every depth_{M_i}. In Fig. 5*d*, the trigger rates remain steady to 100% (50/50), 98% (39/40), 100% (20/20), 97% (29/30), 100% (30/30) and 100% (30/30), implying that with the deepening of the exploration, very few malicious behaviours are beyond the detection. After experiments on all 10° of depth, rates of triggered malicious behaviours in M_{1-6} are shown in Fig. 6.

Fig. 6 demonstrates that rates of triggered malicious behaviours increase as exploration depths raise; when exploration depths are equal to depth_{M_i} in Table 2, except for M_1(88%), all the trigger rates are ≥90% and do not rise sharply as the depths go deeper. This proves depth_{M_i} is sufficient to trigger malicious behaviours hiding in obfuscated binaries. Although greater exploration depths provide more satisfying trigger rates, the system overhead becomes higher because of more execution paths need to be explored. Therefore depth_{M_i} is employed to trigger malicious behaviours in the multipath exploration.

Table 2 Relation between the number of beh and depth_{M_i}

	M_1	M_2	M_3	M_4	M_5	M_6
number of beh	5	4	2	3	3	3
depth_{M_i}	5	6	2	5	3	4

4.3 Results

In this subsection, we introduce two popular malware families as concrete examples to explain the mechanism of the proposed approach and to demonstrate its effect in identifying emerging malware variants. The examples we use are W32.Ramnit and W32.Sality which are the top-two malware families reported in the latest Symantec threat analysis [13]. Win32.Ramnit and W32.Sality are both virus and worm type malware families that infect multiple Windows executable files, spread to removable drives and open a backdoor to connect to a botnet.

As training samples for the classification process, we first select 50 identified samples of W32.Ramnit from its five generations marked as W32.Ramnit.A–E (ten samples for each generation). Mutually, 50 identified samples of W32.Sality are selected from its five generations.

Then, in the classification process, the proposed approach is employed to model the families of W32.Ramnit and W32.Sality. In the first procedure, as we described in Section 3.1, the group of MBOs are constructed to stand for six typical malware features. In the second procedure, the virtual monitor is equipped to trigger MBOs of the selected 100 samples: for each sample under analysis, the path

© The Institution of Engineering and Technology 2014

doi: 10.1049/iet-ifs.2013.0318
Trend Micro, Inc.
EX1002-29

Triggered behaviours	M_1	M_2	M_3	M_4	M_5	M_6
G_1	10	5	3	0	1	7
G_2	0	10	0	1	0	0
G_3	0	0	10	2	0	0
G_4	0	4	0	10	0	5
G_5	0	11	0	0	10	0
G_6	0	2	0	0	0	10

a

Triggered behaviours	M_1	M_2	M_3	M_4	M_5	M_6
G_1	32	5	3	1	1	7
G_2	0	21	0	2	0	0
G_3	2	0	19	2	3	0
G_4	3	5	0	21	0	5
G_5	0	11	0	0	29	1
G_6	0	2	0	1	0	17

b

Triggered behaviours	M_1	M_2	M_3	M_4	M_5	M_6
G_1	48	5	3	1	1	7
G_2	0	39	1	2	0	0
G_3	2	0	20	3	3	0
G_4	5	5	0	28	0	5
G_5	0	11	1	0	30	2
G_6	0	2	0	1	0	30

c

Triggered behaviours	M_1	M_2	M_3	M_4	M_5	M_6
G_1	50	5	3	1	1	7
G_2	0	39	1	2	0	0
G_3	2	0	20	3	3	0
G_4	5	5	0	29	0	5
G_5	0	11	1	0	30	2
G_6	0	3	0	1	0	30

d

Fig. 5 *Trigger results on exploration depths of 1, 3, 6 and 9*

a Trigger result on exploration depths of 1
b Trigger result on exploration depths of 3
c Trigger result on exploration depths of 6
d Trigger result on exploration depths of 9

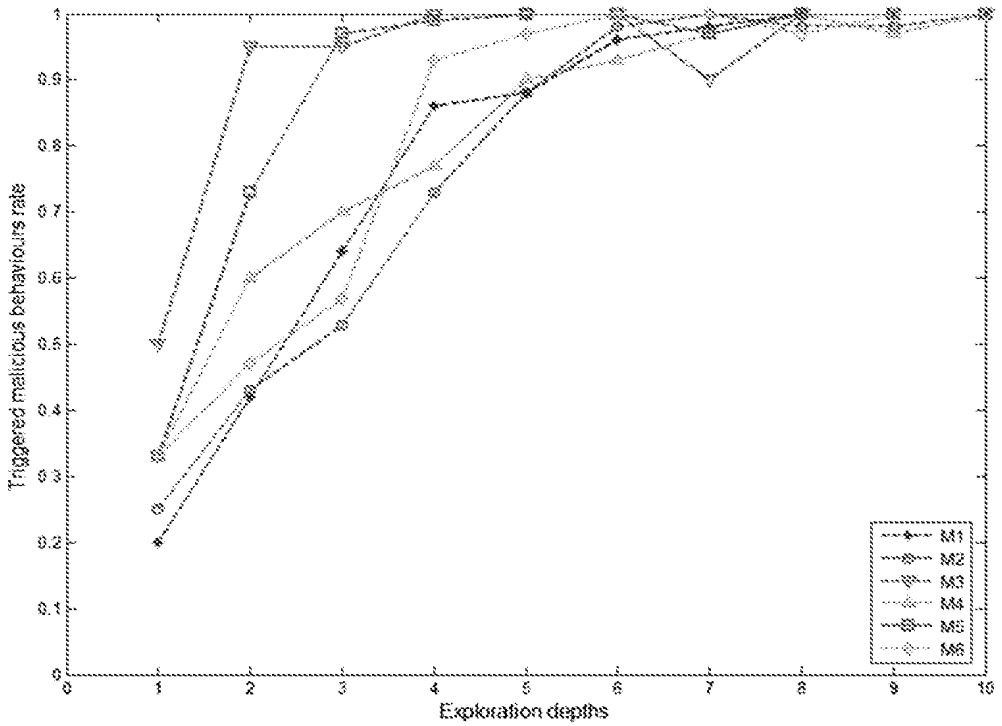


Fig. 6 *Rates of triggered malicious behaviours in M_{1-6} on exploration depths of 1–10*

© The Institution of Engineering and Technology 2013

Trend Micro, Inc.
EX1002-30

explorer parses the instructions of the target binary and triggers malicious behaviours with $depth_{M_i}$ calculated by depth finder.

For the W32.Ramnit samples, triggered malicious behaviours contain: (i) invocation of suspected user APIs and (ii) invocation of suspected native APIs. Monitored malicious outcomes include: (i) creating multiple executable files in all partitions, (ii) injecting malware pieces into other executable files, (iii) creating a new process that contains the malicious binary, (iv) creating an 'auto-start' register entry and (v) connecting to a remote server and receiving data. For the W32.Sality samples, similar MBOs are recorded. It is observed that most triggered MBOs in W32.Ramnit samples are very much like W32.Sality's. However, there is a distinguished pair of behaviour and outcome recorded in W32.Sality – invalid modification of kernel structures and disabling security software process, which can be used as a primary signature to differentiate these two families.

In the last procedure of the classification process, triggered malicious behaviours and their outcomes in the samples are matched with beh and out in MBOs to obtain the malicious degrees w_i. Since there are 50 samples for each of the two families, two 50×6 matrices are built to model for the classification. As the final step, these two matrices are fed to the classifier. The output of this process is two trained models representing the families of W32.Ramnit and W32.Sality, respectively.

In the identification process, our proposed approach is employed to identify 50 new detected variants of W32.Ramnit and another 50 of W32.Sality. The procedures of MBO construction and multipath behaviour triggering are as same as the procedures in the classification process. Consequently, a hundred 1×6 matrices are built to represent all the test variants. In the last procedure, all these matrices are fed as input one by one to the trained classifier, by which the similarity of the variants with the two models is computed.

Table 3 shows the identification results on these 100 test variants. The correct row demonstrates that the proposed approach can identify most of the variants of the two malware families. The missed rate in identifying variants of W32.Sality is relatively high, because polymorphism and other counter-detection techniques are commonly equipped in W32.Sality variants. The low incorrect rates prove the proposed approach can effectively distinguish the dynamic features between these two malware families.

To demonstrate the effectiveness of the proposed approach in identifying variants of newly exposed malware, works [17, 18, 21] are selected to identify the variants of W32.Ramnit and W32.Sality. The experimental samples employed are the 100 test variants showed in Table 3, where 50 samples are new detected variants of W32.Ramnit and another 50 are of W32.Sality. Since in this scenario, the 'correct' rate and the 'incorrect' rate both indicate that the method under test identifies the target malware, where some malware samples may be misidentified as the variants of the other malware family, these two rates are combined as the

Table 3 Identification results on 100 test variants

	Correct	Incorrect	Missed
W32.Ramnit	42(84%)	5(10%)	3(6%)
W32.Sality	40(80%)	2(4%)	8(16%)

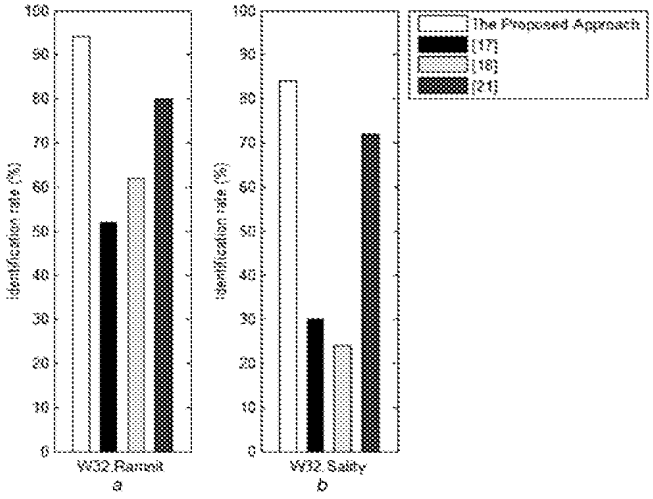


Fig. 7 *Identification rates of the proposed approach and benchmarks on W32.Ramnit and W32.Sality (%)*

a W32.Ramnit
b W32.Sality

'identification' rate which judges if a malware variant is identified by its malicious behavioural features. In the calculation, the 'identification' rate is the sum of 'correct' rate and 'incorrect' rate. For the proposed approach, the 'identification' rates of W32.Ramnit and W32.Sality are 94 and 84% relatively. The full identification rates of the proposed approach and three benchmark methods are illustrated in Fig. 7.

According to the illustration of Fig. 7, the proposed approach achieves higher rates in identifying variants of two up-to-date malware families than the other three benchmark methods. The reason is that the proposed approach employs malicious behaviours with correlated outcomes to determine the maliciousness and belongings of the target variants, reducing false alarms and misclassification. Since the methods [17, 18] are equipped with static pattern matching engines, they have difficulties in identifying obfuscated malware variants. Method [18], especially, relies on traditional static sequential comparing to do pattern matching, so its capability in identifying polymorphic variants of W32.Sality is undermined. Although method [21] achieves a relative high identification rate than the other benchmark methods, it ignores the outcomes of malicious behaviours during identification, and does not have an appropriate data structure for modelling malicious families. As a result, method [21] misclassifies a number of W32.Sailty variants as W32.Ramnit's. In addition, in the real-world scenarios, method [21] may misidentify many benign programs as malware.

4.4 Comparison and analysis

In this subsection, the coverage and performance provided by the proposed approach and the benchmark methods are compared. In all the related malware detection techniques, works [17, 18, 21] are selected to be the benchmark methods, because they are the most representative and are very similar in function with the proposed approach. Since the features of W32.Ramnit and W32.Sality have been recognised and recorded by the proposed approach, it is

© The Institution of Engineering and Technology 2014
IET Inf. Secur., 2014, Vol. 8, Iss. 2, pp. 140–151
doi: 10.1049/iet-ifs.2013.0318
Trend Micro, Inc.
EX1002-31

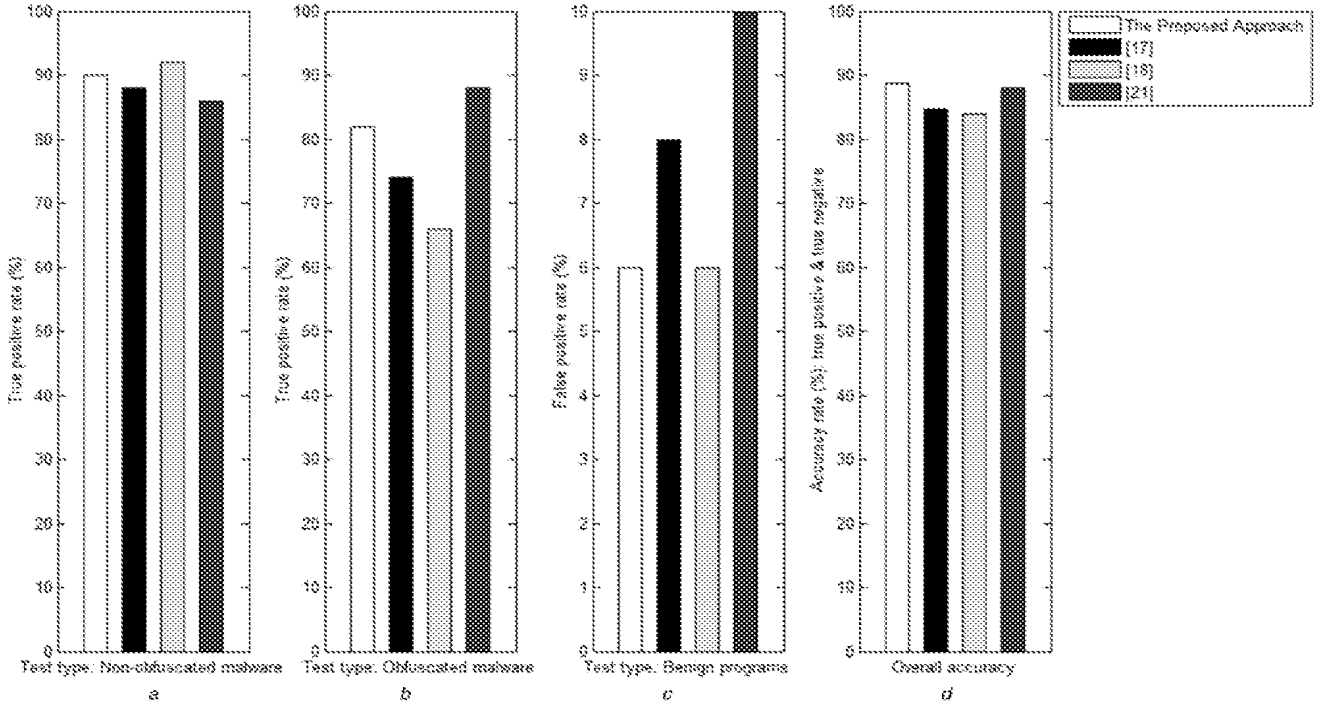


Fig. 8 *Coverage and performance of the proposed approach and the benchmarks (%)*

a Test type: Non-obfuscated malware
b Test type: Obfuscated malware
c Test type: Benign programs
d Overall accuracy

necessary to employ some new malware pieces as the samples. Hundred identified malware instances gathered from VX Heavens [29] are fed as training samples. Another 100 unknown malware pieces in VX database [29] and 50 benign programs are used as test samples, of which 50 malware pieces are obfuscated to hide their features. Since the methods [17, 18] are based on static analysis, signature matching engines are provided for them. The test binaries are analysed by the proposed approach and the methods [17, 18, 21] in a controlled environment.

The comparison results are reported in Fig. 8. As shown in Fig. 8*a*, when analysing the non-obfuscated malware of which the features are relatively easy to recognise, the proposed approach identifies 90.0% of the malware pieces. In contrast, methods [17, 21] only identify 88.0 and 86.0% of the malware pieces, respectively. Although, method [18] obtains an identification rate of 92.0% as it maintains a huge library to do pattern matching. Since the features of obfuscated malware are not as obvious as the non-obfuscated ones, in Fig. 8*b*, the true positive rate of the proposed approach is 82.0%, whereas 18.0% malware pieces are missed, which is a little bit lower than 88.0% of [21]. The reason is that method [21] utilises a more aggressive monitor that ignores malicious outcomes. However, our recognition rate is higher than [17] (74.0%) and [18] (66.0%). In Fig. 8*c*, for benign programs, the false positive rate of the proposed approach is the joint lowest of 6.0% with [18], which is lower than the 8.0% of [17] and the 10.0% of [21]. In Fig. 8*d*, composed of the true positive rates and the true negative rate, the overall accuracy indicates that the proposed approach can provide a better performance of 88.7% in identifying unknown malware instances and filtering out benign programs than the benchmark methods [17, 18, 21], which obtain overall accuracy rate of 84.7, 84.0 and 88.0%, respectively.

The performance comparison showed in Fig. 8 reveals that although the proposed approach achieves the best performance in the overall accuracy, in the individual comparison, some benchmark methods perform as same as or even better than the proposed approach. The major reason is that the malware samples employed in this experiment are selected from the VX Heavens database in which these malware pieces are included throughout years. The benchmark methods may have recorded the features of these VX malware into their own feature database or matching engines, resulting in their better performances in the individual comparisons with the proposed approach. For instance, in Fig. 8*a*, as the test samples are common non-obfuscated malware instances picked from VX Heavens, method [18] that are based on static feature matching may have registered the structural features of these malware during the construction of its feature database. Accordingly, in Fig. 8*a*, method [18] obtains higher true positive rate than the proposed approach. Moreover, in Fig. 8*b*, because method [21] identifies those target binaries as malware merely by judging suspect behaviours without outcomes, it achieves better performance in detecting malware pieces which have been recognised by VX Heavens. Although, in Fig. 8*c*, the aggressive monitor of method [21] leads to a high-false positive rate in filtering out benign programs, where the proposed approach performs as good as method [18] which employs precise code sequence matching. It is acknowledged that the technique of the proposed approach needs more improvements. Especially, in order to increase the identification rates, the group of MBO will be expanded

© The Institution of Engineering and Technology 2014

Trend Micro, Inc.
EX1002-32

and updated to record more malware features in the wild. In addition, for the prototype system, more engineering efforts are required to boost the experimental performances.

5 Discussion

In this section, we discuss the limitations of the proposed approach. There are two main limitations that may counteract the proposed approach.

The first major limitation is because of the use of I/O exceptions and clock interrupts in some malware. Commonly, malware instances do not take advantage of I/O exceptions or clock interrupts, because utilising these two functions may increase the risk in exposing their behaviours. However, some malware pieces downloaded from VX Heavens [29], like the botnet hosts, may utilise physical clock interrupts or I/O exceptions to send heartbeat packages or other messages to their masters. Since in the QEMU virtual machine, all the I/O exceptions and clock interrupts received from the host are implemented by QEMU's software, the software clock interrupts may be less accurate than the physical's. In like manner, some I/O exceptions may not be translated in the source code of QEMU. The solution to this issue would be to add functions in the proposed system to translate and monitor instructions related with I/O exceptions and with clock interrupts. This is practical because all clock interrupts and exceptions made by the physical machine can be simulated by software in QEMU.

The second major limitation results from anti-virtualisation techniques which have been equipped in the 'analysis-aware' malware [30]. The 'analysis-aware' malware can detect the presence of virtual analysis environments such as virtual machines and debuggers. Once the malware realises that it is running on a virtual machine, it either substitutes benign functions for malicious behaviours or simply terminates the execution. Practically, in order to know if it is running on a virtual machine, the malware needs to read and verify some hardware information. For instance, the malware reads some specific field in the BIOS to test if it is being monitored in the QEMU virtual machine. For implementing the above operation we need to write a check code in the memory mapping space with instructions which employ one to two bytes. After receiving the check code, the machine sends the BIOS information to the malware. Since a virtual machine cannot acquire the BIOS information, it is unable to send such information to the malware. Consequently, the malware can be aware of not running in the physical environment. To address this problem, it is necessary for the proposed virtual monitor to capture the check code sent by malware. Since in the QEMU-based system, all instructions are interpreted and then executed by software, the monitored malware has no chance to directly contact with hardware. Thus, the virtual monitor is capable of capturing all the instructions including the check code. After intercepting the check code, the virtual monitor can feedback required information to the malware.

6 Conclusions and future work

In this study, a novel malware identifier based on multipath behaviour triggering is proposed. A group of featured MBOs are established to represent typical malicious features and to weigh the malicious degree of a target binary in each malicious feature. Then, MBOs hiding in binaries are triggered in the virtual monitor with practical exploration depths and breadths. Based on the MBO and the detected malicious behaviours in the virtual monitor, the malicious feature matrix is built for modelling classified malware families and for identifying unknown malware variants. In the experiment, the multipath exploration depths determined by the proposed approach are proved that they can achieve high trigger rates with suitable system consumption. The experimental results and comparison demonstrate that the proposed approach provides higher coverage in identifying unknown malware variants, meanwhile being more reliable in recognising benign programs than benchmark methods.

In the future work, our approach will be improved more specifically to identify malware that counteracts dynamic analysis. To make it more difficult to be detected by malware, the virtual monitor needs to be reconfigured to act more like a real system than an emulator. We also plan to improve the classification process to make an inline analysis which can speedup the identification progress and achieve a real-time detection.

7 Acknowledgments

The authors would like to thank Professor Chun-he Xia for helpful discussions and feedback. The authors would also like to thank the anonymous reviewers for their insightful and helpful comments. This work was supported by the National Nature Science Foundation of China under grant no. 91118007.

8 References

1 Wilhelm, J., Chiueh, T.: 'A forced sampled execution approach to kernel rootkit identification'. Proc. RAID 2007, 2007, pp. 219–235
2 Bayer, U., Moser, A., Kruegel, C., Kirda, E.: 'Dynamic analysis of malicious code', *J. Comput. Virol.*, 2006, **2**, (1), pp. 67–77
3 The IDA Pro Dissassembler and Debugger. Available at http://www.hex-rays.com/idapro/
4 Bai, H., Hu, C., Zhang, G., Jing, X., Li, N.: 'Binary oriented vulnerability analyzer based on hidden markov model', *IEICE Trans. Inf. Syst.*, 2010, **E93-D**, (12), pp. 3410–3413
5 Yetiser, T.: 'Polymorphic viruses-implementation, detection and protection. Available at http://www.vx.netlux.org, 1993
6 Santamarta, R.: 'Generic detection and classification of polymorphic malware using neural pattern recognition. Available at http://www.vx.netlux.org, 2006
7 Szor, P.: 'The art of computer virus research and defense' (Addison Wesley, 2005)
8 Leder, F., Steinbock, B., Martini, P.: 'Classification and detection of metamorphic malware using value set analysis'. Proc. Fourth Int. Conf. Malicious and Unwanted Software (MALWARE), October 2009, pp. 39–46
9 Nair, V.P., Jain, H., Golecha, Y.K., Gaur, M.S., Laxmi, V.: 'MEDUSA: metamorphic malware dynamic analysis using signature from API'. Proc. Third Int. Conf. Security of Information and Networks, 2010, pp. 263–269
10 Preda, M.D.: 'The grand challenge in metamorphic analysis', *Inf. Syst. Technol. Manage.*, 2012, **285**, Part 9, pp. 439–444
11 Moser, A., Kruegel, C., Kirda, E.: 'Limits of static analysis for malware detection'. Proc. 23rd Annual Computer Security Applications Conf. (ACSAC'07), December 2007, pp. 421–430
12 Kirda, E., Kruegel, C.: 'Behaviour-based spyware detection'. Proc. 15th USENIX Security Symp., July 2006, pp. 273–288
13 Symantec Corporation: Internet Security Threat Report. Available at http://www.symantec.com/threatreport, April 2012
14 Tian, D., Zeng, Q., Wu, D., Liu, P., Hu, C.: 'Kruiser: semi-synchronized non-blocking concurrent kernel heap buffer overflow monitoring'. Proc. 19th Network and Distributed System Security Symp. (NDSS 2012), February 2012
15 Bayer, U., Habibi, I., Balzarotti, D., Kirda, E., Kruegel, C.: 'A view on current malware behaviors'. Proc. Second USENIX Conf. Large-scale Exploits and Emergent Threats (LEET'09), February 2009

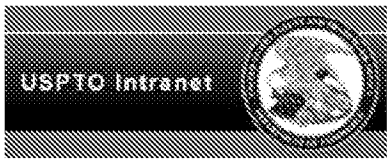
© The Institution of Engineering and Technology 2014

Trend Micro, Inc.
EX1002-33

16 Zhou, G., Cui, Y., Liu, Y., Yang, J.: 'A binary tree structured terrain classifier for Pol-SAR images', *IEICE Trans. Commun.*, 2011, **E94-B**, (5), pp. 1515–1518
17 Kang, B., Kim, H.S., Kim, T., Kwon, H., Im, E.G.: 'Fast malware family detection method using control flow graphs'. Proc. 2011 ACM Symp. Research in Applied Computation, November 2011, pp. 287–292
18 Cha, S.K., Moraru, I., Truelove, J.J., Brumley, D., Andersen, D.: 'SplitScreen: enabling efficient, distributed malware detection'. Proc. Seventh USENIX Symp. Networked Systems Design and Implementation, April 2010
19 Lakhotia, A., Kumar, E.U., Venable, M.: 'A method for detecting obfuscated calls in malicious binaries', *IEEE Trans. Softw. Eng.*, 2005, **31**, (11), pp. 955–967
20 Tabish, S.M., Shafiq, M.Z., Farooq, M.: 'Malware detection using statistical analysis of byte-level file content'. Proc. ACM SIGKDD Workshop on Cyber Security and Intelligence Informatics (CSI-KDD' 09), June 2009, pp. 23–31
21 Jiang, X., Wang, X., Xu, D.: 'Stealthy malware detection and monitoring through VMM-based 'out-of-the-box' semantic view reconstruction', *ACM Trans. Inf. Syst. Secur.*, 2010, **13**, (2), pp. 1–28
22 Kolbitsch, C., Comparetti, P.M., Kruegel, C., Kirda, E., Zhou, X., Wang, X.: 'Effective and efficient malware detection at the end host'. Proc. 18th Conf. USENIX Security Symp. (SSYM'09), August 2009, pp. 351–366
23 Anderson, B., Quist, D., Neil, J., Storlie, C., Lane, T.: 'Graph-based malware detection using dynamic analysis', *J. Comput. Virol.*, 2011, **7**, (4), pp. 247–258
24 Pleszkoch, M., Daly, T., Cohen, C.: 'Automatic generation of hidden markov models for the detection of polymorphic and metamorphic malware, results of SEI independent research and development project (FY2010)' (Carnegie Mellon University, 2011)
25 Bellard, F.: QEMU, a Fast and Portable Dynamic Translator. Available at http://www.bellard.org/qemu
26 Brumley, D., Hartwig, C., Liang, Z., Newsome, J., Song, D., Yin, H.: 'Automatically identifying trigger-based behavior in malware', in Lee, W., Wang, C., Dagon, D. (Eds.): 'Botnet detection. Countering the largest security threat series: advances in information security' (Springer, 2008), vol. 36, pp. 65–88
27 Cover, T.M., Thomas, J.A.: 'Elements of information theory' (Wiley, 1991)
28 Shannon, C.E., Weaver, W.: 'The mathematical theory of communication' (University of Illinois Press, 1949)
29 VX Heavens database. Available at http://www.vx.netlux.org, last updated in March 2012
30 Lalwani, N., Chandak, M.B., Dharaskar, R.V.: 'Split personality malware: a security threat'. IJCA Proc. National Conf. Innovative Paradigms in Engineering and Technology (NCIPET 2012), No. 14, Foundation of Computer Science (FCS), 2012

© The Institution of Engineering and Technology 2014

Trend Micro, Inc.
EX1002-34



PATENTS INTRANET



Scientific and Technical Information Center

Topics | Date | Journals | Authors
Clustered Results

All Topics
- Devices, Security (5)
- Other (15)

Results By Source

- **Computer Source: Consumer Edition (0)**
- **Computers & Applied Sciences Complete (0)**
- **eFunda**
- **IEEE Electronic Library Online (56)**
- **Information Science & Technology Abstracts (0)**
- **Internet and Personal Computing Abstracts**
- **TechWeb**

Sort By: Date | Title | Author | Source Filter By: Full Text

Results 1-20 of 20 returned for "full text contains (pestware OR malware) AND boot AND native AND application AND (monitor OR analyze) AND rules" (56 total)

1 Next Results for All Resources >

Select all records :: 0 selected records: Email or Export/Save

1. **Approach for malware identification using dynamic behaviour and outcome triggering**
Hao Bai; Chang-zhen Hu; Xiao-chuan Jing; Ning Li; Xiao-yin Wang.
Information Security, IET (1751-8709)
2014. Vol.8,Iss.2;p.140-151
Source: **IEEE Electronic Library Online**
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

2. **Design and Implementation of Efficient Integrity Protection for Open Mobile Platforms**
Xinwen Zhang; Seifert, J.-P.; Aciicmez, O.
Mobile Computing, IEEE Transactions on (1536-1233)
2014. Vol.13,Iss.1;p.188-201
Source: **IEEE Electronic Library Online**
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

3.

3.Evolution, Detection and Analysis of Malware for Smart Devices
Suarez-Tangil, G.; Tapiador, J.E.; Peris-Lopez, P.; Ribagorda, A.
Communications Surveys & Tutorials, IEEE (1553-877X)
2014. Vol.16,Iss.2;p.961-987
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

☐ 4.GreenDroid: Automated Diagnosis of Energy Inefficiency for Smartphone Applications
Liu, Y.; Xu, C.; Cheung, S.; Lu, J.
Software Engineering, IEEE Transactions on (0098-5589)
2014. Iss.99;p.1-1
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 5.Growing Grapes in Your Computer to Defend Against Malware
Zhiyong Shan; Xin Wang.
Information Forensics and Security, IEEE Transactions on (1556-6013)
2014. Vol.9,Iss.2;p.196-207
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

☐ 6.Shuttle: Facilitating Inter-Application Interactions for OS-Level Virtualization
Zhiyong Shan; Xin Wang; Tzi-cker Chiueh.
Computers, IEEE Transactions on (0018-9340)
2014. Vol.63,Iss.5;p.1220-1233
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

☐ 7.Thwarting Obfuscated Malware via Differential Fault Analysis
Suarez-Tangil, G.; Tapiador, J.E.; Lombardi, F.; Di Pietro, R.
Computer (0018-9162)
2014. Vol.47,Iss.6;p.24-31
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

☐ 8.Automated Execution Control and Dynamic Behavior Monitoring for Android

(TM) Applications
Ter Louw, M.; Krull, M.; Thomas, T.; Cathey, R.; Frazier, G.; Weber, M.
Military Communications Conference, MILCOM 2013 - 2013 IEEE
2013. p.968-973
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 9.FM 99.9, Radio Virus: Exploiting FM Radio Broadcasts for Malware Deployment
Fernandes, E.; Crispo, B.; Conti, M.
Information Forensics and Security, IEEE Transactions on (1556-6013)
2013. Vol.8,Iss.6;p.1027-1037
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

☐ 10.Security analysis of mobile applications: A case study of a collaboration tool in healthcare
Jang-Jaccard, J.; Li, J.; Nepal, S.; Alem, L.
Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on
2013. p.553-562
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 11.A Survey on Security for Mobile Devices
La Polla, M.; Martinelli, F.; Sgandurra, D.
Communications Surveys & Tutorials, IEEE (1553-877X)
2013. Vol.15,Iss.1;p.446-471
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

☐ 12.Analysis of malicious and benign android applications
Alazab, M.; Moonsamy, V.; Batten, L.; Lantz, P.; Ronghua Tian.
Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on (1545-0678) (978-1-4673-1423-7)
2012. p.608-616
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 13.**Antivirus Software Shield Against Antivirus Terminators**
Fu-Hau Hsu; Min-Hao Wu; Chang-Kuo Tso; Chi-Hsien Hsu; Chieh-Wen Chen.
Information Forensics and Security, IEEE Transactions on (1556-6013)
2012. Vol.7,Iss.5;p.1439-1447
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

☐ 14.**Malware analysis using reverse engineering and data mining tools**
Burji, S.; Liszka, K.J.; Chan, C.
System Science and Engineering (ICSSE), 2010 International Conference on (978-1-4244-6472-2)
2010. p.619-624
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 15.**Analysis of Virtual Machine Applied to Malware Detection System**
Zhang Xiaosong; Pan Xiaohui; Long Xiaoshu.
Information Engineering and Electronic Commerce, 2009. IEEC '09. International Symposium on (978-0-7695-3686-6)
2009. p.290-294
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 16.**Server-side dynamic code analysis**
Guizani, W.; Marion, J.-Y.; Reynaud-Plantey, D.
Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on (978-1-4244-5786-1)
2009. p.55-62
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 17.**A Dynamic Trust Management Scheme to Mitigate Malware Proliferation in P2P Networks**
Xuhua Ding; Wei Yu; Ying Pan.
Communications, 2008. ICC '08. IEEE International Conference on (978-1-4244-2075-9)
2008. p.1605-1609
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 18.**Towards an understanding of anti-virtualization and anti-debugging**

behavior in modern malware
Xu Chen; Andersen, J.; Mao, Z.M.; Bailey, M.; Nazario, Jose.
Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on (978-1-4244-2397-2)
2008. p.177-186
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 19.Back to the Future: A Framework for Automatic Malware Removal and System Repair
Hsu, F.; Hao Chen; Ristenpart, T.; Li, J.; Zhendong Su.
Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual (1063-9527) (0-7695-2716-7)
2006. p.257-268
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 20.Evaluation of Run-Time Detection of Self-Replication in Binary Executable Malware
Volynkin, A.; Skormin, V.A.; Summerville, D.H.; Moronski, J.
Information Assurance Workshop, 2006 IEEE (1-4244-0130-5)
2006. p.184-191
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

*Missing check box indicates resource temporarily unavailable.

*Please obey **MPEP Section 904.02 (c) - Internet Searching [R3]** and USPTO "Rules of the Road (**PDF Doc**)" when using Internet resources.*
If you cannot access a file because of a missing or non-working plugin, please contact the Help Desk at 2-9000 for installation assistance.

Trend Micro, Inc.
EX1002-39

2008 International Conference on Information Security and Assurance

Implicit Detection of Hidden Processes with a Local-Booted Virtual Machine

Yan Wen[1], Jinjing Zhao[2], Huaimin Wang[1]
wenyan@nudt.edu.cn, misszhaojinjing@sina.com.cn, whm_w@163.com
[1] National University of Defense Technology Changsha, China
[2] Beijing Institute of System Engineering, Beijing, China

Abstract

Currently stealth malware is becoming a major threat to the PC computers. Process hiding is the technique commonly used by stealth malware to evade detection by anti-malware scanners. On the defensive side, previous host-based approaches will be defeated once the privileged stealth malware controls a lower reach of the system. The virtual machine (VM) based solutions gain tamper resistance at the cost of losing the OS-level process view. Moreover, existing VM-based approaches cannot introspect the preinstalled OS which is just the protecting concern for PC users. In this paper, we present a new VM-based approach called Libra which accurately reproduces the software environment of the underlying preinstalled OS within the Libra VM and provides an OS-level semantic view of the processes. With our new local-booting technology, Libra VM just boots from the underlying host OS but not a newly installed OS image. Thus, Libra provides a way to detect the existing process-hiding stealth malware in the host OS. In addition, instead of depending on the guest information which is subvertable to the privileged guest malware, Libra adopts a unique technique to implicitly construct the Trusted View of Process List (TVPL) from within the virtualized hardware layer. Our evaluation results with real-world hiding-process rootkits, which are widely used by stealth malware, demonstrate its practicality and effectiveness.

Key words: Hidden process, virtual machine, local-booted virtual machine.

1. Introduction

On PC platforms, with more and more users are willing to download and execute freeware/shareware, potential stealth malware accompanied with the downloaded untrusted code has become a major threat to PC users [1]. The term "stealth malware" refers to the software programs that start up with the sacrificed OS and try to hide their presence from the process enumeration utilities commonly used by malware detectors [2]. *Process hiding* is the most widely used stealth technique. According to statistics released by Microsoft's widely deployed *Malicious Software Removal Tool*, a significant fraction of the malware it encounters and removes consists of stealth components with the capability of process hiding [3]. Consequently, the ability to detect and respond to malicious hidden processes is a clear advantage in the race to prevent the computers against stealth malware.

The most feasible mechanism to detect hidden processes is so-called *cross-view* validation [2]. It operates by observing the process list from two perspectives, *untrusted view* and *trusted view*, and locating the inconsistencies between these two views. Untrusted view is often retrieved from an untrusted, high-level point. Trusted view is obtained from within a lower layer in the system that is unlikely to have been tampered by malware. If a process exists in the trusted view but does not appear in the untrusted view, a cross-view detector can conclude that it has been hidden.

In the defense point of view, cross-view validation will fail in retrieving the "true" process list if an attacker subverts the level from which the trusted view is obtained. So, the key challenge of cross-view validation is the inevitable race that develops between attackers and defenders to control the lowest reaches of a system. Referring to the layer wherein the trusted view of processes is gained, existing cross-view validation approaches can be categorized into two classes: *host-based* (so-called *"in-the-box"*) [3] and *VM-based* (so-called "*out-of-the-box*") [4-7].

The host-based technologies provide the capability of introspecting the host OS with a native, semantic-rich view, whereas they in the meantime make themselves visible and left many chances to privileged malware to dive deeper than the detectors [8-10].

Compared to host-based approaches, VM-based approaches significantly improve the tamper-resistance of detection facilities in virtue of their location in an isolated virtualization layer. However, existing VM-based approaches either suffer from the semantic gaps

978-0-7695-3126-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ISA.2008.22

IEEE computer society

Trend Micro, Inc.
EX1002-44

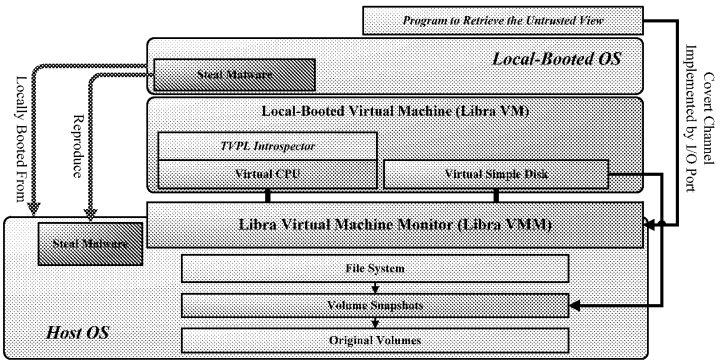


Figure 1. The Architecture of Libra.

between the view of the VM and the view from the inside of the OS [4, 6], or depend on the guest kernel data structures which is still subvertable to the privileged stealth malware [5, 7]. Moreover, the OSes within the VMs don't reproduce the environment of the underlying preinstalled host OSes, which are just our protecting concern on PC platforms. In other words, they only deal with the OS deployed in the VM instead of our daily used host OS.

To address these limitations, we propose a new VM-based cross-view validation approach called Libra for detecting process-hiding malware. Compared to previous VM-based approaches, Libra provides two unique advantages: *host OS environment reproduction*, and *implicit introspection of TVPL*.

Host OS Environment Reproduction. To detect the potential stealth malware existing in the preinstalled host OS at the virtualization layer, Libra introduces a novel *Local-Booted System Virtual Machine* (*Libra VM*). Unlike existing VM-based approaches, Libra VM boots just from the underlying preinstalled host OS instead of booting from a newly installed OS image, viz. Libra VM loads another instance of the host OS. As a consequence, the software environment of host OS, including the possibly existing stealth malware, is faithfully reproduced within this local-booted OS.

Implicit Introspection of TVPL. To decouple from the information subvertable to the privileged guest malware, Libra adopts a novel technique that enable Libra to implicitly discover the "true" process list from within the virtualized hardware layer. By monitoring low-level interactions between OS and the processor memory management unit (MMU), Libra proposes a unique process awareness technology which is capable of accurately determining when an OS creates processes, destroys them, or context-switches between them. With this technology, Libra can detect more process-hiding malware than existing stealth malware detectors.

Libra has been firstly implemented in Windows with Intel x86 processors. Our experimental results with real-world rootkits which are widely used to hide processes demonstrate Libra's unique detection capability.

This paper will discuss the architecture of Libra and focus on the implementation of implicit introspection of TVPL. The rest of the paper is organized as follows. Section 2 describes the design of Libra, followed by the implementation details in Section 3. Section 4 provides the evaluation results of the functionality of hidden process detection. Section 5 reviews previous related works. We summarize the main features of Libra in the last section.

2. Libra Overview

As stated in the previous section, Libra achieves two characteristics: *implicit introspection of TVPL* and *host OS environment reproduction*.

There are two classes of virtual machine monitors (VMM): Type I and Type II [11]. Each of them can serve as the virtualization layer wherein the guest OS is introspected. According to the definition of Goldberg [11], a Type I VMM just runs above a bare computer hardware platform. It tends to be implemented as a lightweight OS with the virtualization capabilities. A Type II VMM is executed as an application. The OS that manages the real computer hardware is called the "host OS". Every OS that runs in the Type II virtual machine is called a "guest OS". In a Type II VMM, the host OS provides process allocation and a standard execution environment to each guest OS.

Trend Micro, Inc.
EX1002-45

Unlike the mainframes that are configured and managed by experienced administrators, desktop PC's are often preinstalled with a standard OS and managed by the end-user. Ignoring the difficulty of proposing a practical and seamless migration approach for PCs, it will maybe take several years to migrate all of them to the Type I VMM. It also might be unacceptable for a PC user to completely replace an existing OS with a Type I VMM. In contrast, Type II VMM allows co-existing with the preinstalled host OS and programs.

Accordingly, taking into account that the preinstalled OS of PC platforms is the prime protecting concern for Libra, we select Type II VMM over Type I VMM to act as the introspecting layer. As illustrated in Figure 1, Libra is composed of three key components: *Libra Virtual Machine (Libra VM)*, *Virtual Simple Disk*, and *TVPL Introspector*.

Libra Virtual Machine (Libra VM). It is a novel local-booted virtual machine built on *Libra VMM* (a Type II VMM). With our local-booting technology, the *Local-Booted OS* started in *Libra VM* reproduces accurately the software environment of *Host OS*, along with the potential stealth malware. This feature makes Libra hold predominance over previous VM-based solutions with the capability of examining the preinstalled host environment at virtualization layer.

Virtual Simple Disk Based on Volume Snapshot. A key challenge to accomplish local-booting VM is reusing the existing system volume wherein *Host OS* has been preinstalled. While *Libra VM* is running, the *Local-Booted OS* is unaware of the modifications made by *Host OS* and vice versa. So they likely crash because of the content inconsistency between the file system and disk drive. Libra resolves these conflicts by dint of the *Virtual Simple Disk* based on *Volume Snapshot*. *Volume Snapshot* introduces *Copy-on-Write* mechanism to shield the modification effects of *Host OS* from *Libra VM* and vice versa. *Virtual Simple Disk* acts as the virtual storage device to combine and export the *Volume Snapshots* to *Libra VM*. Thus, *host OS environment reproduction* is achieved.

TVPL Introspector. To retrieve the "true" process list from within the lowest reaches of a system, Libra implements four introspectors within the virtualized hardware devices layer. *Libra Process Introspector* (*LPI*) is located in the *Virtual CPU*.

Assumption on trusted Libra VM. In this paper, we make an assumption that the stealth malware in host OS will not tamper the Libra program. This is practical because the ultimate target of the stealth malware is hiding its presence and extending its lifetime as long as possible. However, subverting a closed-source program often means unstable reverse-engineering and likely crashes the sacrificed program. Examining existing popular stealth malware programs, all of them try to attack the protection mechanism used by the stealth malware detectors instead of tampering these detectors to evade from detection at the risk of being exposed. Note that this assumption is consistent with that of all of the security research efforts which focuses on a compromised computer system, such as anti-virus programs, anti-rootkits software and so on.

3. Implementation of Libra

The Libra architecture discussed in Section 2 is OS-independent. Considering Windows and Intel processors are prevalent on PC platforms, Libra has been firstly implemented in Windows with Intel x86 processors.

To accomplish the two features of Libra, we are faced with two challenges: *local-booting technology with Intel x86 processors virtualization* and *implicit introspection of TVPL*. The detailed description about how we implement the local-booted VM for Intel x86 processors has been presented in our previous work [12]. This section focuses on the implementation of *implicit introspection of TVPL*.

Intel x86 processors use a two-level, in-memory, architecturally-defined page table. The page table is organized as a tree with a single fixed-sized, commonly 4KB, memory page called the *page directory* at its root. A single address space is active per processor at any given time. The address of the page directory is therefore characteristic of a single address space.

Consequently, we use the physical address of the page directory as the process unique identifier (*PUID*). OS informs the processor's memory management unit (MMU) that a new address space should become active by writing the physical address of the page directory for the new address space into a specific control register - *CR3 register*. Within the local-booted VM, access to the CR3 register is privileged and Libra VMM will catch this event and emulate its semantic. By tracking virtual address space creation and destruction within Libra VMM, *LPI* implicitly obtains the information about the events of process creations and exits, namely the *TVPL*.

LPI maintains TVPL using a Black-Red tree. Each node indexed by PUID refers to a "true" running process in the OS. The key to maintain the TVPL is analyzing the process creation and destruction.

Process Creation. The creation of a process always refers to a new address space represented by a new *PUID*. If we observe a PUID value being writing to CR3 that has not been seen in the TVPL, we can infer that a new address space has been created. Thus, LPI will fetch the process information and insert it along with its PUID into the TVPL.

Process Destruction. Our approach to detect process destruction stems from the mechanism used by an OS to

Trend Micro, Inc.
EX1002-46

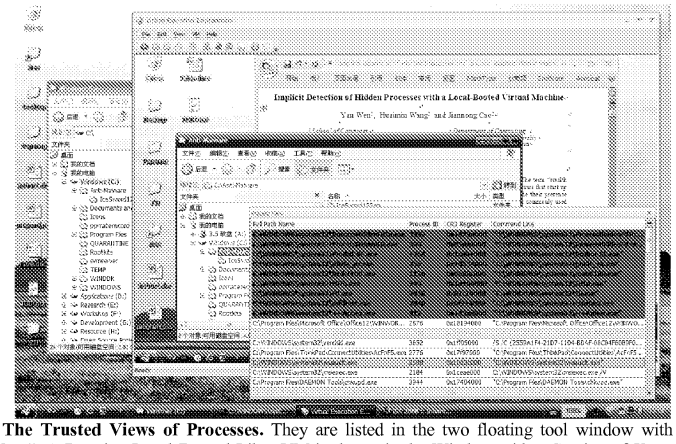


Figure 2. The Trusted Views of Processes. They are listed in the two floating tool window with the title of "*Process View*". A Running Local-Booted Libra VM is shown in the Window with a Caption of *Virtual Execution Environment*. The resolution of Local-Booted Windows is 1024x600 while the resolution of host Windows is 1280x800, so the icon arrangement within its desktop differs from that of Host Windows. As shown in this Figure, the programs of *Explorer* and *Word 2007* are running in this *Local-Booted Windows*. The topmost window is Explorer which is browsing the volume *C:*. The detected hidden processes are highlighted with red background.

deallocate the address space. Once a process has exited, Windows (and Linux) systematically will clear the non-privileged portions of its page table pages used prior to reusing them. Privileged portions of the pages used to implement the protected kernel address space need not be cleared because they are shared between processes and map memory not accessible to user-mode programs. But the non-paged memory pages among these un-cleared privileged pages must be set un-present (a bit in a page entry in the page table).

So, to detect the address space deallocations, LPI sends a request to the memory manger of Libra VMM to remap the address of a specific non-paged kernel-mode memory page for each process in TVPL. If the remapped page is set un-present, the process should have been destructed. In this case, LPI will remove the tree node of this process from TVPL.

Picking a kernel-mode non-paged memory page for each process is an OS-dependent issue. This address of this page must satisfy three requirements: 1) being assigned as soon as a process is created; 2) being fixed among a process's lifeline; 3) being accessible among a process's lifeline; 4) being unsubvertable for privileged malware.

On Windows platforms, we select the memory page containing the *Executive Process Block (EPROCESS)* which meets all of the four requirements and consists of all the most important control information about a process. The address of *EPROCESS* is stored in the *Executive Thread Block (ETHREAD)* while the *ETHREAD* address of an active thread (0xFFDFF000+ 0x124) is immutable on x86 Windows platforms. Tampering the *EPROCESS* entry in *ETHREAD* structure will break up the association between a thread and its process, or even crash the Windows kernel.

Considering the peculiarity of Windows, only if the active PUID refers to ***System*** process, we are supposed to do the remapping operations because Windows will schedule **System** process to synchronize some kernel data structures after a process exits. Consequently, we significantly decrease the count of the address remapping operations.

To construct the untrusted view of processes, we call *ZwSetSystemInformation*, a Windows native API, with a process information related parameter termed of *SystemProcessesAndThreadsInformation* to enumerate all the processes. This is the most common API to list the running processes.

4. Evaluation

We have evaluated the local-booting capability of Libra on several desktop PCs with different hardware equipments[12]. In this section, to demonstrate the new implicit process introspection capability, we will evaluate our prototype with various real-world process-

Trend Micro, Inc.
EX1002-47

Table 1 Completeness of Process List Introspection. The three columns refer to respective test results in three Windows versions: Windows 2000 SP4, Windows XP SP2 and Windows Server 2003 SP1.

	2000	**XP**	**2003**
Process Creation	2200	2200	2200
Address Space Creation	2200	2200	2200
Detected by Libra	2200	2200	2200
Process Exit	2200	2200	2200
Address Space Destruction	2200	2200	2200
Detected by Libra	2200	2200	2200

hiding rootkits which are widely used by stealth malware. Each of them adopts a different representative process-hiding technique. For a desktop-oriented workload, the performance evaluation results show that Libra only suffers a slowdown over native of 0.62-6.28%, with a 2.18% average slowdown for *Libra VMM* [12].

Hidden Process Detection. A running Libra is shown in Figure 2. This sub-section will describe the test set we used to evaluate Libra at length.

Firstly, we quantify the completeness of the implicit TVPL introspection. Our evaluation uses multiple versions of uncompromised Windows OSes which have been instrumented to report process creation, exit, and context switch. Table 1 reports the process and address space event counts gathered by the OS interfaces and by Libra during an experiment utilizing a process intensive workload. The workload creates 2200 processes, each of which runs for 25 seconds then exits. We create 10 processes per second using the *CreateProcess* API. As shown in Table 1, in this tested case, all process-related events reported by our instrumented OSes are detected by the Libra. Furthermore, Libra can precisely detect address space events, indicating that there is a one-to-one match between address spaces and processes for these OSes.

We then evaluate Libra with five popular process-hiding rootkits. *Aphex* [13] modifies the Import Address Table entry for the *ZwQuerySystemInformation* API to intercept the process list queries . *Hacker Defender* [14] also hijacks the queries with another technique: tempering the first few machine codes of *NtQuerySystemInformation* with a "jmp" instruction. *FU* [15] hides a process by using so-called *Direct Kernel Object Manipulation* (*DKOM*) technique to remove its corresponding entry from the *Active Process List* (a kernel data structure in Windows). *FUTo* [8] (an improved version of the *FU* rootkit) has the added ability to manipulate the *PspCidTable* without using any function calls. *PE386* proposed another powerful DKOM-based process-hiding rootkit called *phide_ex* [16] which have bypassed existing hidden process detectors.

We compare Libra to ten anti-rootkit programs which are recommended by Anti Rootkit Group. As summarized by Table 2, the first two rootkits, *Aphex*, *Hacker Defender*, can be found out by using the *Active Process List* as the truth. *FU* can be detected by all the above anti-rootkit programs while *Icesword* and *F-Secure Blacklight* fail to find the FUTo-hiding processes. Libra can detect all the processes hidden by the above five rootkits (shown in Figure 2). Evaluation results show that Libra is the only detector which can identify the hidden process of *phide_ex* (*phide_ex.exe*).

5. Related Work

Cross-view validation for hiding resource detection has been proposed and variously implemented in user-mode applications, within the OS kernel [2], inside a VMM [7]. The key difference of cross-view validation is the mechanism used to obtain the trusted view.

The VM introspection methodology is pioneered by the Livewire system [7]. VM introspection in Livewire is capable of examining low-level VM states from outside the VM. Livewire uses the explicit OS

Table 2. Experimental Results for Libra Hidden Processes Detection. *D* refers to the tool has detect the hidden process successfully while *F* denotes failure of detecting.

	Aphex	*Hacker Defender*	*FU*	FUTo	*phide_ex*
Blacklight	D	D	D	**F**	**F**
DarkSpy	D	D	D	**D**	**F**
IceSword	D	D	D	**F**	**F**
RkUnhooker	D	D	D	**D**	**F**
Bitdefender Antirootkit	D	D	D	**D**	**F**
UnHackMe	D	D	D	**F**	**F**
GMER	D	D	D	**D**	**F**
KProcCheck	D	D	D	**D**	**F**
Process Hunter	D	D	D	**F**	**F**
TaskInfo	D	D	D	**F**	**F**
Libra	D	D	D	**D**	**D**

Trend Micro, Inc.
EX1002-48

debugging information like the memory addresses of variables and the layout and semantics of compound structures to locate and interpret private kernel data types at runtime. VMwatcher [5] is another VM-based guest view introspector. It still highly depends upon the subvertable guest memory contents. As mentioned in literature [5], without the knowledge of the subverted scheduler, VMwatcher is not able to accurately identify all running processes.

VM-level services based on explicit implementation information are effective, but there are drawbacks. One drawback is that they may be just as susceptible to evasion by an attacker that has subverted the guest OS as if they were located within the guest itself. In spite of their location at the VM-layer, these services depend on guest-level information which is still open to simple guest-level manipulation. Instead, the introspectors of Libra are independent of the guest-level information.

Besides, desktop PCs, the concern of our approach, are often preinstalled with a standard OS. However, existing VM-based approaches only deal with the newly deployed OS within the VM. They have less contribution on detecting the existing stealth malware in the existing OSes. In contrast, Libra accurately reproduced the software environment of the host OS within Libra VM and provide a way to introspect it from within a virtualization layer.

6. Conclusion

Stealth malware are a current and alarming security issue. In this paper, we have presented a new VM-based approach called Libra to detect hidden processes implicitly. Like previous VM-based security services, Libra is resilient to kernel-mode guest malware attack by virtue of its location within a VMM layer.

Unlike prior VMM-layer process hiding detectors, by introducing a novel local-booting technology, Libra achieves the unique capability of introspecting the preinstalled host OS at virtualization layer. Moreover, Libra adopts a novel implicit process introspection technology which is decoupled from the subvertable guest OS information. Our evaluation results with real-world stealth rootkits prove Libra's detection capability.

Acknowledgements

This research is supported by National Basic Research Program of China under Grant No. 2005CB321801, National High-Tech Research and Development Plan of China under Grant No. 2007AA010301, and National Science Fund for Outstanding Youths under Grant No. 60625203.

References

[1] "Zombie PCs: Silent, Growing Threat. PC World," http://www.pcworld.com/news/article/0,aid,116841,00.asp, July 2004.

[2] Y.-M. Wang, D. Beck, B. Vo, R. Roussev, and C. Verbowski, "Detecting Stealth Software with Strider GhostBuster," in *Proceedings of 35th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'05)*, 2005, pp. 368–377.

[3] R. Naraine, "Microsoft: Stealth Rootkits Are Bombarding XP SP2 Boxes," http://www.eweek.com/article2/0,1895,1896605,00.asp, December 2005.

[4] A. Joshi, S. T. King, G. W. Dunlap, and P. M. Chen, "Detecting Past and Present Intrusions through Vulnerability-Specific Predicates," in *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP'05)*, Brighton, United Kingdom, 2005, pp. 91 - 104.

[5] X. Jiang, X. Wang, and D. Xu, "Stealthy Malware Detection Through VMM-Based "Out-of-the-Box" Semantic View Reconstruction," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*, 2007.

[6] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen, "ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay," in *Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[7] T. Garfinkel and M. Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection," in *Proceedings of Network and Distributed System Security Symposium (NDSS'03)*, 2003.

[8] P. Silberman and C.H.A.O.S, "FUTo: Bypassing Blacklight and IceSword," https://www.rootkit.com/newsread.php?newsid=433, 2007.

[9] "Effective file hiding : Bypassing Raw File System I/O Rootkit Detector," http://www.rootkit.com/newsread.php?newsid=690.

[10] "Bypassing Klister 0.4 with No Hooks or Running a Controlled Thread Scheduler," http://hi-tech.nsys.by/33/.

[11] R. P. Goldberg, "Architectural Principles for Virtual Computer Systems, Ph.D. Thesis," Cambridge, MA: Harvard University, 1972.

[12] Y. Wen, J. Zhao, and H. Wang, "A Novel Approach for Untrusted Code Execution," in *Proceedings of 9th International Conference on Information and Communications Security (ICICS'07)*, 2007, pp. 398-411.

[13] "Aphex: AFX Windows Rootkit 2003," http://www.iamaphex.cjb.net.

[14] "Hacker Defender," http://hxdef.org/.

[15] fuzen_op, "FU Rootkit," http://www.rootkit.com/project.php?id=12.

[16] PE386, "phide_ex -untimate process hiding example," http://forum.sysinternals.com/printer_friendly_posts.asp?TID=8527.

Trend Micro, Inc.
EX1002-49

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/460,655	04/30/2012	Jerome L. Schneider	6890-45-CON	9502

23552 7590 09/03/2014

MERCHANT & GOULD PC
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903

EXAMINER	
LANE, GREGORY A	

ART UNIT	PAPER NUMBER
2438	

NOTIFICATION DATE	DELIVERY MODE
09/03/2014	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

USPTO23552@merchantgould.com

Trend Micro, Inc.
EX1002-50

Corrected Notice of Allowability	**Application No.** 13/460,655	**Applicant(s)** SCHNEIDER, JEROME L.	
	Examiner GREGORY LANE	**Art Unit** 2438	**AIA (First Inventor to File) Status** No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *1/22/2014*.
   ☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on_____.
2. ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.
3. ☒ The allowed claim(s) is/are *1,2,4-8,10-14 and 16-23*. As a result of the allowed claim(s), you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov .
4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

   Certified copies:

   a) ☐ All b) ☐ Some *c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).
6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☒ Notice of References Cited (PTO-892)
2. ☐ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
3. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
4. ☒ Interview Summary (PTO-413), Paper No./Mail Date *4/14/2014* .
5. ☒ Examiner's Amendment/Comment
6. ☒ Examiner's Statement of Reasons for Allowance
7. ☐ Other _____.

/GREGORY LANE/
Examiner.Art Unit 2438

Trend Micro, Inc.
EX1002-51

The present application is being examined under the pre-AIA first to invent provisions.

DETAILED ACTION

1. The text of those sections of Title 35 U.S. Code not included in this section can be found in the prior office action.
2. The prior office actions are incorporated herein by reference. In particular, the observations with respect to claim language, and response to previously presented arguments.
3. Claims 1, 4, 7, 8, 10, 13, 16, and 20 are amended
4. Claims 3, 9, and 15 are cancelled
5. Claims 21-23 are new
6. Claims 1, 2, 4-8, 10-14, and 16-23, now re-numbered 1-20, are pending.

EXAMINER'S AMENDMENT

7. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Brent Lindon on April 14, 2014.

8. **CLAIMS:**

Please amend claims as follows:

1. (Currently Amended) A method for managing pestware on a computer comprising:

monitoring events during a boot sequence of the computer;

managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a subsystem of an operating system is loaded, and after a kernel is loaded;

managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;

generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;

analyzing the record of events so as to identify the pestware-related events;

modifying the set of behavior rules so as to prevent the pestware related events; and

scanning data in a registry of the computer for pestware during the second period in the boot sequence.

3. (Canceled)

4. (Currently Amended) The method of ~~claim 3~~claim 1, wherein the record of events includes information selected from ~~the~~ a group consisting of: process identification information, file identification information, and hook generation information.

7. (Currently Amended) A system for managing pestware on a computer comprising:

a processor;

a memory comprising:

a first instruction set configured to monitor events during a boot sequence of the computer;

a second instruction set configured to manage pestware-related events during first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a WIN32 subsystem is loaded, and after a kernel is loaded;

a third instruction set configured to manage pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;

a fourth instruction set configured to generate, in response to the monitoring, a record of events, the record of events including the pestware-related events;

a fifth instruction set configured to analyze the record of events so as to identify the pestware-related events; and

a sixth instruction set configured to modify the set of behavior rules so as to prevent the pestware related events; and

a ~~fourth~~ seventh instruction set configured to scan data in a registry of the computer for pestware during the second period in the boot sequence.

8. (Currently Amended) The system of claim 7, wherein the memory further comprises ~~a fifth~~an eighth instruction set configured to launch, after an operating system is initiated, a pestware management engine; and wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

9. (Canceled)

10. (Currently Amended) The system of ~~claim 9~~claim 1, wherein the record of events includes information selected from ~~the~~ a group consisting of: process identification information, file identification information, and hook generation information.

13. (Currently Amended) A non-transitory tangible computer-readable storage medium comprising program instructions for:

monitoring events during a boot sequence of a computer;

managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a WIN32 subsystem is loaded, and after a kernel is loaded;

managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;

generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;

analyzing the record of events so as to identify the pestware-related events;

modifying the set of behavior rules so as to prevent the pestware related events; and

scanning data in a registry of the computer for pestware during the second period in the boot sequence.

15. (Canceled)

16. (Currently Amended) The non-transitory tangible computer-readable storage medium of ~~claim 15~~claim 13, wherein the record of events includes information selected from ~~the~~ a group consisting of: process identification information, file identification information, and hook generation information.

20. (Currently Amended) The method of claim ~~3~~ 1, further comprising modifying the set of behavior rules based on the scanned registry so as to prevent the pestware-related events.

21. (New) The method of claim 1, wherein the pestware-related invents include events associated with at least one of loading pestware, executing pestware, and one or more behaviors of pestware.

22. (New) The system of claim 7, wherein the pestware-related invents include events associated with at least one of loading pestware, executing pestware, and one or more behaviors of pestware.

23. (New) The non-transitory tangible computer-readable storage medium of claim 13, wherein the pestware-related invents include events associated with at least one of loading pestware, executing pestware, and one or more behaviors of pestware.

Response to Arguments

9. Applicant's arguments filed on January 22, 2014 have been fully considered and they are persuasive.

Allowable Subject matter

10. Claims 1, 2, 4-8, 10-14, and 16-23 are allowed over prior art of record.

Examiner's Statement of Reasons for Allowance

11. The following is an examiner's statement of reasons for allowance:
Newly amended independent claims 1, 7, and 13 are allowed in view of the examiner's amendment and for reasons argued by applicant in pages 8-11 of the Applicant's remarks, filed January 22, 2014 and for filing a Terminal Disclaimer on April 17, 2014. The prior art of record do not teach monitoring pestware related operations during a first period in a boot sequence before native applications are running and after the kernel is loaded and before a subsystem of an operating system is loaded. Managing pestware

related events in accordance with a set of behavior rules, pestware related events during a second period in the boot sequence when native applications are running and generating a file containing a list of detected pestware related events. Subsequently, scanning the registry of the computer for the listed pestware related events during the boot sequence.

None of the prior art of record, either taken by itself or in any combination, would have anticipated or made obvious the invention of the present application at or before the time it was filed. The dependent claims, 2, 4-6, 8, 10-12, 14, and 16-23 are also allowable by virtue of their dependence on the independent claims.

Any comments considered necessary by applicant must be submitted no later than payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled “Comments on Statement of Reasons for Allowance."

CONCLUSION

Any inquiry concerning this communication or earlier communications from the examiner should be directed to GREGORY LANE whose telephone number is (571)270-7469. The examiner can normally be reached on 571 270 7469 from 8:00 AM to 6:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Taghi Arani, can be reached on 571 272 3787. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/GREGORY LANE/ Examiner, Art Unit 2438

/MADHURI HERZOG/

Primary Examiner, Art Unit 2438

Examiner-Initiated Interview Summary	Application No.	Applicant(s)	
	Examiner	Art Unit	

All participants (applicant, applicant's representative, PTO personnel):

(1) *GREGORY LANE*. (3)_____.

(2) *BRENT LINDON*. (4)_____.

Date of Interview: *12 August 2014*.

Type: ☒ Telephonic ☐ Video Conference
☐ Personal [copy given to: ☐ applicant ☐ applicant's representative]

Exhibit shown or demonstration conducted: ☐ Yes ☐ No.
If Yes, brief description: _____.

Issues Discussed ☐101 ☐112 ☐102 ☐103 ☐Others
(For each of the checked box(es) above, please describe below the issue and detailed description of the discussion)

Claim(s) discussed: *1,3,7,9,13 and 15*.

Identification of prior art discussed: _____.

Substance of Interview
(For each issue discussed, provide a detailed description and indicate if agreement was reached. Some topics may include: identification or clarification of a reference or a portion thereof, claim interpretation, proposed amendments, arguments of any applied references etc...)

Applicant agreed to amend independent claims 1,7, and 13 with limitations from dependent claims 3, 9, and 15 and to file a terminal disclaimer in order to place application in condition for allowance. Also, notified applicant that a corrected notice of allowance was being issued..

Applicant recordation instructions: It is not necessary for applicant to provide a separate record of the substance of interview.

Examiner recordation instructions: Examiners must summarize the substance of any interview of record. A complete and proper recordation of the substance of an interview should include the items listed in MPEP 713.04 for complete and proper recordation including the identification of the general thrust of each argument or issue discussed, a general indication of any other pertinent matters discussed regarding patentability and the general results or outcome of the interview, to include an indication as to whether or not agreement was reached on the issues raised.

☒ Attachment

/GREGORY LANE/ Examiner, Art Unit 2438	

Trend Micro, Inc.
EX1002-59

Notice of References Cited	Application/Control No. 13/460,655	Applicant(s)/Patent Under Reexamination SCHNEIDER, JEROME L.	
	Examiner GREGORY LANE	Art Unit 2438	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-			
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Wen, "Implicit Detection of Hidden Processes with a Local-Booted Virtual Machine", 2008, IEEE, pg. 150-155 Bai, "Approach for Malware Identification using Dynamic Behavior and Outcome Triggering", 2014, IEEE, Vol. 8, pg. 140-151
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13460655	SCHNEIDER, JEROME L.
	Examiner	Art Unit
	GREGORY LANE	2438

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☒ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	10/07/2013	04/17/2014	08/14/2014						
1	1	✓	=	=						
2	2	✓	=	=						
	3	✓	-	-						
3	4	✓	=	=						
4	5	✓	=	=						
5	6	✓	=	=						
6	7	✓	=	=						
7	8	✓	=	=						
	9	✓	-	-						
8	10	✓	=	=						
9	11	✓	=	=						
10	12	✓	=	=						
11	13	✓	=	=						
12	14	✓	=	=						
	15	✓	-	-						
13	16	✓	=	=						
14	17	✓	=	=						
15	18	✓	=	=						
16	19		=	=						
17	20		=	=						
18	21		=	=						
19	22		=	=						
20	23		=	=						

Trend Micro, Inc.
EX1002-61

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13460655	SCHNEIDER, JEROME L.
	Examiner	**Art Unit**
	GREGORY LANE	2438

CPC							
Symbol						**Type**	**Version**
G06F		21	/	566		F	2013-01-01
G06F		21	/	56		I	2013-01-01
G06F		21	/	554		I	2013-01-01
			/				
			/				
			/				
			/				
			/				
			/				
			/				
			/				
			/				
			/				
			/				

CPC Combination Sets								
Symbol					**Type**	**Set**	**Ranking**	**Version**
			/					
			/					

/GREGORY LANE/ Examiner.Art Unit 2438	08/12/2014	**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	20	
/MADHURI HERZOG/ Primary Examiner.Art Unit 2438	08/14/2014	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13460655	SCHNEIDER, JEROME L.
	Examiner	**Art Unit**
	GREGORY LANE	2438

US ORIGINAL CLASSIFICATION		INTERNATIONAL CLASSIFICATION											
CLASS	SUBCLASS	CLAIMED					NON-CLAIMED						
713	2	G	0	6	F	21 / 00 (2013.01.01)							
CROSS REFERENCE(S)													
CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)												
713	193												

/GREGORY LANE/ Examiner.Art Unit 2438	08/12/2014	**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	20	
/MADHURI HERZOG/ Primary Examiner.Art Unit 2438	08/14/2014	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13460655	SCHNEIDER, JEROME L.
	Examiner	**Art Unit**
	GREGORY LANE	2438

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☒ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	17	20												
2	2	18	21												
3	4	19	22												
4	5	20	23												
5	6														
6	7														
7	8														
8	10														
9	11														
10	12														
11	13														
12	14														
13	16														
14	17														
15	18														
16	19														

/GREGORY LANE/ Examiner.Art Unit 2438	08/12/2014	**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	20	
/MADHURI HERZOG/ Primary Examiner.Art Unit 2438	08/14/2014	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13460655	SCHNEIDER, JEROME L.
	Examiner	Art Unit
	GREGORY LANE	2438

CPC- SEARCHED		
Symbol	Date	Examiner
G06F 21/56	4/17/2014	GL
G06F 21/566	4/17/2014	GL
G06F 21/554	4/17/2014	GL

CPC COMBINATION SETS - SEARCHED		
Symbol	Date	Examiner

US CLASSIFICATION SEARCHED			
Class	Subclass	Date	Examiner
713	2,193	10/7/2013	GL

SEARCH NOTES		
Search Notes	Date	Examiner
searched for the inventive concept of kernel level malware	10/7/2013	GL
searched the inventors' names to check for double patenting violation	10/7/2013	GL
considered 35 USC 101 and 112 violations	10/7/2013	GL
Performed EAST search	10/7/2013	GL
consulted Taghi Arani, SPE AU2438	4/9/2014	GL
updated EAST search using CPC classification	4/17/2014	GL
updated NPL search	4/17/2014	GL
updated EAST and NPL searches	8/12/2014	GL

INTERFERENCE SEARCH			
US Class/ CPC Symbol	US Subclass / CPC Group	Date	Examiner

/GREGORY LANE/ Examiner.Art Unit 2438	/MADHURI HERZOG/ Primary Examiner.Art Unit 2438

Trend Micro, Inc.
EX1002-65

INTERFERENCE SEARCH			
US Class/ CPC Symbol	**US Subclass / CPC Group**	**Date**	**Examiner**
713	2,193 AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	8/12/2014	GL
G06F	21/56 AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	8/12/2014	GL
G06F	21/566 AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	8/12/2014	GL
G06F	21/554 AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	8/12/2014	GL

/GREGORY LANE/ Examiner.Art Unit 2438	/MADHURI HERZOG/ Primary Examiner.Art Unit 2438

Trend Micro, Inc.
EX1002-66

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

22442 7590 04/29/2014
Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission
I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/460,655	04/30/2012	Jerome L. Schneider	6890-45-CON	9502

TITLE OF INVENTION: Malware Management Through Kernel Detection During a Boot Sequence

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	SMALL	$480	$0	$0	$480	07/29/2014

EXAMINER	ART UNIT	CLASS-SUBCLASS
LANE, GREGORY A	2438	713-002000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
- ☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- ☑ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list
(1) The names of up to 3 registered patent attorneys or agents OR, alternatively,
(2) The name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 Merchant & Gould P.C.
2
3

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: Webroot Inc.

(B) RESIDENCE: (CITY and STATE OR COUNTRY): Broomfield, CO

Please check the appropriate assignee category or categories (will not be printed on the patent): ☐ Individual ☑ Corporation or other private group entity ☐ Government

4a. The following fee(s) are submitted:
- ☑ Issue Fee
- ☐ Publication Fee (No small entity discount permitted)
- ☐ Advance Order - # of Copies .....

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**
- ☐ A check is enclosed.
- ☑ Payment by credit card. Form PTO-2038 is attached.
- ☑ The Director is hereby authorized to charge the required fee(s), any deficiency, or credits any overpayment, to Deposit Account Number 132726 (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)
- ☐ Applicant certifying micro entity status. See 37 CFR 1.29 — NOTE: Absent a valid certification of Micro Entity Status (see forms PTO/SB/15A and 15B), issue fee payment in the micro entity amount will not be accepted at the risk of application abandonment.
- ☐ Applicant asserting small entity status. See 37 CFR 1.27 — NOTE: If the application was previously under micro entity status, checking this box will be taken to be a notification of loss of entitlement to micro entity status.
- ☐ Applicant changing to regular undiscounted fee status. — NOTE: Checking this box will be taken to be a notification of loss of entitlement to small or micro entity status, as applicable.

NOTE: This form must be signed in accordance with 37 CFR 1.31 and 1.33. See 37 CFR 1.4 for signature requirements and certifications.

Authorized Signature /Andrew T. Pouzeshi/ Date July 29, 2014

Typed or printed name Andrew T. Pouzeshi Registration No. 67672

PTOL-85 Part B (10-13) Approved for use through 10/31/2013. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Trend Micro, Inc.
EX1002-67

PTO/SB/47 (03-09)
Approved for use through 03/31/2012. OMB 0651-0016
U.S. Patent and Trademark Office; U. S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

"FEE ADDRESS" INDICATION FORM

Address to:
Mail Stop M Correspondence
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

- OR -

Fax to:
571-273-6500

INSTRUCTIONS: The issue fee must have been paid for application(s) listed on this form. In addition, only an address represented by a Customer Number can be established as the fee address for maintenance fee purposes (hereafter, fee address). A fee address should be established when correspondence related to maintenance fees should be mailed to a different address than the correspondence address for the application. **When to check the first box below**: If you have a Customer Number to represent the fee address. **When to check the second box below**: If you have no Customer Number representing the desired fee address, in which case a completed Request for Customer Number (PTO/SB/125) must be attached to this form. For more information on Customer Numbers, see the Manual of Patent Examining Procedure (MPEP) § 403.

For the following listed application(s), please recognize as the "Fee Address" under the provisions of 37 CFR 1.363 the address associated with:

☑ Customer Number: 23552

OR

☐ The attached Request for Customer Number (PTO/SB/125) form.

PATENT NUMBER (if known)	APPLICATION NUMBER
	13/460,655

Completed by (check one):

☐ Applicant/Inventor — /Andrew T. Pouzeshi/ (Signature)

☑ Attorney or Agent of record 67672 (Reg. No.) — Andrew T. Pouzeshi (Typed or printed name)

☐ Assignee of record of the entire interest. See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96) — 303-357-1639 (Requester's telephone number)

☐ Assignee recorded at Reel _____ Frame _____ — July 29, 2014 (Date)

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more that one signature is required, see below*.

☐ * Total of _____ forms are submitted.

This collection of information is required by 37 CFR 1.363. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 5 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND COMPLETED FORMS TO THIS ADDRESS.
SEND TO: Mail Stop M Correspondence, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.
If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Jerome L. Schneider	Examiner:	Lane, Gregory A.
Serial No.:	13/460,655	Group Art Unit:	2438
Filed:	April 30, 2012	Docket No.:	04584.0011USC1
Title:	MALWARE MANAGEMENT THROUGH KERNEL DETECTION DURING A BOOT SEQUENCE		

COMMENTS ON STATEMENT OF REASONS FOR ALLOWANCE

Mail Stop Issue Fee
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

Attorney for Applicants acknowledges the allowance of claims 1, 2, 4-8, 10-14, and 16-23 by the Examiner with appreciation. Attorney for Applicants agrees with the Examiner's Statement of Reasons for Allowance to the extent that the claims of the present invention are patentable over the references in the record. Attorney for Applicants expressly traverses the Examiner's Statement of Reasons for Allowance to the extent that any comment is intended or has the effect of limiting a claim scope, explicitly or implicitly, by not reciting verbatim the respective claim language, or is intended or has the effect of limiting a claim scope by stating or implying that all the reasons for patentability are in any way fully enumerated.

Attorney for Applicants further points out that the reasons for allowance set forth by the Examiner are not the only reasons that claims 1, 2, 4-8, 10-14, and 16-23 are allowable. Further reasons for allowance of the claims beyond those enumerated by the Examiner are described and set forth in the specification. In addition, structures that perform substantially the same function in substantially the same way to achieve the same results are included within the scope of the claims.

Finally, as the Examiner's reasons for allowance are not exhaustive, such reasons for allowance do not establish an estoppel against Attorney for Applicants seeking and obtaining allowance of additional, broader claims in a continuation application, which Attorney for Applicants reserves the right to file.

It is believed that no further fees are due with this Communication. However, the Commissioner is hereby authorized to charge any deficiencies or credit any overpayment with respect to this patent application to deposit account number 13-2725.

Respectfully submitted,

Dated: July 29, 2014

/Andrew T. Pouzeshi/

Andrew T. Pouzeshi, Reg. No. 67,672
Merchant & Gould P.C.
P.O. Box 2903
Minneapolis, MN 55402-0903
(303) 357-1639

23552
PATENT TRADEMARK OFFICE

Electronic Patent Application Fee Transmittal

Application Number:	13460655
Filing Date:	30-Apr-2012
Title of Invention:	Malware Management Through Kernel Detection During a Boot Sequence
First Named Inventor/Applicant Name:	Jerome L. Schneider
Filer:	Andrew T. Pouzeshi/Jacquelyn Slate
Attorney Docket Number:	6890-45-CON

Filed as Small Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Utility Appl Issue Fee	2501	1	480	480
Extension-of-Time:				

Trend Micro, Inc.
EX1002-72

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**480**

Electronic Acknowledgement Receipt

EFS ID:	19713674
Application Number:	13460655
International Application Number:	
Confirmation Number:	9502
Title of Invention:	Malware Management Through Kernel Detection During a Boot Sequence
First Named Inventor/Applicant Name:	Jerome L. Schneider
Customer Number:	23552
Filer:	Andrew T. Pouzeshi/Jacquelyn Slate
Filer Authorized By:	Andrew T. Pouzeshi
Attorney Docket Number:	6890-45-CON
Receipt Date:	29-JUL-2014
Filing Date:	30-APR-2012
Time Stamp:	14:12:34
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$480
RAM confirmation Number	384
Deposit Account	132725
Authorized User	POUZESHI, ANDREW

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.20 (Post Issuance fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Issue Fee Payment (PTO-85B)	2014_07_29_IssueFeePartB_04584_0011USC1.PDF	1099637 86e9684f97f2facb487842f3df438b9b7c42737d	no	3
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
2	Post Allowance Communication - Incoming	2014_07_29_CommentsForAllow_04584_0011USC1.pdf	69803 e661c41fe101fae5af023fcd27a846a776aa3f59	no	2
Warnings:					
Information:					
3	Fee Worksheet (SB06)	fee-info.pdf	30481 c100fda8ad4ce11cec5957e2ab8acf8846c31e06	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	1199921		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
13/460,655	04/30/2012	Jerome L. Schneider	6890-45-CON

CONFIRMATION NO. 9502

23552
MERCHANT & GOULD PC
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903

POA ACCEPTANCE LETTER

OC00000069327562

Date Mailed: 07/09/2014

NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 06/26/2014.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

/ktesfaye/

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

Trend Micro, Inc.
EX1002-76

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
13/460,655	04/30/2012	Jerome L. Schneider	6890-45-CON

CONFIRMATION NO. 9502

POWER OF ATTORNEY NOTICE

22442
Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

OC00000069327558

Date Mailed: 07/09/2014

NOTICE REGARDING CHANGE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 06/26/2014.

- The Power of Attorney to you in this application has been revoked by the assignee who has intervened as provided by 37 CFR 3.71. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

/ktesfaye/

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

Trend Micro, Inc.
EX1002-77

Doc Code: PA..
Document Description: Power of Attorney

PTO/AIA/82A (07-13)
Approved for use through 11/30/2014. OMB 0651-0051
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FOR POWER OF ATTORNEY TO ONE OR MORE REGISTERED PRACTITIONERS

NOTE: This form is to be submitted with the Power of Attorney by Applicant form (PTO/AIA/82B) to identify the application to which the Power of Attorney is directed, in accordance with 37 CFR 1.5, unless the application number and filing date are identified in the Power of Attorney by Applicant form. If neither form PTO/AIA/82A nor form PTO/AIA82B identifies the application to which the Power of Attorney is directed, the Power of Attorney will not be recognized in the application.

Application Number	13/460,655
Filing Date	April 30, 2012
First Named Inventor	Jerome L. Schneider
Title	Malware Management Through Kernel Detection During a Boot Sequence
Art Unit	2438
Examiner Name	LANE, GREGORY A.
Attorney Docket Number	04584.0011USC1

SIGNATURE of Applicant or Patent Practitioner

Signature	/Andrew T. Pouzeshi/	Date (Optional)	June 26, 2014
Name	Andrew T. Pouzeshi	Registration Number	67672
Title (if Applicant is a juristic entity)			
Applicant Name (if Applicant is a juristic entity)			

NOTE: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4(d) for signature requirements and certifications. If more than one applicant, use multiple forms.

☐ *Total of _____ forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.32, and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Doc Code: PA..
Document Description: Power of Attorney

PTO/AIA/82B (07-13)
Approved for use through 11/30/2014. OMB 0651-0051
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

POWER OF ATTORNEY BY APPLICANT

I hereby revoke all previous powers of attorney given in the application identified in either the attached transmittal letter or the boxes below.

Application Number	Filing Date

(Note: The boxes above may be left blank if information is provided on form PTO/AIA/82A.)

☐ I hereby appoint the Patent Practitioner(s) associated with the following Customer Number as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above: 23552

OR

☐ I hereby appoint Practitioner(s) named in the attached list (form PTO/AIA/82C) as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the patent application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above. (Note: Complete form PTO/AIA/82C.)

Please recognize or change the correspondence address for the application identified in the attached transmittal letter or the boxes above to:

☑ The address associated with the above-mentioned Customer Number

OR

☐ The address associated with Customer Number:

OR

Firm or Individual Name					
Address					
City		State		Zip	
Country					
Telephone		Email			

I am the Applicant (if the Applicant is a juristic entity, list the Applicant name in the box):

WEBROOT INC.

☐ Inventor or Joint Inventor (title not required below)

☐ Legal Representative of a Deceased or Legally Incapacitated Inventor (title not required below)

☑ Assignee or Person to Whom the Inventor is Under an Obligation to Assign (provide signer's title if applicant is a juristic entity)

☐ Person Who Otherwise Shows Sufficient Proprietary Interest (e.g., a petition under 37 CFR 1.46(b)(2) was granted in the application or is concurrently being filed with this document) (provide signer's title if applicant is a juristic entity)

SIGNATURE of Applicant for Patent

The undersigned (whose title is supplied below) is authorized to act on behalf of the applicant (e.g., where the applicant is a juristic entity).

Signature		Date (Optional)	May 12, 2014
Name	Ryan Howe		
Title	Associate General Counsel		

NOTE: Signature - This form must be signed by the applicant in accordance with 37 CFR 1.33. See 37 CFR 1.4 for signature requirements and certifications. If more than one applicant, use multiple forms.

☐ Total of forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.32, and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/AIA/96 (08-12)
Approved for use through 01/31/2013. OMB 0651-0031
U.S. Patent and Trademark Office;U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of1995, no persons arerequired to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(c)

Applicant/Patent Owner: Webroot Inc.

Application No./Patent No.: 13/460,655 Filed/Issue Date: April 30, 2012

Titled: Malware Management Through Kernel Detection During a Boot Sequence

Webroot Inc., a corporation

(Name of Assignee) (Type of Assignee, e.g., corporation, partnership, university, government agency, etc.)

states that, for the patent application/patent identified above, it is (choose **one** of options 1, 2, 3 or 4 below):

1. ☑ The assignee of the entire right, title, and interest.

2. ☐ An assignee of less than the entire right, title, and interest (check applicable box):

   ☐ The extent (by percentage) of its ownership interest is _____%. Additional Statement(s) by the owners holding the balance of the interest must be submitted to account for 100% of the ownership interest.

   ☐ There are unspecified percentages of ownership. The other parties, including inventors, who together own the entire right, title and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

3. ☐ The assignee of an undivided interest in the entirety (a complete assignment from one of the joint inventors was made). The other parties, including inventors, who together own the entire right, title, and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

4. ☐ The recipient, via a court proceeding or the like (*e.g.*, bankruptcy, probate), of an undivided interest in the entirety (a complete transfer of ownership interest was made). The certified document(s) showing the transfer is attached.

The interest identified in option 1, 2 or 3 above (not option 4) is evidenced by either (choose **one** of options A or B below):

A. ☐ An assignment from the inventor(s) of the patent application/patent identified above. The assignment was recorded in the United States Patent and Trademark Office at Reel _____, Frame _____, or for which a copy thereof is attached.

B. ☑ A chain of title from the inventor(s), of the patent application/patent identified above, to the current assignee as follows:

1. From: Jerome L. Schneider To: Webroot Software, Inc.

   The document was recorded in the United States Patent and Trademark Office at
   Reel 028201, Frame 0093, or for which a copy thereof is attached.

2. From: Webroot Software, Inc. To: Webroot Inc.

   The document was recorded in the United States Patent and Trademark Office at
   Reel 029066, Frame 0457, or for which a copy thereof is attached.

This collection of information is required by37 CFR3.73(b). The information is required toobtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentialityis governed by35 U.S.C. 122and 37 CFR1.11 and1.14. Thiscollection is estimated to take 12 minutes to complete, including gathering, preparing, and submittingthe completed application form to the USPTO.Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent tothe Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS.**SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/AIA/96 (08-12)
Approved for use through 01/31/2013. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(c)

3. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

4. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

5. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

6. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

☐ Additional documents in the chain of title are listed on a supplemental sheet(s).

☑ As required by 37 CFR 3.73(c)(1)(i), the documentary evidence of the chain of title from the original owner to the assignee was, or concurrently is being, submitted for recordation pursuant to 37 CFR 3.11.

[NOTE: A separate copy (i.e., a true copy of the original assignment document(s)) must be submitted to Assignment Division in accordance with 37 CFR Part 3, to record the assignment in the records of the USPTO. See MPEP 302.08]

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignee.

/Andrew T. Pouzeshi/	June 26, 2014
Signature	Date
Andrew T. Pouzeshi	67672
Printed or Typed Name	Title or Registration Number

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that yoube given certain informationin connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, pleasebe advised that: (1) the general authority forthe collection of thisinformation is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and(3) the principal purpose forwhich the information isused by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent applicationor patent. If you do not furnish the requested information,the U.S. Patent and Trademark Office may not be able to process and/or examineyour submission,which may result in termination of proceedings or abandonment of the applicationor expiration of the patent.

The informationprovided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the informationin order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an InternationalApplication filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, arecord may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from thissystem of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Acknowledgement Receipt

EFS ID:	19424672
Application Number:	13460655
International Application Number:	
Confirmation Number:	9502
Title of Invention:	Malware Management Through Kernel Detection During a Boot Sequence
First Named Inventor/Applicant Name:	Jerome L. Schneider
Customer Number:	22442
Filer:	Andrew T. Pouzeshi/Jacquelyn Slate
Filer Authorized By:	Andrew T. Pouzeshi
Attorney Docket Number:	6890-45-CON
Receipt Date:	26-JUN-2014
Filing Date:	30-APR-2012
Time Stamp:	16:05:35
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Power of Attorney	2014_06_26_POA_Executed_04584_0011USC1.PDF	271540 2f6e5c33baaa677b6a4d16c8e2eff70cb91944cd	no	2

Warnings:

The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing

Information:

2	Assignee showing of ownership per 37 CFR 3.73.	2014_06_26_373c_04584_0011USC1.PDF	119840 643642777551552719fc34da6a4e646574bb37191	no	3

Warnings:

Information:

Total Files Size (in bytes):	391380

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

22442 7590 04/29/2014

Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

EXAMINER
LANE, GREGORY A

ART UNIT	PAPER NUMBER
2438	

DATE MAILED: 04/29/2014

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/460,655	04/30/2012	Jerome L. Schneider	6890-45-CON	9502

TITLE OF INVENTION: Malware Management Through Kernel Detection During a Boot Sequence

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	SMALL	$480	$0	$0	$480	07/29/2014

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the ENTITY STATUS shown above. If the ENTITY STATUS is shown as SMALL or MICRO, verify whether entitlement to that entity status still applies.

If the ENTITY STATUS is the same as shown above, pay the TOTAL FEE(S) DUE shown above.

If the ENTITY STATUS is changed from that shown above, on PART B - FEE(S) TRANSMITTAL, complete section number 5 titled "Change in Entity Status (from status indicated above)".

For purposes of this notice, small entity fees are 1/2 the amount of undiscounted fees, and micro entity fees are 1/2 the amount of small entity fees.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

Trend Micro, Inc.
EX1002-85

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

22442 7590 04/29/2014
Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission
I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/460,655	04/30/2012	Jerome L. Schneider	6890-45-CON	9502

TITLE OF INVENTION: Malware Management Through Kernel Detection During a Boot Sequence

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	SMALL	$480	$0	$0	$480	07/29/2014

EXAMINER	ART UNIT	CLASS-SUBCLASS
LANE, GREGORY A	2438	713-002000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
- ❑ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- ❑ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list
(1) The names of up to 3 registered patent attorneys or agents OR, alternatively,
(2) The name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
1 _____
2 _____
3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ❑ Individual ❑ Corporation or other private group entity ❑ Government

4a. The following fee(s) are submitted:
- ❑ Issue Fee
- ❑ Publication Fee (No small entity discount permitted)
- ❑ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**
- ❑ A check is enclosed.
- ❑ Payment by credit card. Form PTO-2038 is attached.
- ❑ The Director is hereby authorized to charge the required fee(s), any deficiency, or credits any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)
- ❑ Applicant certifying micro entity status. See 37 CFR 1.29
  NOTE: Absent a valid certification of Micro Entity Status (see forms PTO/SB/15A and 15B), issue fee payment in the micro entity amount will not be accepted at the risk of application abandonment.
- ❑ Applicant asserting small entity status. See 37 CFR 1.27
  NOTE: If the application was previously under micro entity status, checking this box will be taken to be a notification of loss of entitlement to micro entity status.
- ❑ Applicant changing to regular undiscounted fee status.
  NOTE: Checking this box will be taken to be a notification of loss of entitlement to small or micro entity status, as applicable.

NOTE: This form must be signed in accordance with 37 CFR 1.31 and 1.33. See 37 CFR 1.4 for signature requirements and certifications.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

Trend Micro, Inc.
EX1002-86

PTOL-85 Part B (10-13) Approved for use through 10/31/2013. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/460,655	04/30/2012	Jerome L. Schneider	6890-45-CON	9502

22442 7590 04/29/2014

Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

EXAMINER	
LANE, GREGORY A	

ART UNIT	PAPER NUMBER
2438	

DATE MAILED: 04/29/2014

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(Applications filed on or after May 29, 2000)

The Office has discontinued providing a Patent Term Adjustment (PTA) calculation with the Notice of Allowance.

Section 1(h)(2) of the AIA Technical Corrections Act amended 35 U.S.C. 154(b)(3)(B)(i) to eliminate the requirement that the Office provide a patent term adjustment determination with the notice of allowance. See Revisions to Patent Term Adjustment, 78 Fed. Reg. 19416, 19417 (Apr. 1, 2013). Therefore, the Office is no longer providing an initial patent term adjustment determination with the notice of allowance. The Office will continue to provide a patent term adjustment determination with the Issue Notification Letter that is mailed to applicant approximately three weeks prior to the issue date of the patent, and will include the patent term adjustment on the patent. Any request for reconsideration of the patent term adjustment determination (or reinstatement of patent term adjustment) should follow the process outlined in 37 CFR 1.705.

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

Trend Micro, Inc.
EX1002-87

OMB Clearance and PRA Burden Statement for PTOL-85 Part B

The Paperwork Reduction Act (PRA) of 1995 requires Federal agencies to obtain Office of Management and Budget approval before requesting most types of information from the public. When OMB approves an agency request to collect information from the public, OMB (i) provides a valid OMB Control Number and expiration date for the agency to display on the instrument that will be used to collect the information and (ii) requires the agency to inform the public about the OMB Control Number's legal significance in accordance with 5 CFR 1320.5(b).

The information collected by PTOL-85 Part B is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450. Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Notice of Allowability	Application No. 13/460,655	Applicant(s) SCHNEIDER, JEROME L.	
	Examiner GREGORY LANE	Art Unit 2438	AIA (First Inventor to File) Status No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☐ This communication is responsive to *1/22/2014*.
   ☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on_____.
2. ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.
3. ☒ The allowed claim(s) is/are *1,2,4-8,10-14 and 16-23*. As a result of the allowed claim(s), you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov .
4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

   Certified copies:

   a) ☐ All b) ☐ Some *c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).
6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☒ Notice of References Cited (PTO-892)
2. ☐ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
3. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
4. ☒ Interview Summary (PTO-413), Paper No./Mail Date *4/14/2014* .
5. ☒ Examiner's Amendment/Comment
6. ☒ Examiner's Statement of Reasons for Allowance
7. ☐ Other _____.

/GREGORY LANE/
Examiner, Art Unit 2438

Trend Micro, Inc.
EX1002-89

The present application is being examined under the pre-AIA first to invent provisions.

DETAILED ACTION

1. The text of those sections of Title 35 U.S. Code not included in this section can be found in the prior office action.
2. The prior office actions are incorporated herein by reference. In particular, the observations with respect to claim language, and response to previously presented arguments.
3. Claims 1, 4, 7, 8, 10, 13, and 16 are amended
4. Claims 3, 9, and 15 are cancelled
5. Claims 21-23 are new
6. Claims 1, 2, 4-8, 10-14, and 16-23, now re-numbered 1-20, are pending.

EXAMINER'S AMENDMENT

7. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Brent Lindon on April 14, 2014.

8. **CLAIMS:**

Please amend claims as follows:

1. (Currently Amended) A method for managing pestware on a computer comprising:

monitoring events during a boot sequence of the computer;

managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a subsystem of an operating system is loaded, and after a kernel is loaded;

managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;

generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;

analyzing the record of events so as to identify the pestware-related events;

modifying the set of behavior rules so as to prevent the pestware related events; and

scanning data in a registry of the computer for pestware during the second period in the boot sequence.

3. (Canceled)

4. (Currently Amended) The method of ~~claim 3~~claim 1, wherein the record of events includes information selected from ~~the~~ a group consisting of: process identification information, file identification information, and hook generation information.

7. (Currently Amended) A system for managing pestware on a computer comprising:

a processor;

a memory comprising:

a first instruction set configured to monitor events during a boot sequence of the computer;

a second instruction set configured to manage pestware-related events during first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a WIN32 subsystem is loaded, and after a kernel is loaded;

a third instruction set configured to manage pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;

a fourth instruction set configured to generate, in response to the monitoring, a record of events, the record of events including the pestware-related events;

a fifth instruction set configured to analyze the record of events so as to identify the pestware-related events; and

a sixth instruction set configured to modify the set of behavior rules so as to prevent the pestware related events; and

a ~~fourth~~ seventh instruction set configured to scan data in a registry of the computer for pestware during the second period in the boot sequence.

8. (Currently Amended) The system of claim 7, wherein the memory further comprises ~~a fifth~~an eighth instruction set configured to launch, after an operating system is initiated, a pestware management engine; and wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

9. (Canceled)

10. (Currently Amended) The system of ~~claim 9~~claim 1, wherein the record of events includes information selected from ~~the~~ a group consisting of: process identification information, file identification information, and hook generation information.

13. (Currently Amended) A non-transitory tangible computer-readable storage medium comprising program instructions for:

monitoring events during a boot sequence of a computer;

managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a WIN32 subsystem is loaded, and after a kernel is loaded;

managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;

generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;

analyzing the record of events so as to identify the pestware-related events;

modifying the set of behavior rules so as to prevent the pestware related events; and

scanning data in a registry of the computer for pestware during the second period in the boot sequence.

15. (Canceled)

16. (Currently Amended) The non-transitory tangible computer-readable storage medium of ~~claim 15~~claim 13, wherein the record of events includes information selected from ~~the~~ a group consisting of: process identification information, file identification information, and hook generation information.

21. (New) The method of claim 1, wherein the pestware-related invents include events associated with at least one of loading pestware, executing pestware, and one or more behaviors of pestware.

22. (New) The system of claim 7, wherein the pestware-related invents include events associated with at least one of loading pestware, executing pestware, and one or more behaviors of pestware.

23. (New) The non-transitory tangible computer-readable storage medium of claim 13, wherein the pestware-related invents include events associated with at least one of loading pestware, executing pestware, and one or more behaviors of pestware.

Response to Arguments

9. Applicant's arguments filed on January 22, 2014 have been fully considered and they are persuasive.

Allowable Subject matter

10. Claims 1, 2, 4-8, 10-14, and 16-23 are allowed over prior art of record.

Examiner's Statement of Reasons for Allowance

11. The following is an examiner's statement of reasons for allowance:

Newly amended independent claims 1, 7, and 13 are allowed in view of the examiner's amendment and for reasons argued by applicant in pages 8-11 of the Applicant's remarks, filed January 22, 2014 and for filing a Terminal Disclaimer on April 17, 2014. The prior art of record do not teach monitoring pestware related operations during a first period in a boot sequence before native applications are running and after the kernel is loaded and before a subsystem of an operating system is loaded. Managing pestware related events in accordance with a set of behavior rules, pestware related events during a second period in the boot sequence when native applications are running and generating a file containing a list of detected pestware related events. Subsequently,

scanning the registry of the computer for the listed pestware related events during the boot sequence.

None of the prior art of record, either taken by itself or in any combination, would have anticipated or made obvious the invention of the present application at or before the time it was filed. The dependent claims, 2, 4-6, 8, 10-12, 14, and 16-23 are also allowable by virtue of their dependence on the independent claims.

Any comments considered necessary by applicant must be submitted no later than payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled “Comments on Statement of Reasons for Allowance."

CONCLUSION

Any inquiry concerning this communication or earlier communications from the examiner should be directed to GREGORY LANE whose telephone number is (571)270-7469. The examiner can normally be reached on 571 270 7469 from 8:00 AM to 6:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Taghi Arani, can be reached on 571 272 3787. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for

published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/GREGORY LANE/ Examiner, Art Unit 2438

/TAGHI ARANI/
Supervisory Patent Examiner, Art Unit 2438

Examiner-Initiated Interview Summary	Application No.	Applicant(s)	
	13/460,655	SCHNEIDER, JEROME L.	
	Examiner	Art Unit	
	GREGORY LANE	2438	

All participants (applicant, applicant's representative, PTO personnel):

(1) *GREGORY LANE*. (3)_____.

(2) *BRENT LINDON*. (4)_____.

Date of Interview: *14 April 2014*.

Type: ☒ Telephonic ☐ Video Conference
☐ Personal [copy given to: ☐ applicant ☐ applicant's representative]

Exhibit shown or demonstration conducted: ☐ Yes ☐ No.
If Yes, brief description: _____.

Issues Discussed ☐101 ☐112 ☐102 ☐103 ☐Others
(For each of the checked box(es) above, please describe below the issue and detailed description of the discussion)

Claim(s) discussed: *1,3,7,9,13 and 15*.

Identification of prior art discussed: _____.

Substance of Interview
(For each issue discussed, provide a detailed description and indicate if agreement was reached. Some topics may include: identification or clarification of a reference or a portion thereof, claim interpretation, proposed amendments, arguments of any applied references etc...)

Applicant agreed to amend independent claims 1, 7, and 13 with limitations from dependent claims 3, 9, and 15 and to file a terminal disclaimer in order to place application in condition for allowance..

Applicant recordation instructions: It is not necessary for applicant to provide a separate record of the substance of interview.

Examiner recordation instructions: Examiners must summarize the substance of any interview of record. A complete and proper recordation of the substance of an interview should include the items listed in MPEP 713.04 for complete and proper recordation including the identification of the general thrust of each argument or issue discussed, a general indication of any other pertinent matters discussed regarding patentability and the general results or outcome of the interview, to include an indication as to whether or not agreement was reached on the issues raised.

☐ Attachment

/GREGORY LANE/ Examiner, Art Unit 2438	

Trend Micro, Inc.
EX1002-97

Notice of References Cited	Application/Control No. 13/460,655	Applicant(s)/Patent Under Reexamination SCHNEIDER, JEROME L.	
	Examiner GREGORY LANE	Art Unit 2438	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-			
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Wen, "Implicit Detection od Hidden Processes with a Local-Booted Virtual Machinr", 2008, IEEE, pg. 150-155
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13460655	SCHNEIDER, JEROME L.
	Examiner	Art Unit
	GREGORY LANE	2438

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☒ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	10/07/2013	04/17/2014							
1	1	✓	=							
2	2	✓	=							
	3	✓	-							
3	4	✓	=							
4	5	✓	=							
5	6	✓	=							
6	7	✓	=							
7	8	✓	=							
	9	✓	-							
8	10	✓	=							
9	11	✓	=							
10	12	✓	=							
11	13	✓	=							
12	14	✓	=							
	15	✓	-							
13	16	✓	=							
14	17	✓	=							
15	18	✓	=							
16	19		=							
17	20		=							
18	21		=							
19	22		=							
20	23		=							

Trend Micro, Inc.
EX1002-99

Issue Classification	**Application/Control No.** 13460655	**Applicant(s)/Patent Under Reexamination** SCHNEIDER, JEROME L.
	Examiner GREGORY LANE	**Art Unit** 2438

CPC						
Symbol					**Type**	**Version**
G06F		21	/	56	F	20130101
G06F		21	/	566	I	20130101
G06F		21	/	554	I	20130101
			/			
			/			
			/			
			/			
			/			
			/			
			/			
			/			
			/			
			/			
			/			
			/			

CPC Combination Sets								
Symbol					**Type**	**Set**	**Ranking**	**Version**
			/					
			/					

/GREGORY LANE/ Examiner.Art Unit 2438 (Assistant Examiner)	04/17/2014 (Date)	**Total Claims Allowed:** 20
/TAGHI ARANI/ Supervisory Patent Examiner.Art Unit 2438 (Primary Examiner)	04/18/2014 (Date)	O.G. Print Claim(s): 1 / O.G. Print Figure: 1

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13460655	SCHNEIDER, JEROME L.
	Examiner	**Art Unit**
	GREGORY LANE	2438

US ORIGINAL CLASSIFICATION		INTERNATIONAL CLASSIFICATION										
CLASS	SUBCLASS	CLAIMED					NON-CLAIMED					
713	2	G	0	6	F	21 / 00 (2013.01.01)						

CROSS REFERENCE(S)					
CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)				
713	193				

/GREGORY LANE/ Examiner.Art Unit 2438	04/17/2014	**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	20	
/TAGHI ARANI/ Supervisory Patent Examiner.Art Unit 2438	04/18/2014	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13460655	SCHNEIDER, JEROME L.
	Examiner	**Art Unit**
	GREGORY LANE	2438

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☒ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	17	20												
2	2	18	21												
3	4	19	22												
4	5	20	23												
5	6														
6	7														
7	8														
8	10														
9	11														
10	12														
11	13														
12	14														
13	16														
14	17														
15	18														
16	19														

/GREGORY LANE/ Examiner.Art Unit 2438 (Assistant Examiner)	04/17/2014 (Date)	**Total Claims Allowed:** 20	
/TAGHI ARANI/ Supervisory Patent Examiner.Art Unit 2438 (Primary Examiner)	04/18/2014 (Date)	O.G. Print Claim(s) 1	O.G. Print Figure 1

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	105	kernel AND rootkit	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 12:54
S2	3	S1 AND "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 12:55
S3	16	S1 AND @pd < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 12:56
S4	17	kernel AND rootkit SAME signature	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 13:45
S5	68	kernel AND rootkit AND signature	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 13:45
S6	13	S5 AND @pd < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 13:46
S7	36	S5 AND @ad < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 16:04
S8	0	driver SAME kernel SAME (detector	US-PGPUB;	ADJ	ON	2009/02/03

		OR monitor OR inspec$5) AND rootkey	USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			19:09
S9	690	driver SAME kernel SAME (detector OR monitor OR inspec$5)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/03 19:09
S10	523	S9 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/03 19:10
S11	34	(device driver) AND rootkit	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 11:11
S12	8	S11 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 11:11
S13	8	S11 AND @rlad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:10
S14	3	kernel NEAR2 detector SAME driver AND (malicious OR rootkit OR infect$3) NEAR2 driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:32
S15	4	kernel level detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:38
S16	5	kernel detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	ADJ	ON	2009/02/04 12:38

			IBM_TDB			
S17	2	(detector NEAR3 driver) AND kernel AND (malicious OR infect$4) driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:39
S18	3	(detector) AND kernel AND (malicious OR infect$4) driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:40
S19	4	detector AND kernel AND (malicious OR infect$4) (driver OR dll)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:41
S20	35	detector AND kernel AND ((malicious OR infect$4) (driver OR dll) OR rootkit)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:46
S21	20	S20 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:46
S22	145	(kernel level driver) OR (detector NEAR2 driver) AND kernel	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:05
S23	105	S22 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:05
S24	145	(kernel level driver) OR (detector NEAR2 driver) AND kernel AND detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:08
S25	105	S24 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS;	ADJ	ON	2009/02/04 13:08

			EPO; JPO; DERWENT; IBM_TDB			
S26	56	((kernel level driver) OR (detector NEAR2 driver)) AND kernel AND detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:10
S27	50	S26 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:10
S28	2	S26 AND @ad < "20051004" AND (malware OR virus OR rootkit OR worm)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:14
S29	94	kernel level driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:49
S30	0	("20070055711")".pn"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 14:41
S31	2	("20070055711").pn.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 14:41
S32	1	("20070055711").pn.	US-PGPUB; USPAT	ADJ	ON	2009/02/04 16:44
S33	1	("20070055711").pn. AND signature	US-PGPUB; USPAT	ADJ	ON	2009/02/04 17:49
S34	1	("20040049693").pn.	US-PGPUB; USPAT	ADJ	ON	2009/02/05 09:18
S35	1	("7398389").pn.	US-PGPUB; USPAT	ADJ	ON	2009/02/05 09:20
S36	14844	(380/276,277,278,284,286 OR 713/182,183,184,185,176 OR 726/28,30,33 OR 714/22-30).CCLS.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/05 11:30

S37	38	paul NEAR gassoway	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/13 15:28
S38	24	kernel AND (rootkit OR malware OR virus OR worm OR pestware) AND boot SAME (initialize OR start OR bootup) AND kernel WITH monitor	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:02
S39	14	S38 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:02
S40	83	kernel AND (rootkit OR malware OR virus OR worm OR pestware) AND boot$3 SAME (initializ$5 OR start$3 OR bootup) AND kernel WITH monitor$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:09
S41	55	S40 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:09
S42	762	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR "20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:39
S43	58	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:39

		"20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433").PN.				
S44	3	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR "20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433").PN. AND kernel AND boot$3 AND (malware OR virus OR worm OR pestware)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:41
S45	6	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR "20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433" OR "20020129277" OR "20030159070" OR "20040225877" OR "20050120242" OR "20050149726" OR "20050172115" OR "20050188272" OR "20050257266" OR "2006026576").PN. AND kernel AND boot$3 AND (malware OR virus OR worm OR pestware)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:46
S46	42	S40 AND @ad < "20060801" AND @pd < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:47
S47	37986112	S40 AND @ad < "20060801" OR @pd < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	ADJ	ON	2009/08/20 14:47

			IBM_TDB			
S48	55	S40 AND (@ad < "20060801" OR @pd < "20060801")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:47
S49	8349	S40 AND @ad < "20060801" AND (boot$3 WITH before) OR pre-boot$3 OR (prior WITH boot$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:50
S50	14	S40 AND @ad < "20060801" AND ((boot$3 WITH before) OR pre-boot$3 OR (prior WITH boot$3))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:51
S51	180	kernel AND boot AND Win32 AND (scan$4 OR monitor$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:09
S52	158	S51 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:10
S53	94	S51 AND @ad < "20060801" AND registry	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:11
S54	12	kernel AND boot AND Win32 AND (scan$4 OR monitor$3) WITH registry	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:13
S55	8	S54 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:14
S56	1	("20050125687").pn.	US-PGPUB; USPAT	ADJ	ON	2009/08/24 13:57
S57	1	("20050125687").pn. AND log	US-PGPUB;	ADJ	ON	2009/08/24

			USPAT			13:57
S58	18	kernel AND boot AND (change OR modify OR update) WITH (rule$2 OR polic$3) SAME (read WITH log)	US-PGPUB; USPAT	ADJ	ON	2009/08/24 14:28
S59	18	kernel AND boot AND (change OR modify OR update) WITH (rule$2 OR polic$3) SAME (read WITH log)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 14:32
S60	51	kernel AND boot AND (change OR modify OR update) WITH (rule$2 OR polic$3) AND (read WITH log)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 14:38
S61	26	S60 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 14:38
S62	1	("20030135791").pn.	US-PGPUB; USPAT	ADJ	ON	2009/08/25 12:03
S63	71	(713/164 OR 726/2,4,23-25) AND kernel AND (malware OR virus OR worm) AND boot$3	US-PGPUB; USPAT	ADJ	ON	2009/08/25 12:55
S64	7	jerome NEAR schneider	US-PGPUB; USPAT	ADJ	ON	2009/08/25 12:58
S65	331	(system OR device) driver SAME (malware OR pestware OR virus)	US-PGPUB; USPAT	ADJ	ON	2011/04/09 10:44
S66	209	S65 AND @AD < "20060801"	US-PGPUB; USPAT	ADJ	ON	2011/04/09 10:44
S67	85	S65 AND @AD < "20060801" AND kernel	US-PGPUB; USPAT	ADJ	ON	2011/04/09 10:45
S68	41	S65 AND @AD < "20060801" AND kernel AND boot	US-PGPUB; USPAT	ADJ	ON	2011/04/09 11:05
S69	1	native NEAR2 scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT	ADJ	ON	2011/04/09 11:59
S70	1	native NEAR2 scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2011/04/09 12:00
S71	433	scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2011/04/09 12:01
S72	16	boot With scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2011/04/09 12:20
S73	11	S72 AND @AD < "20060801"	US-PGPUB; USPAT	ADJ	ON	2011/04/09 12:20
S76	2	("8190868").PN.	US-PGPUB;	ADJ	ON	2013/10/04

			USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			16:03
S77	236	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus)	US-PGPUB; USPAT	ADJ	ON	2014/04/03 11:50
S78	53	S77 AND @ad < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 11:51
S79	53	S77 AND @ad < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 11:51
S80	9	kernel WITH (scanner OR monitor) WITH (rule OR policy) SAME (malware OR pestware OR antivirus OR anti NEAR virus)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:45
S81	0	S80 AND @ad < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:46
S82	31	kernel WITH (scanner OR monitor) WITH (rule OR policy) AND (malware OR pestware OR antivirus OR anti NEAR virus)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:47
S83	0	S82 AND @ad < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:47
S84	0	S82 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:49
S85	45	kernel WITH (scan$4 OR monitor$4) WITH (rule OR policy) AND (malware OR pestware OR antivirus OR anti	US-PGPUB; USPAT; USOCR;	ADJ	ON	2014/04/03 13:50

		NEAR virus)	FPRS; EPO; JPO; DERWENT; IBM_TDB			
S86	4	S85 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:51
S87	97	kernel WITH (scan$4 OR monitor$4) SAME (rule OR policy) AND (malware OR pestware OR antivirus OR anti NEAR virus)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:54
S88	13	S87 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:54
S89	9	S88 NOT S86	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 13:54
S90	114	kernel WITH (scan$4 OR monitor$4) SAME (rule OR policy) AND (malware OR pestware OR antivirus OR anti NEAR virus OR spyware)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:01
S91	25	S90 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:01
S92	21	S91 NOT S86	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:01
S93	12	S92 NOT S89	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:02
S94	185	kernel WITH (scan$4 OR monitor$4)	US-PGPUB;	ADJ	ON	2014/04/03

		SAME (rule OR policy) AND (malware OR pestware OR antivirus OR anti NEAR virus OR spyware OR malicious)	USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			14:05
S95	56	S94 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:05
S96	47	S95 NOT S89	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:05
S97	43	S96 NOT S86	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2014/04/03 14:06
S98	1	("20030159070").PN.	US-PGPUB; USPAT	ADJ	ON	2014/04/03 16:30
S99	236	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus)	US-PGPUB; USPAT	ADJ	ON	2014/04/17 15:43
S100	28	boot With scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2014/04/17 15:48
S101	1	("20120216027").PN.	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2014/04/17 15:52
S102	223	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus) AND (G06F$8).CPC.	US-PGPUB; USPAT	ADJ	ON	2014/04/17 15:53
S103	45	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus) AND (G06F21/56).CPC.	US-PGPUB; USPAT	ADJ	ON	2014/04/17 15:55
S104	41	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus) AND (G06F21/566).CPC.	US-PGPUB; USPAT	ADJ	ON	2014/04/17 15:55
S105	23	(scan$4 OR monitor$4) WITH registry SAME (pestware OR malware OR antivirus OR anti NEAR virus) AND (G06F21/554).CPC.	US-PGPUB; USPAT	ADJ	ON	2014/04/17 15:55

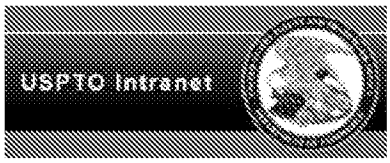
EAST Search History (Interference)

Ref	Hits	Search Query	DBs	Default	Plurals	Time

#				Operator		Stamp
L1	0	(713/2,193) AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:10
L2	0	(G06F21/56).CPC. AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:11
L3	0	(713/2,193).CCLS. AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:12
L4	0	(G06F21/566).CPC. AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:12
L5	0	(G06F21/554).CPC. AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:13
L6	0	(managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/04/17 17:13
S74	0	(713/2,164,193).CCLS. AND ((pestware AND boot AND sequence AND period AND drivers AND initialized AND kernel-level AND monitor AND boot AND managing AND related AND before AND native AND application AND loaded AND subsystem AND scanning AND scanner AND registry AND behavior AND rules) OR (launching AND operating AND system AND	USPAT; UPAD	ADJ	ON	2012/01/14 20:55

		pestware AND management AND behavior AND rules AND portion AND compiled AND engine) OR (generating AND response AND monitoring AND record AND events AND pestware-related AND analyzing AND identify AND behavior AND rules AND prevent)).CLM.				
S75	0	(726/4,22-27).CCLS. AND ((pestware AND boot AND sequence AND period AND drivers AND initialized AND kernel-level AND monitor AND boot AND managing AND related AND before AND native AND application AND loaded AND subsystem AND scanning AND scanner AND registry AND behavior AND rules) OR (launching AND operating AND system AND pestware AND management AND behavior AND rules AND portion AND compiled AND engine) OR (generating AND response AND monitoring AND record AND events AND pestware-related AND analyzing AND identify AND behavior AND rules AND prevent)).CLM.	USPAT; UPAD	ADJ	ON	2012/01/14 20:56

4/ 17/ 2014 5:14:23 PM
C:\ Users\ glane\ Documents\ EAST\ Workspaces\ case_13460655.wsp



PATENTS INTRANET



Scientific and Technical Information Center

Topics | Date | Journals | Authors
Clustered Results
All Topics
Smartphones (8)
Other (12)

Results By Source

- **Computer Source: Consumer Edition (0)**
- **Computers & Applied Sciences Complete (0)**
- **eFunda**
- **Encyclopedia Americana**
- **IEEE Electronic Library Online (132)**
- **Information Science & Technology Abstracts (0)**
- **Internet and Personal Computing Abstracts (0)**

Sort By: Date | Title | Author | Source Filter By: Full Text

Results 1-20 of 20 returned for "full text contains boot AND (malware OR peatware OR anti-virus) AND native and application" (132 total)

1 Next Results for All Resources >

Select all records :: 0 selected records: Email or Export/Save

1. **Analysis of Bayesian classification-based approaches for Android malware detection**
Yerima, S.Y.; Sezer, S.; McWilliams, G.
Information Security, IET (1751-8709)
2014. Vol.8,Iss.1;p.25-36
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

2. **Approach for malware identification using dynamic behaviour and outcome triggering**
Hao Bai; Chang-zhen Hu; Xiao-chuan Jing; Ning Li; Xiao-yin Wang.
Information Security, IET (1751-8709)
2014. Vol.8,Iss.2;p.140-151
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

3. **Design and Implementation of Efficient Integrity Protection for Open Mobile**

Trend Micro, Inc.
EX1002-116

Platforms
Xinwen Zhang; Seifert, J.-P.; Aciicmez, O.
Mobile Computing, IEEE Transactions on (1536-1233)
2014. Vol.13,Iss.1;p.188-201
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

☐ 4.Growing Grapes in Your Computer to Defend Against Malware
Zhiyong Shan; Xin Wang.
Information Forensics and Security, IEEE Transactions on (1556-6013)
2014. Vol.9,Iss.2;p.196-207
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

☐ 5.Evolution, Detection and Analysis of Malware for Smart Devices
Suarez-Tangil, G.; Tapiador, J.; Peris-Lopez, P.; Ribagorda, A.
Communications Surveys & Tutorials, IEEE (1553-877X)
2013. Iss.99;p.1-27
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 6.Investigation of Static and Dynamic Android Anti-virus Strategies
Yan Ma; Sharbaf, M.S.
Information Technology: New Generations (ITNG), 2013 Tenth International Conference on (978-0-7695-4967-5)
2013. p.398-403
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 7.A New Android Malware Detection Approach Using Bayesian Classification
Yerima, S.Y.; Sezer, S.; McWilliams, G.; Muttik, I.
Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on (1550-445X) (978-1-4673-5550-6)
2013. p.121-128
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 8.A scalable approach for malware detection through bounded feature

space behavior modeling
Chandramohan, M.; Hee Beng Kuan Tan; Briand, L.C.; Lwin Khin Shar; Padmanabhuni, B.M.
Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on
2013. p.312-322
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 9.A Survey on Security for Mobile Devices
La Polla, M.; Martinelli, F.; Sgandurra, D.
Communications Surveys & Tutorials, IEEE (1553-877X)
2013. Vol.15,Iss.1;p.446-471
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters
STIC Full Text Retrieval Options

☐ 10.Analysis of malicious and benign android applications
Alazab, M.; Moonsamy, V.; Batten, L.; Lantz, P.; Ronghua Tian.
Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on (1545-0678) (978-1-4673-1423-7)
2012. p.608-616
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 11.Dissecting Android Malware: Characterization and Evolution
Yajin Zhou; Xuxian Jiang.
Security and Privacy (SP), 2012 IEEE Symposium on (1081-6011) (978-1-4673-1244-8)
2012. p.95-109
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 12.DroidLogger: Reveal suspicious behavior of Android applications via instrumentation
Shuaifu Dai; Tao Wei; Wei Zou.
Computing and Convergence Technology (ICCCT), 2012 7th International Conference on (978-1-4673-0894-6)
2012. p.550-555
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 13.Malware analysis using reverse engineering and data mining tools
Burji, S.; Liszka, K.J.; Chan, C.
System Science and Engineering (ICSSE), 2010 International Conference on (978-1-4244-6472-2)
2010. p.619-624
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 14.Analysis of Virtual Machine Applied to Malware Detection System
Zhang Xiaosong; Pan Xiaohui; Long Xiaoshu.
Information Engineering and Electronic Commerce, 2009. IEEC '09. International Symposium on (978-0-7695-3686-6)
2009. p.290-294
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 15.Study of fuzzy clustering methods for malicious codes using native API call frequency
Ochul Kwon; Seongjae Bae; Jaeik Cho; Jongsub Moon.
Computational Intelligence in Cyber Security, 2009. CICS '09. IEEE Symposium on (978-1-4244-2769-7)
2009. p.24-29
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 16.Implicit Detection of Hidden Processes with a Local-Booted Virtual Machine
Yan Wen; Jinjing Zhao; Huaimin Wang.
Information Security and Assurance, 2008. ISA 2008. International Conference on (978-0-7695-3126-7)
2008. p.150-155
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 17.Malware Behavior Analysis in Isolated Miniature Network for Revealing Malware's Network Activity
Inoue, D.; Yoshioka, K.; Eto, M.; Hoshizawa, Y.; Nakao, K.
Communications, 2008. ICC '08. IEEE International Conference on (978-1-4244-2075-9)
2008. p.1715-1721
Source: IEEE Electronic Library Online

Trend Micro, Inc.
EX1002-119

Show Abstract | Show In Clusters

☐ 18.Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware
Xu Chen; Andersen, J.; Mao, Z.M.; Bailey, M.; Nazario, Jose.
Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on (978-1-4244-2397-2)
2008. p.177-186
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 19.OmniUnpack: Fast, Generic, and Safe Unpacking of Malware
Martignoni, L.; Christodorescu, M.; Jha, S.
Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual (1063-9527) (978-0-7695-3060-4)
2007. p.431-441
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

☐ 20.Back to the Future: A Framework for Automatic Malware Removal and System Repair
Hsu, F.; Hao Chen; Ristenpart, T.; Li, J.; Zhendong Su.
Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual (1063-9527) (0-7695-2716-7)
2006. p.257-268
Source: IEEE Electronic Library Online
Show Abstract | Show In Clusters

*Missing check box indicates resource temporarily unavailable.

*Please obey **MPEP Section 904.02 (c) - Internet Searching [R3]** and USPTO "Rules of the Road (**PDF Doc**)" when using Internet resources. If you cannot access a file because of a missing or non-working plugin, please contact the Help Desk at 2-6000 for installation assistance.*

Trend Micro, Inc.
EX1002-120

2008 International Conference on Information Security and Assurance

Implicit Detection of Hidden Processes with a Local-Booted Virtual Machine

Yan Wen[1], Jinjing Zhao[2], Huaimin Wang[1]
wenyan@nudt.edu.cn, misszhaojinjing@sina.com.cn, whm_w@163.com
[1] National University of Defense Technology Changsha, China
[2] Beijing Institute of System Engineering, Beijing, China

Abstract

Currently stealth malware is becoming a major threat to the PC computers. Process hiding is the technique commonly used by stealth malware to evade detection by anti-malware scanners. On the defensive side, previous host-based approaches will be defeated once the privileged stealth malware controls a lower reach of the system. The virtual machine (VM) based solutions gain tamper resistance at the cost of losing the OS-level process view. Moreover, existing VM-based approaches cannot introspect the preinstalled OS which is just the protecting concern for PC users. In this paper, we present a new VM-based approach called Libra which accurately reproduces the software environment of the underlying preinstalled OS within the Libra VM and provides an OS-level semantic view of the processes. With our new local-booting technology, Libra VM just boots from the underlying host OS but not a newly installed OS image. Thus, Libra provides a way to detect the existing process-hiding stealth malware in the host OS. In addition, instead of depending on the guest information which is subvertable to the privileged guest malware, Libra adopts a unique technique to implicitly construct the Trusted View of Process List (TVPL) from within the virtualized hardware layer. Our evaluation results with real-world hiding-process rootkits, which are widely used by stealth malware, demonstrate its practicality and effectiveness.

Key words: Hidden process, virtual machine, local-booted virtual machine.

1. Introduction

On PC platforms, with more and more users are willing to download and execute freeware/shareware, potential stealth malware accompanied with the downloaded untrusted code has become a major threat to PC users [1]. The term "stealth malware" refers to the software programs that start up with the sacrificed OS and try to hide their presence from the process enumeration utilities commonly used by malware detectors [2]. *Process hiding* is the most widely used stealth technique. According to statistics released by Microsoft's widely deployed *Malicious Software Removal Tool*, a significant fraction of the malware it encounters and removes consists of stealth components with the capability of process hiding [3]. Consequently, the ability to detect and respond to malicious hidden processes is a clear advantage in the race to prevent the computers against stealth malware.

The most feasible mechanism to detect hidden processes is so-called *cross-view* validation [2]. It operates by observing the process list from two perspectives, *untrusted view* and *trusted view*, and locating the inconsistencies between these two views. Untrusted view is often retrieved from an untrusted, high-level point. Trusted view is obtained from within a lower layer in the system that is unlikely to have been tampered by malware. If a process exists in the trusted view but does not appear in the untrusted view, a cross-view detector can conclude that it has been hidden.

In the defense point of view, cross-view validation will fail in retrieving the "true" process list if an attacker subverts the level from which the trusted view is obtained. So, the key challenge of cross-view validation is the inevitable race that develops between attackers and defenders to control the lowest reaches of a system. Referring to the layer wherein the trusted view of processes is gained, existing cross-view validation approaches can be categorized into two classes: *host-based* (so-called *"in-the-box"*) [3] and *VM-based* (so-called "*out-of-the-box*") [4-7].

The host-based technologies provide the capability of introspecting the host OS with a native, semantic-rich view, whereas they in the meantime make themselves visible and left many chances to privileged malware to dive deeper than the detectors [8-10].

Compared to host-based approaches, VM-based approaches significantly improve the tamper-resistance of detection facilities in virtue of their location in an isolated virtualization layer. However, existing VM-based approaches either suffer from the semantic gaps

978-0-7695-3126-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ISA.2008.22

IEEE computer society

Trend Micro, Inc.
EX1002-125

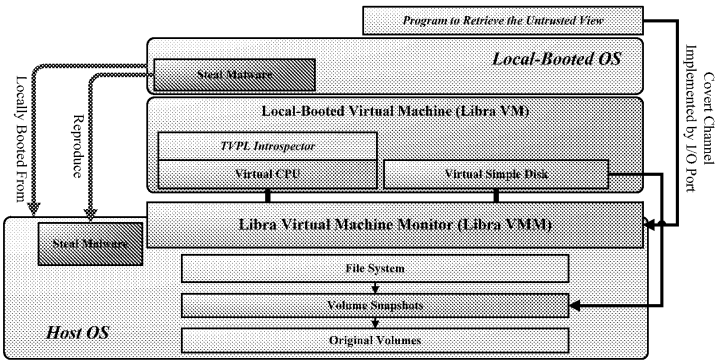


Figure 1. The Architecture of Libra.

between the view of the VM and the view from the inside of the OS [4, 6], or depend on the guest kernel data structures which is still subvertable to the privileged stealth malware [5, 7]. Moreover, the OSes within the VMs don't reproduce the environment of the underlying preinstalled host OSes, which are just our protecting concern on PC platforms. In other words, they only deal with the OS deployed in the VM instead of our daily used host OS.

To address these limitations, we propose a new VM-based cross-view validation approach called Libra for detecting process-hiding malware. Compared to previous VM-based approaches, Libra provides two unique advantages: *host OS environment reproduction*, and *implicit introspection of TVPL*.

Host OS Environment Reproduction. To detect the potential stealth malware existing in the preinstalled host OS at the virtualization layer, Libra introduces a novel *Local-Booted System Virtual Machine* (*Libra VM*). Unlike existing VM-based approaches, Libra VM boots just from the underlying preinstalled host OS instead of booting from a newly installed OS image, viz. Libra VM loads another instance of the host OS. As a consequence, the software environment of host OS, including the possibly existing stealth malware, is faithfully reproduced within this local-booted OS.

Implicit Introspection of TVPL. To decouple from the information subvertable to the privileged guest malware, Libra adopts a novel technique that enable Libra to implicitly discover the "true" process list from within the virtualized hardware layer. By monitoring low-level interactions between OS and the processor memory management unit (MMU), Libra proposes a unique process awareness technology which is capable of accurately determining when an OS creates processes, destroys them, or context-switches between them. With this technology, Libra can detect more process-hiding malware than existing stealth malware detectors.

Libra has been firstly implemented in Windows with Intel x86 processors. Our experimental results with real-world rootkits which are widely used to hide processes demonstrate Libra's unique detection capability.

This paper will discuss the architecture of Libra and focus on the implementation of implicit introspection of TVPL. The rest of the paper is organized as follows. Section 2 describes the design of Libra, followed by the implementation details in Section 3. Section 4 provides the evaluation results of the functionality of hidden process detection. Section 5 reviews previous related works. We summarize the main features of Libra in the last section.

2. Libra Overview

As stated in the previous section, Libra achieves two characteristics: *implicit introspection of TVPL* and *host OS environment reproduction*.

There are two classes of virtual machine monitors (VMM): Type I and Type II [11]. Each of them can serve as the virtualization layer wherein the guest OS is introspected. According to the definition of Goldberg [11], a Type I VMM just runs above a bare computer hardware platform. It tends to be implemented as a lightweight OS with the virtualization capabilities. A Type II VMM is executed as an application. The OS that manages the real computer hardware is called the "host OS". Every OS that runs in the Type II virtual machine is called a "guest OS". In a Type II VMM, the host OS provides process allocation and a standard execution environment to each guest OS.

Trend Micro, Inc.
EX1002-126

Unlike the mainframes that are configured and managed by experienced administrators, desktop PC's are often preinstalled with a standard OS and managed by the end-user. Ignoring the difficulty of proposing a practical and seamless migration approach for PCs, it will maybe take several years to migrate all of them to the Type I VMM. It also might be unacceptable for a PC user to completely replace an existing OS with a Type I VMM. In contrast, Type II VMM allows co-existing with the preinstalled host OS and programs.

Accordingly, taking into account that the preinstalled OS of PC platforms is the prime protecting concern for Libra, we select Type II VMM over Type I VMM to act as the introspecting layer. As illustrated in Figure 1, Libra is composed of three key components: *Libra Virtual Machine (Libra VM)*, *Virtual Simple Disk*, and *TVPL Introspector*.

Libra Virtual Machine (Libra VM). It is a novel local-booted virtual machine built on *Libra VMM* (a Type II VMM). With our local-booting technology, the *Local-Booted OS* started in *Libra VM* reproduces accurately the software environment of *Host OS*, along with the potential stealth malware. This feature makes Libra hold predominance over previous VM-based solutions with the capability of examining the preinstalled host environment at virtualization layer.

Virtual Simple Disk Based on Volume Snapshot. A key challenge to accomplish local-booting VM is reusing the existing system volume wherein *Host OS* has been preinstalled. While *Libra VM* is running, the *Local-Booted OS* is unaware of the modifications made by *Host OS* and vice versa. So they likely crash because of the content inconsistency between the file system and disk drive. Libra resolves these conflicts by dint of the *Virtual Simple Disk* based on *Volume Snapshot*. *Volume Snapshot* introduces *Copy-on-Write* mechanism to shield the modification effects of *Host OS* from *Libra VM* and vice versa. *Virtual Simple Disk* acts as the virtual storage device to combine and export the *Volume Snapshots* to *Libra VM*. Thus, *host OS environment reproduction* is achieved.

TVPL Introspector. To retrieve the "true" process list from within the lowest reaches of a system, Libra implements four introspectors within the virtualized hardware devices layer. *Libra Process Introspector* (*LPI*) is located in the *Virtual CPU*.

Assumption on trusted Libra VM. In this paper, we make an assumption that the stealth malware in host OS will not tamper the Libra program. This is practical because the ultimate target of the stealth malware is hiding its presence and extending its lifetime as long as possible. However, subverting a closed-source program often means unstable reverse-engineering and likely crashes the sacrificed program. Examining existing popular stealth malware programs, all of them try to attack the protection mechanism used by the stealth malware detectors instead of tampering these detectors to evade from detection at the risk of being exposed. Note that this assumption is consistent with that of all of the security research efforts which focuses on a compromised computer system, such as anti-virus programs, anti-rootkits software and so on.

3. Implementation of Libra

The Libra architecture discussed in Section 2 is OS-independent. Considering Windows and Intel processors are prevalent on PC platforms, Libra has been firstly implemented in Windows with Intel x86 processors.

To accomplish the two features of Libra, we are faced with two challenges: *local-booting technology with Intel x86 processors virtualization* and *implicit introspection of TVPL*. The detailed description about how we implement the local-booted VM for Intel x86 processors has been presented in our previous work [12]. This section focuses on the implementation of *implicit introspection of TVPL*.

Intel x86 processors use a two-level, in-memory, architecturally-defined page table. The page table is organized as a tree with a single fixed-sized, commonly 4KB, memory page called the *page directory* at its root. A single address space is active per processor at any given time. The address of the page directory is therefore characteristic of a single address space.

Consequently, we use the physical address of the page directory as the process unique identifier (*PUID*). OS informs the processor's memory management unit (MMU) that a new address space should become active by writing the physical address of the page directory for the new address space into a specific control register - *CR3 register*. Within the local-booted VM, access to the CR3 register is privileged and Libra VMM will catch this event and emulate its semantic. By tracking virtual address space creation and destruction within Libra VMM, *LPI* implicitly obtains the information about the events of process creations and exits, namely the *TVPL*.

LPI maintains TVPL using a Black-Red tree. Each node indexed by PUID refers to a "true" running process in the OS. The key to maintain the TVPL is analyzing the process creation and destruction.

Process Creation. The creation of a process always refers to a new address space represented by a new *PUID*. If we observe a PUID value being writing to CR3 that has not been seen in the TVPL, we can infer that a new address space has been created. Thus, LPI will fetch the process information and insert it along with its PUID into the TVPL.

Process Destruction. Our approach to detect process destruction stems from the mechanism used by an OS to

Trend Micro, Inc.
EX1002-127

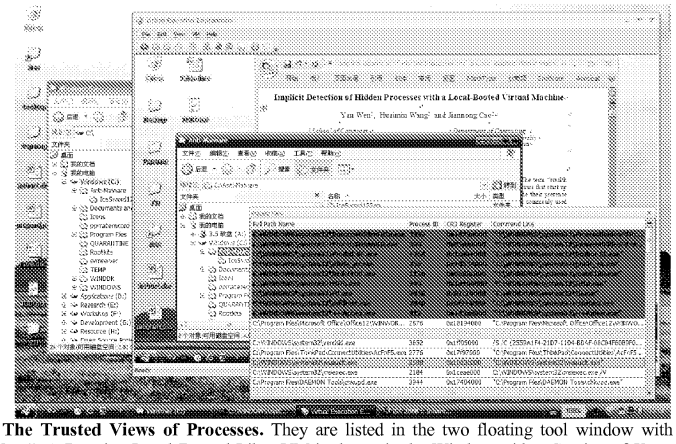


Figure 2. The Trusted Views of Processes. They are listed in the two floating tool window with the title of "*Process View*". A Running Local-Booted Libra VM is shown in the Window with a Caption of *Virtual Execution Environment*. The resolution of Local-Booted Windows is 1024x600 while the resolution of host Windows is 1280x800, so the icon arrangement within its desktop differs from that of Host Windows. As shown in this Figure, the programs of *Explorer* and *Word 2007* are running in this *Local-Booted Windows*. The topmost window is Explorer which is browsing the volume *C:*. The detected hidden processes are highlighted with red background.

deallocate the address space. Once a process has exited, Windows (and Linux) systematically will clear the non-privileged portions of its page table pages used prior to reusing them. Privileged portions of the pages used to implement the protected kernel address space need not be cleared because they are shared between processes and map memory not accessible to user-mode programs. But the non-paged memory pages among these un-cleared privileged pages must be set un-present (a bit in a page entry in the page table).

So, to detect the address space deallocations, LPI sends a request to the memory manger of Libra VMM to remap the address of a specific non-paged kernel-mode memory page for each process in TVPL. If the remapped page is set un-present, the process should have been destructed. In this case, LPI will remove the tree node of this process from TVPL.

Picking a kernel-mode non-paged memory page for each process is an OS-dependent issue. This address of this page must satisfy three requirements: 1) being assigned as soon as a process is created; 2) being fixed among a process's lifeline; 3) being accessible among a process's lifeline; 4) being unsubvertable for privileged malware.

On Windows platforms, we select the memory page containing the *Executive Process Block (EPROCESS)* which meets all of the four requirements and consists of all the most important control information about a process. The address of *EPROCESS* is stored in the *Executive Thread Block (ETHREAD)* while the *ETHREAD* address of an active thread (0xFFDFF000+ 0x124) is immutable on x86 Windows platforms. Tampering the *EPROCESS* entry in *ETHREAD* structure will break up the association between a thread and its process, or even crash the Windows kernel.

Considering the peculiarity of Windows, only if the active PUID refers to ***System*** process, we are supposed to do the remapping operations because Windows will schedule **System** process to synchronize some kernel data structures after a process exits. Consequently, we significantly decrease the count of the address remapping operations.

To construct the untrusted view of processes, we call *ZwSetSystemInformation*, a Windows native API, with a process information related parameter termed of *SystemProcessesAndThreadsInformation* to enumerate all the processes. This is the most common API to list the running processes.

4. Evaluation

We have evaluated the local-booting capability of Libra on several desktop PCs with different hardware equipments[12]. In this section, to demonstrate the new implicit process introspection capability, we will evaluate our prototype with various real-world process-

Trend Micro, Inc.
EX1002-128

Table 1 Completeness of Process List Introspection. The three columns refer to respective test results in three Windows versions: Windows 2000 SP4, Windows XP SP2 and Windows Server 2003 SP1.

	2000	**XP**	**2003**
Process Creation	2200	2200	2200
Address Space Creation	2200	2200	2200
Detected by Libra	2200	2200	2200
Process Exit	2200	2200	2200
Address Space Destruction	2200	2200	2200
Detected by Libra	2200	2200	2200

hiding rootkits which are widely used by stealth malware. Each of them adopts a different representative process-hiding technique. For a desktop-oriented workload, the performance evaluation results show that Libra only suffers a slowdown over native of 0.62-6.28%, with a 2.18% average slowdown for *Libra VMM* [12].

Hidden Process Detection. A running Libra is shown in Figure 2. This sub-section will describe the test set we used to evaluate Libra at length.

Firstly, we quantify the completeness of the implicit TVPL introspection. Our evaluation uses multiple versions of uncompromised Windows OSes which have been instrumented to report process creation, exit, and context switch. Table 1 reports the process and address space event counts gathered by the OS interfaces and by Libra during an experiment utilizing a process intensive workload. The workload creates 2200 processes, each of which runs for 25 seconds then exits. We create 10 processes per second using the *CreateProcess* API. As shown in Table 1, in this tested case, all process-related events reported by our instrumented OSes are detected by the Libra. Furthermore, Libra can precisely detect address space events, indicating that there is a one-to-one match between address spaces and processes for these OSes.

We then evaluate Libra with five popular process-hiding rootkits. *Aphex* [13] modifies the Import Address Table entry for the *ZwQuerySystemInformation* API to intercept the process list queries . *Hacker Defender* [14] also hijacks the queries with another technique: tempering the first few machine codes of *NtQuerySystemInformation* with a "jmp" instruction. *FU* [15] hides a process by using so-called *Direct Kernel Object Manipulation* (*DKOM*) technique to remove its corresponding entry from the *Active Process List* (a kernel data structure in Windows). *FUTo* [8] (an improved version of the *FU* rootkit) has the added ability to manipulate the *PspCidTable* without using any function calls. *PE386* proposed another powerful DKOM-based process-hiding rootkit called *phide_ex* [16] which have bypassed existing hidden process detectors.

We compare Libra to ten anti-rootkit programs which are recommended by Anti Rootkit Group. As summarized by Table 2, the first two rootkits, *Aphex*, *Hacker Defender*, can be found out by using the *Active Process List* as the truth. *FU* can be detected by all the above anti-rootkit programs while *Icesword* and *F-Secure Blacklight* fail to find the FUTo-hiding processes. Libra can detect all the processes hidden by the above five rootkits (shown in Figure 2). Evaluation results show that Libra is the only detector which can identify the hidden process of *phide_ex* (*phide_ex.exe*).

5. Related Work

Cross-view validation for hiding resource detection has been proposed and variously implemented in user-mode applications, within the OS kernel [2], inside a VMM [7]. The key difference of cross-view validation is the mechanism used to obtain the trusted view.

The VM introspection methodology is pioneered by the Livewire system [7]. VM introspection in Livewire is capable of examining low-level VM states from outside the VM. Livewire uses the explicit OS

Table 2. Experimental Results for Libra Hidden Processes Detection. *D* refers to the tool has detect the hidden process successfully while *F* denotes failure of detecting.

	Aphex	*Hacker Defender*	*FU*	FUTo	*phide_ex*
Blacklight	D	D	D	**F**	**F**
DarkSpy	D	D	D	**D**	**F**
IceSword	D	D	D	**F**	**F**
RkUnhooker	D	D	D	**D**	**F**
Bitdefender Antirootkit	D	D	D	**D**	**F**
UnHackMe	D	D	D	**F**	**F**
GMER	D	D	D	**D**	**F**
KProcCheck	D	D	D	**D**	**F**
Process Hunter	D	D	D	**F**	**F**
TaskInfo	D	D	D	**F**	**F**
Libra	D	D	D	**D**	**D**

Trend Micro, Inc.
EX1002-129

debugging information like the memory addresses of variables and the layout and semantics of compound structures to locate and interpret private kernel data types at runtime. VMwatcher [5] is another VM-based guest view introspector. It still highly depends upon the subvertable guest memory contents. As mentioned in literature [5], without the knowledge of the subverted scheduler, VMwatcher is not able to accurately identify all running processes.

VM-level services based on explicit implementation information are effective, but there are drawbacks. One drawback is that they may be just as susceptible to evasion by an attacker that has subverted the guest OS as if they were located within the guest itself. In spite of their location at the VM-layer, these services depend on guest-level information which is still open to simple guest-level manipulation. Instead, the introspectors of Libra are independent of the guest-level information.

Besides, desktop PCs, the concern of our approach, are often preinstalled with a standard OS. However, existing VM-based approaches only deal with the newly deployed OS within the VM. They have less contribution on detecting the existing stealth malware in the existing OSes. In contrast, Libra accurately reproduced the software environment of the host OS within Libra VM and provide a way to introspect it from within a virtualization layer.

6. Conclusion

Stealth malware are a current and alarming security issue. In this paper, we have presented a new VM-based approach called Libra to detect hidden processes implicitly. Like previous VM-based security services, Libra is resilient to kernel-mode guest malware attack by virtue of its location within a VMM layer.

Unlike prior VMM-layer process hiding detectors, by introducing a novel local-booting technology, Libra achieves the unique capability of introspecting the preinstalled host OS at virtualization layer. Moreover, Libra adopts a novel implicit process introspection technology which is decoupled from the subvertable guest OS information. Our evaluation results with real-world stealth rootkits prove Libra's detection capability.

Acknowledgements

This research is supported by National Basic Research Program of China under Grant No. 2005CB321801, National High-Tech Research and Development Plan of China under Grant No. 2007AA010301, and National Science Fund for Outstanding Youths under Grant No. 60625203.

References

[1] "Zombie PCs: Silent, Growing Threat. PC World," http://www.pcworld.com/news/article/0,aid,116841,00.asp, July 2004.

[2] Y.-M. Wang, D. Beck, B. Vo, R. Roussev, and C. Verbowski, "Detecting Stealth Software with Strider GhostBuster," in *Proceedings of 35th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'05)*, 2005, pp. 368–377.

[3] R. Naraine, "Microsoft: Stealth Rootkits Are Bombarding XP SP2 Boxes," http://www.eweek.com/article2/0,1895,1896605,00.asp, December 2005.

[4] A. Joshi, S. T. King, G. W. Dunlap, and P. M. Chen, "Detecting Past and Present Intrusions through Vulnerability-Specific Predicates," in *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP'05)*, Brighton, United Kingdom, 2005, pp. 91 - 104.

[5] X. Jiang, X. Wang, and D. Xu, "Stealthy Malware Detection Through VMM-Based "Out-of-the-Box" Semantic View Reconstruction," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*, 2007.

[6] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen, "ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay," in *Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[7] T. Garfinkel and M. Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection," in *Proceedings of Network and Distributed System Security Symposium (NDSS'03)*, 2003.

[8] P. Silberman and C.H.A.O.S, "FUTo: Bypassing Blacklight and IceSword," https://www.rootkit.com/newsread.php?newsid=433, 2007.

[9] "Effective file hiding : Bypassing Raw File System I/O Rootkit Detector," http://www.rootkit.com/newsread.php?newsid=690.

[10] "Bypassing Klister 0.4 with No Hooks or Running a Controlled Thread Scheduler," http://hi-tech.nsys.by/33/.

[11] R. P. Goldberg, "Architectural Principles for Virtual Computer Systems, Ph.D. Thesis," Cambridge, MA: Harvard University, 1972.

[12] Y. Wen, J. Zhao, and H. Wang, "A Novel Approach for Untrusted Code Execution," in *Proceedings of 9th International Conference on Information and Communications Security (ICICS'07)*, 2007, pp. 398-411.

[13] "Aphex: AFX Windows Rootkit 2003," http://www.iamaphex.cjb.net.

[14] "Hacker Defender," http://hxdef.org/.

[15] fuzen_op, "FU Rootkit," http://www.rootkit.com/project.php?id=12.

[16] PE386, "phide_ex -untimate process hiding example," http://forum.sysinternals.com/printer_friendly_posts.asp?TID=8527.

Trend Micro, Inc.
EX1002-130

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13460655	SCHNEIDER, JEROME L.
	Examiner	Art Unit
	GREGORY LANE	2438

CPC- SEARCHED		
Symbol	Date	Examiner
G06F 21/56	4/17/2014	GL
G06F 21/566	4/17/2014	GL
G06F 21/554	4/17/2014	GL

CPC COMBINATION SETS - SEARCHED		
Symbol	Date	Examiner

US CLASSIFICATION SEARCHED			
Class	Subclass	Date	Examiner
713	2,193	10/7/2013	GL

SEARCH NOTES		
Search Notes	Date	Examiner
searched for the inventive concept of kernel level malware	10/7/2013	GL
searched the inventors' names to check for double patenting violation	10/7/2013	GL
considered 35 USC 101 and 112 violations	10/7/2013	GL
Performed EAST search	10/7/2013	GL
consulted Taghi Arani, SPE AU2438	4/9/2014	GL
updated EAST search using CPC classification	4/17/2014	GL
updated NPL search	4/17/2014	GL

INTERFERENCE SEARCH			
US Class/ CPC Symbol	US Subclass / CPC Group	Date	Examiner

/GREGORY LANE/ Examiner.Art Unit 2438	

Trend Micro, Inc.
EX1002-131

INTERFERENCE SEARCH			
US Class/ CPC Symbol	**US Subclass / CPC Group**	**Date**	**Examiner**
713	2,193 AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	4/17/2014	GL
G06F	21/56 AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	4/17/2014	GL
G06F	21/566 AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	4/17/2014	GL
G06F	21/554 AND (managing AND pestware AND monitoring AND boot AND sequence AND related AND events AND first AND period AND configured AND run AND native AND application AND native AND loaded AND set AND behavior AND rule AND second AND generating AND record AND analyzing AND identify AND scanning AND data AND registry).CLM.	4/17/2014	GL

/GREGORY LANE/ Examiner.Art Unit 2438	

Trend Micro, Inc.
EX1002-132

PTO/SB/06 (09-11)
Approved for use through 1/31/2014. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 13/460,655	Filing Date 04/30/2012	☐ To be Mailed

ENTITY: ☐ LARGE ☒ SMALL ☐ MICRO

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $310 ($155 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).			
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))				
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT **04/24/2014**

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	* 20	Minus	** 20	= 0	X $40 =	0
Independent (37 CFR 1.16(h))	* 3	Minus	***3	= 0	X $210 =	0
☐ Application Size Fee (37 CFR 1.16(s))						
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))						
					TOTAL ADD'L FEE	0

AMENDMENT

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))						
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))						
					TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

LIE
/VIOLA ROGERS/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**
If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Doc Code: DIST.E.FILE **Document Description: Electronic Terminal Disclaimer - Filed**	PTO/SB/26 U.S. Patent and Trademark Office Department of Commerce

Electronic Petition Request	**TERMINAL DISCLAIMER TO OBVIATE A DOUBLE PATENTING REJECTION OVER A "PRIOR" PATENT**
Application Number	13460655
Filing Date	30-Apr-2012
First Named Inventor	Jerome Schneider
Attorney Docket Number	6890-45-CON
Title of Invention	Malware Management Through Kernel Detection During a Boot Sequence

☒ Filing of terminal disclaimer does not obviate requirement for response under 37 CFR 1.111 to outstanding Office Action

☒ This electronic Terminal Disclaimer is not being used for a Joint Research Agreement.

Owner	Percent Interest
Webroot, Inc.	100%

The owner(s) with percent interest listed above in the instant application hereby disclaims, except as provided below, the terminal part of the statutory term of any patent granted on the instant application which would extend beyond the expiration date of the full statutory term of prior patent number(s)

8190868

as the term of said prior patent is presently shortened by any terminal disclaimer. The owner hereby agrees that any patent so granted on the instant application shall be enforceable only for and during such period that it and the prior patent are commonly owned. This agreement runs with any patent granted on the instant application and is binding upon the grantee, its successors or assigns.

In making the above disclaimer, the owner does not disclaim the terminal part of the term of any patent granted on the instant application that would extend to the expiration date of the full statutory term of the prior patent, "as the term of said prior patent is presently shortened by any terminal disclaimer," in the event that said prior patent later:
- expires for failure to pay a maintenance fee;
- is held unenforceable;
- is found invalid by a court of competent jurisdiction;
- is statutorily disclaimed in whole or terminally disclaimed under 37 CFR 1.321;
- has all claims canceled by a reexamination certificate;
- is reissued; or
- is in any manner terminated prior to the expiration of its full statutory term as presently shortened by any terminal disclaimer.

◉ Terminal disclaimer fee under 37 CFR 1.20(d) is included with Electronic Terminal Disclaimer request.

Trend Micro, Inc.
EX1002-134

○ I certify, in accordance with 37 CFR 1.4(d)(4), that the terminal disclaimer fee under 37 CFR 1.20(d) required for this terminal disclaimer has already been paid in the above-identified application.
Applicant claims the following fee status: ◉ Small Entity ○ Micro Entity ○ Regular Undiscounted I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.
THIS PORTION MUST BE COMPLETED BY THE SIGNATORY OR SIGNATORIES I certify, in accordance with 37 CFR 1.4(d)(4) that I am: ◉ An attorney or agent registered to practice before the Patent and Trademark Office who is of record in this application Registration Number 67424 ○ A sole inventor ○ A joint inventor; I certify that I am authorized to sign this submission on behalf of all of the inventors as evidenced by the power of attorney in the application ○ A joint inventor; all of whom are signing this request

Signature	/Brent R. Lindon/
Name	Brent R. Lindon

*Statement under 37 CFR 3.73(b) is required if terminal disclaimer is signed by the assignee (owner).
Form PTO/SB/96 may be used for making this certification. See MPEP § 324.

Electronic Patent Application Fee Transmittal

Application Number:	13460655
Filing Date:	30-Apr-2012
Title of Invention:	Malware Management Through Kernel Detection During a Boot Sequence
First Named Inventor/Applicant Name:	Jerome L. Schneider
Filer:	Brent Robert Lindon/Kathy Johnston
Attorney Docket Number:	6890-45-CON

Filed as Small Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Statutory or Terminal Disclaimer	1814	1	160	160
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**160**

Doc Code: DISQ.E.FILE
Document Description: Electronic Terminal Disclaimer – Approved

Application No.: 13460655

Filing Date: 30-Apr-2012

Applicant/Patent under Reexamination: Schneider et al.

Electronic Terminal Disclaimer filed on April 17, 2014

☒ APPROVED

This patent is subject to a terminal disclaimer

☐ DISAPPROVED

Approved/Disapproved by: Electronic Terminal Disclaimer automatically approved by EFS-Web

U.S. Patent and Trademark Office

Electronic Acknowledgement Receipt

EFS ID:	18791228
Application Number:	13460655
International Application Number:	
Confirmation Number:	9502
Title of Invention:	Malware Management Through Kernel Detection During a Boot Sequence
First Named Inventor/Applicant Name:	Jerome L. Schneider
Customer Number:	22442
Filer:	Brent Robert Lindon/Kathy Johnston
Filer Authorized By:	Brent Robert Lindon
Attorney Docket Number:	6890-45-CON
Receipt Date:	17-APR-2014
Filing Date:	30-APR-2012
Time Stamp:	13:57:21
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$160
RAM confirmation Number	165
Deposit Account	191970
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.20 (Post Issuance fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Electronic Terminal Disclaimer-Filed	eTerminal-Disclaimer.pdf	33515 a6c3c4fe9514371dc8f602120d4eece6441b74a4	no	2
Warnings:					
Information:					
2	Fee Worksheet (SB06)	fee-info.pdf	30378 68a2f41b2e6f18d3057d06f8f3c449f9c19d5328	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	63893		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.: 13/460,655	)	Confirmation No.: 9502
Applicant: Jerome L. Schneider	)	TC/A.U.: 2438
Filed: 04/30/2012	)	Examiner: Lane, Gregory A
Docket No.: 6890-45-CON	)	
	)	***Filed Electronically***
For: "MALWARE MANAGEMENT THROUGH KERNEL DETECTION DURING A BOOT SEQUENCE"	)	

AMENDMENT AND RESPONSE

Mailstop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Madam:

Applicant submits this Amendment and Response to address the Office Action having a mailing date of October 18, 2013. Applicant considers this filing to be timely since the U.S. Patent and Trademark Office was closed on January 21, 2014 due to adverse weather; accordingly, as stated in MPEP section 510(I), "[a]ny action or fee due on such a day may be taken, or fee paid, on the next succeeding business day the Office is open." Although Applicant believes that no fees are due in connection with filing this Amendment and Response, please charge any fees deemed necessary to Deposit Account No. 19-1970.

AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method for managing pestware on a computer comprising:

monitoring events during a boot sequence of the computer;

managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence ~~occuring~~occurring before the computer becomes configured to run native applications, before a subsystem of an operating system is loaded, and after a kernel is loaded;

managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence ~~occuring~~occurring when the computer is configured to run native applications; and

scanning data in a registry of the computer for pestware during the second period in the boot sequence.

2. (Original) The method of claim 1, further comprising launching, after an operating system is initiated, a pestware management engine; and

wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

3. (Original) The method of claim 1, further comprising:

generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;

analyzing the record of events so as to identify the pestware-related events; and

modifying the set of behavior rules so as to prevent the pestware related events.

4. (Original) The method of claim 3, wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information.

5. (Original) The method of claim 1, further comprising:

Trend Micro, Inc.
EX1002-142

managing pestware-related events after the computer becomes configured to run native applications.

6. (Original) The method of claim 1, wherein the monitoring includes monitoring the boot sequence while boot drivers are initiated.

7. (Currently Amended) A system for managing pestware on a computer comprising:

a processor;

a memory comprising:

a first instruction set configured to monitor events during a boot sequence of the computer;

a second instruction set configured to manage pestware-related events during first period in a boot sequence of the computer, the first period in the boot sequence ~~occuring~~ occurring before the computer becomes configured to run native applications, before a ~~Win32~~ WIN32 subsystem is loaded, and after a kernel is loaded;

a third instruction set configured to manage pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence ~~occuring~~occurring when the computer is configured to run native applications; and

a fourth instruction set configured to scan data in a registry of the computer for pestware during the second period in the boot sequence.

8. (Original) The system of claim 7, wherein the memory further comprises a fifth instruction set configured to launch, after an operating system is initiated, a pestware management engine; and wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

9. (Original) The system of claim 7, wherein the memory further comprises:

a sixth instruction set configured to generate, in response to the monitoring, a record of events, the record of events including the pestware-related events;

Trend Micro, Inc.
EX1002-143

a seventh instruction set configured to analyze the record of events so as to identify the pestware-related events; and

an eighth instruction set configured to modify the set of behavior rules so as to prevent the pestware related events.

10. (Original) The system of claim 9, wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information.

11. (Original) The system of claim 7, wherein the memory further comprises:

a ninth instruction set configured to manage pestware-related events after the computer becomes configured to run native applications.

12. (Original) The system of claim 7, wherein the first instruction set is further configured to monitor the boot sequence while boot drivers are initiated.

13. (Currently Amended) A non-transitory tangible computer-readable storage medium comprising program instructions for:

monitoring events during a boot sequence of ~~the~~ a computer;

managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence ~~occuring~~ occurring before the computer becomes configured to run native applications, before a ~~Win32~~ WIN32 subsystem is loaded, and after a kernel is loaded;

managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence ~~occuring~~occurring when the computer is configured to run native applications; and

scanning data in a registry of the computer for pestware during the second period in the boot sequence.

Trend Micro, Inc.
EX1002-144

14. (Original) The non-transitory tangible computer-readable storage medium of claim 13 further comprising program instructions for launching, after an operating system is initiated, a pestware management engine; and

wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

15. (Original) The non-transitory tangible computer-readable storage medium of claim 13 further comprising program instructions for:

generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;

analyzing the record of events so as to identify the pestware-related events; and

modifying the set of behavior rules so as to prevent the pestware related events.

16. (Original) The non-transitory tangible computer-readable storage medium of claim 15, wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information.

17. (Original) The non-transitory tangible computer-readable storage medium of claim 13 further comprising program instructions for:

managing pestware-related events after the computer becomes configured to run native applications.

18. (Original) The non-transitory tangible computer-readable storage medium of claim 13, wherein the monitoring includes monitoring the boot sequence while boot drivers are initiated.

19. (New) The method of claim 2, further comprising modifying, by the pestware management engine, the behavior rules utilized by a kernel-level monitor, wherein the kernel-level monitor manages pestware-related events in the first and second periods in the boot sequence.

Trend Micro, Inc.
EX1002-145

20. (New) The method of claim 3, further comprising modifying the set of behavior rules based on the scanned registry so as to prevent the pestware-related events.

Trend Micro, Inc.
EX1002-146

REMARKS

Status of the Claims

The above-identified patent application has been reviewed in light of the Examiner's Action dated October 18, 2013. Claims 1, 7, and 13 have been amended, without intending to abandon or to dedicate to the public any patentable subject matter. Claims 19-20 have been added. Accordingly, claims 1-20 are now pending. As set forth herein, reconsideration and withdrawal of the rejections of the claims are respectfully requested.

Examiner Interview

Applicant would like to thank Examiner Lane for his time during the telephonic interview on January 20, 2014. Examiner Lane and Applicant's representative, Brent Lindon, discussed the current rejections to clarify the issues at hand and further discussed possible additions to the claims. The changes discussed are some of the changes included in the current amendments. Again, Applicant appreciates the Examiner's attention and assistance.

Terminal Disclaimer

1-18 have been rejected on the ground of nonstatutory double patenting as being unpatentable over claims 1-5, 9, 10, 12, and 14 of U.S. Patent No. 8,190,868. In the event that the Examiner maintains this rejection in view of the amendments presented herein, Applicant is prepared to submit a Terminal Disclaimer to address this rejection after the remaining rejections have been properly withdrawn and the application is otherwise in condition for allowance.

Claim Objections

As indicated in the Office Action dated October 18, 2013, Win32 has been amended to WIN32. Accordingly, Applicant respectfully submits that the amendment overcomes this rejection.

Claim Rejections Under 35 U.S.C. §112

Trend Micro, Inc.
EX1002-147

Claim 13 was rejected under 35 U.S.C. §112 second paragraph; claim 13 has been amended to overcome this rejection. Accordingly, Applicant respectfully submits that the rejection of claim 13 under 35 U.S.C. §112 be reconsidered and withdrawn.

Claim Rejections Under 35 U.S.C. §103

Claims 1, 2, 5-8, 11-14, 17, and 18 have been rejected under 35 U.S.C. 103(a) as being unpatentable over US 2005/0114687 to Zimmer et al. (hereinafter "Zimmer") in view of WO 2006077443 to Rothwell (hereinafter "Rothwell") and further in view of U.S. Patent Application No. 2005/0216759 to Rothman et al. (hereinafter "Rothman"). Claims 3, 9, and 15 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Zimmer, Rothwell, Rothman, and further in view of U.S. Patent Application No. 2005/0125687 to Townsend et al. (hereinafter "Townsend"). Claims 4, 10, and 16 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Zimmer, Rothwell, Rothman, Townsend, and further in view of U.S. Patent Application No. 2003/0135791 to Natvig (hereinafter "Natvig").

To establish a *prima facie* case of obviousness, all claim limitations must first be taught or suggested by the prior art. *See, e.g., DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.*, 464 F.3d 1356, 1360 (Fed. Cir. 2006); 80 U.S.P.Q.2d 1641, 1645 (2006). "All words in a claim must be considered in judging the patentability of that claim against the prior art." MPEP § 2143.03 (citing *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970)). The Examiner must then provide an explicit analysis of the motivation for combining the references. *See KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007); 82 U.S.P.Q.2d 1385, 1396 (2007) ("a patent composed of several elements is not proved obvious merely by demonstrating that each of its elements was, independently, known in the prior art . . . ."). While the Examiner can choose one of several exemplary rationales from the MPEP to support an obviousness rejection, all the rationales in the MPEP still require the Examiner to demonstrate that all the claim elements are shown in the prior art. *See* MPEP § 2143, Original Eighth Edition, August 2001, Latest Revision July 2008. Applicant respectfully submits that the cited references fail to teach, suggest, or render obvious at least the combination of features as claimed.

Trend Micro, Inc.
EX1002-148

The claimed invention is generally directed to managing pestware in a boot sequence of a protected computer. As one example, a native scanner may be initialized early during the period of the boot sequence during which native applications may be run. *See e.g., Applicant's specification-as-published, ¶ 0039.* The native scanner may be configured to examine registry, file system, and other operation of a protected computer before the WIN32 subsystem and most services are loaded. *See I.d.* In this way, additional information may be generated for use by a pestware management engine, for example used to generate new behavior rules for a kernel-level monitor. *See Id.*. In particular, claim 1 recites, *inter alia, managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications; and* scanning data in a registry of the computer for pestware during the second period in the boot sequence. Applicant respectfully submits that none of the cited references teach, suggest, or render obvious at least this feature.

For example, Zimmer is generally directed to pre-boot code which can be configured to protect the firmware resources in a pre-boot environment and send a firmware resource protection request to a post-boot environment, thereby enabling a secure kernel to protect the firmware resources in the post-boot environment. *See e.g., Zimmer,* ¶ 0017. In particular, Zimmer teaches a pre-boot firmware that is executed in the pre-boot environment and is configured to initialize firmware resources and establish a hardware/software interface for use in the post-boot environment. *See e.g., Zimmer,* ¶ 0019. In particular, the pre-boot firmware may be configured to select the protected firmware resources in a memory space and generate a resource protection list based on the protected firmware resources. *See e.g., Zimmer,* ¶ 0018. Some example firmware resources provided by Zimmer include hardware interfaces (e.g., display output, keyboard input, disk drive operation, etc.), firmware code resources (e.g., the pre-boot firmware 102), and firmware data resources. See *e.g., Zimmer*, ¶ 0019. Although Zimmer teaches a boot process implemented by firmware and/or software that may be implemented by, for example, the pre-boot firmware, the operating system, or applications of an untrusted partition and that the boot process may be used to boot/launch a secure kernel, retrieve the resource protection list, and establish firmware

Trend Micro, Inc.
EX1002-149

resource protection policies, (See *e.g., Zimmer*, ¶ 0015), nowhere does Zimmer teach or suggest at least managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications **and** scanning data in a registry of the computer for pestware during the second period in the boot sequence. That is, nowhere does Zimmer teach where the protection policies are established before the computer is configured to run native applications. Moreover, nowhere does Zimmer teach that data in a registry of the computer is scanned for pestware during this second period.

Moreover, Rothwell fails to overcome the deficiencies of Zimmer. In particular, Rothwell provides:

> In a preferred embodiment the program code means are adapted to cause the computer to run the filter module as a file system filter driver **in a kernel of the operating system**. This assures that all read/write access requests originated from any user mode application or from any system process, which are directed to any data storage device associated to the computer, are treated by the filter module. The filter module **running in the kernel** does not have any additional impact on the behaviour or the performance of the computer. Since the kernel of the operating system is launched at a very early stage of the booting process of the computer, the malware protection is effective from scratch and especially before system processes are running. Furthermore, running the filter module in the kernel allows a malware protection irrespective of users' operating system access rights or the operating system access rights of a user simulated by the malware.

See e.g.,Rothwell, page 8 line 31-page 9 line 9 (*emphasis added).* That is, although Rothwell teaches that an embodiment is integrated into the file system and loaded automatically during system boot up sequence (See e.g., Rothwell, page 18, lines 30-31), nowhere does Rothwell disclose at least managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a subsystem of an operating system is loaded, and after a kernel is loaded. Moreover, Applicant respectfully submits that there is no disclosure in Rothwell pertaining to managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications; and scanning data in a registry of the computer for pestware during the second period in the boot sequence. Accordingly, Zimmer and Rothwell, taken alone, or in any combination, do not teach the combination of features as claimed.

In addition, the Office relies on paragraph [0040] of Rothman. Paragraph [0040] of Rothman discloses the following:

> In another embodiment of the invention, the virus scanner performs behavioral checking of input/output activity. Behavioral checking involves identifying behavior that is non-normal even though a virus has not been detected. For example, the virus scanner may notice repeated pings received at a network interface card of the computer system. Such behavior may indicate a denial-of-service attack on the computer system. In another example, the virus scanner may detect an attempt to modify the master boot record. In yet another example, the virus scanner may detect suspicious reads of system files, such as registry information, that indicate a virus is looking for vulnerabilities in the computer system.

See e.g., Rothman, ¶ 0040. That is, as disclosed above, nowhere does Rothman teach at least scanning data in a registry of the computer for pestware during the second period in the boot sequence. Instead, Rothman clearly teaches detecting suspicious reads of system files, such as registry information, that indicate a virus is looking for vulnerabilities in the computer system. Detecting suspicious reads of a registry is not the same as scanning data in a registry of the computer for pestware during the second period in the boot sequence. Accordingly, at least this feature is not taught or suggested by Rothman. Therefore, Zimmer, Rothwell, and Rothman, taken alone, or in any combination do teach at least the combination of features in the pending claims. Accordingly, Applicant respectfully submits that the rejections of claims 1-18 should be reconsidered and withdrawn.

Independently allowable dependent claims

In addition to depending from allowable independent claims, the dependent claims provide additional and independent reasons for allowance. For instance, dependent claim 19 requires that the behavior rules utilized by a kernel-level monitor are modified, wherein the kernel-level monitor manages pestware-related events in the first and second periods in the boot sequence. There is no teaching in Townsend related to modifying behavior rules utilized by a kernel-level monitor. Accordingly, the rejection of claim 19 should be reconsidered and withdrawn.

Trend Micro, Inc.
EX1002-151

Conclusion

The application now appearing to be in form for allowance, early notification of same is respectfully requested. Applicant does not acquiesce to any objection, rejection, or argument not specifically addressed herein. Rather, the Applicant believes the amendments and arguments contained herein overcome all objections, rejections, or arguments.

The Examiner is invited to contact the undersigned by telephone if doing so would be of assistance.

Respectfully submitted,

SHERIDAN ROSS P.C.

Date: January 22, 2014

By: /Brent R. Lindon/

Brent R. Lindon
Reg. No. 67,424
1560 Broadway, Suite 1200
Denver, CO 80202
Telephone: 303-863-9700

Trend Micro, Inc.
EX1002-152

Electronic Acknowledgement Receipt

EFS ID:	17994661
Application Number:	13460655
International Application Number:	
Confirmation Number:	9502
Title of Invention:	Malware Management Through Kernel Detection During a Boot Sequence
First Named Inventor/Applicant Name:	Jerome L. Schneider
Customer Number:	22442
Filer:	Brent Robert Lindon
Filer Authorized By:	
Attorney Docket Number:	6890-45-CON
Receipt Date:	22-JAN-2014
Filing Date:	30-APR-2012
Time Stamp:	17:39:28
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		Amend_01.pdf	145990 16a271d3d475d515207d401999d12a3bb399f4d9	yes	12

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Amendment/Req. Reconsideration-After Non-Final Reject	1	1
	Claims	2	6
	Applicant Arguments/Remarks Made in an Amendment	7	12
Warnings:			
Information:			
	Total Files Size (in bytes):	145990	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (09-11)
Approved for use through 1/31/2014. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 13/460,655	Filing Date 04/30/2012	☐ To be Mailed

ENTITY: ☒ LARGE ☐ SMALL ☐ MICRO

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	RATE ($)	FEE ($)
☒ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A	**380**
☒ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A	**620**
☒ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A	**250**
TOTAL CLAIMS (37 CFR 1.16(i))	18 minus 20 =	* 0	x $60 =	**0**
INDEPENDENT CLAIMS (37 CFR 1.16(h))	3 minus 3 =	* 0	x $250 =	**0**
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $310 ($155 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).			
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))				
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL	**1250**

APPLICATION AS AMENDED – PART II

AMENDMENT **01/22/2014**

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	* 20	Minus	** 20	= 0	x $80 =	0
Independent (37 CFR 1.16(h))	* 3	Minus	***3	= 0	x $420 =	0
☐ Application Size Fee (37 CFR 1.16(s))						
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))						
					TOTAL ADD'L FEE	**0**

AMENDMENT

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	x $ =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	x $ =	
☐ Application Size Fee (37 CFR 1.16(s))						
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))						
					TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

LIE
/DIANE JOHNSON/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**
If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/460,655	04/30/2012	Jerome L. Schneider	6890-45-CON	9502

22442 7590 10/18/2013

Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

EXAMINER	
LANE, GREGORY A	
ART UNIT	PAPER NUMBER
2438	

NOTIFICATION DATE	DELIVERY MODE
10/18/2013	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

e-docket@sheridanross.com

Office Action Summary	Application No. 13/460,655	Applicant(s) SCHNEIDER, JEROME L.	
	Examiner GREGORY LANE	Art Unit 2438	AIA (First Inventor to File) Status No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *4/30/12*.

☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on _____.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

4)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

5)☒ Claim(s) *1-18* is/are pending in the application.

5a) Of the above claim(s) _____ is/are withdrawn from consideration.

6)☐ Claim(s) _____ is/are allowed.

7)☒ Claim(s) *1-18* is/are rejected.

8)☐ Claim(s) _____ is/are objected to.

9)☐ Claim(s) _____ are subject to restriction and/or election requirement.

* If any claims have been determined allowable, you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov.

Application Papers

10)☐ The specification is objected to by the Examiner.

11)☒ The drawing(s) filed on *4/30/12* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

Certified copies:

a)☐ All b)☐ Some * c)☐ None of the:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

3) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

4) ☐ Other: _____.

Trend Micro, Inc.
EX1002-157

The present application is being examined under the pre-AIA first to invent provisions.

DETAILED ACTION

1. The following is a non-Final Office Action in response to applicant's filing on April 30, 2012

Claims 1-18 are pending

Examiner's Note: Claim 7 does not invoke 35 U.S.C. 112, 6th Paragraph since the term "instruction…configured" is a structural term.

Drawings

1. Acknowledgment is made of applicant's drawings submitted on 4/30/2012.

Oath/Declaration

2. Acknowledgment is made of applicant's oath submitted on 4/30/2012

Application Data Sheet

3. Acknowledgment is made of applicant's application data sheet submitted on 4/30/2012.

35 USC § 112 Rejection

The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention

Claim 13 recites the limitation "the computer". There is insufficient antecedent basis for this limitation in the claim.

Claim Objection

1. The use of the trademark Win32 has been noted in this application. It should be capitalized wherever it appears and be accompanied by the generic terminology.

Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner which might adversely affect their validity as trademarks.

Double Patenting

1. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the “right to exclude” granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated

by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

2. Claim 1-18 are rejected on the ground of nonstatutory double patenting over claims 1-5, 9, 10, 12, and 14 of U. S. Patent No. 8,190,868 since the claims, if allowed, would improperly extend the "right to exclude" already granted in the patent.

The subject matter claimed in the instant application is fully disclosed in the patent and is covered by the patent since the patent and the application are claiming common subject matter, as follows: see table below

Furthermore, there is no apparent reason why applicant was prevented from presenting claims corresponding to those of the instant application during prosecution of

the application which matured into a patent. See *In re Schneller*, 397 F.2d 350, 158 USPQ 210 (CCPA 1968). See also MPEP § 804.

Instant Application	Patent 8,190,868
1. A method for managing pestware on a computer comprising: monitoring events during a boot sequence of the computer; managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occuring before the computer becomes configured to run native applications, before a subsystem of an operating system is loaded, and after a kernel is loaded; managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occuring when the computer is configured to run native applications; and scanning a registry of the computer for pestware during the second period in the boot sequence. 2. The method of claim 1, further comprising launching, after an operating system is initiated, a pestware management engine; and wherein the set of behavior rules includes behavior rules compiled by the pestware management engine. 3. The method of claim 1, further comprising: generating, in response to the monitoring, a record of events, the record of events including the pestware-related events; analyzing the record of events so as to identify the pestware-related events; and modifying the set of behavior rules so as to prevent the pestware related events.	1. A method for managing pestware on a computer comprising: starting a boot sequence, the boot sequence including a period when boot drivers are initialized; initiating a kernel-level monitor during the period when boot drivers are initialized; monitoring, while the boot sequence is being carried out, events with the kernel-level monitor; managing pestware-related events with the kernel-level monitor before a period in the boot sequence when the computer is configured to run native applications, the period in the boot sequence when the computer is configured to run native applications being after a kernel is loaded and before a Win32 subsystem is loaded; loading and initializing a native scanner during the period in the boot sequence when the computer is configured to run native applications; managing pestware-related events during the period in the boot sequence when the computer is configured to run native applications; acquiring a set of behavior rules, wherein the managing pestware-related events is carried out in accordance with the behavior rules; and scanning, using the native scanner, a registry of the computer for pestware during the period in the boot sequence when the computer is configured to run native applications.

4. The method of claim 3, wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information. 5. The method of claim 1, further comprising: managing pestware-related events after the computer becomes configured to run native applications. 6. The method of claim 1, wherein the monitoring includes monitoring the boot sequence while boot drivers are initiated. 7. A system for managing pestware on a computer comprising: a processor; a memory comprising: a first instruction set configured to monitor events during a boot sequence of the computer; a second instruction set configured to manage pestware-related events during first period in a boot sequence of the computer, the first period in the boot sequence occuring before the computer becomes configured to run native applications, before a Win32 subsystem is loaded, and after a kernel is loaded; a third instruction set configured to manage pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occuring when the computer is configured to run native applications; and a fourth instruction set configured to scan a registry of the computer for pestware during the second period in the boot sequence.	2. The method of claim 1, including: launching, after an operating system is initiated, a pestware management engine; wherein the acquiring the set of behavior rules includes acquiring at least a portion of the behavior rules from behavior rules compiled by the pestware management engine. 3. The method of claim 1, including: generating, in response to the monitoring, a record of events, the record of events including the pestware-related events; analyzing the record of events so as to identify the pestware-related events; and modifying the set of behavior rules so as to prevent the pestware-related events. 4. The method of claim 3, wherein the record of events includes information selected from the group consisting of process identification information, file identification information and hook generation information. 5. The method of claim 1, including: managing pestware-related events with the kernel-level monitor after the period in the boot sequence when the computer is configured to run native applications. 9. The method of claim 8, wherein the monitoring includes monitoring the boot sequence while the boot drivers are initialized. 10. A system for managing pestware on a protected computer comprising: a processor; a kernel-level monitor executed by the processor, the kernel-level

8. The system of claim 7, wherein the memory further comprises a fifth instruction set configured to launch, after an operating system is initiated, a pestware management engine; and wherein the set of behavior rules includes behavior rules compiled by the pestware management engine. 9. The system of claim 7, wherein the memory further comprises: a sixth instruction set configured to generate, in response to the monitoring, a record of events, the record of events including the pestware-related events; a seventh instruction set configured to analyze the record of events so as to identify the pestware-related events; and an eighth instruction set configured to modify the set of behavior rules so as to prevent the pestware related events. 10. The system of claim 9, wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information. 11. The system of claim 7, wherein the memory further comprises: a ninth instruction set configured to manage pestware-related events after the computer becomes configured to run native applications. 12. The system of claim 7, wherein the first instruction set is further configured to monitor the boot sequence while boot drivers are initiated. 13. A non-transitory tangible computer-readable storage medium comprising	monitor configured to be initialized before at least a portion of boot drivers on the protected computer are initialized and to monitor, according to a set of behavior rules, activities on the protected computer before a period in a boot sequence of the protected computer when the computer is configured to run native applications, the period when the computer is configured to run native applications being after a kernel is loaded but before a Win32 subsystem is loaded; a pestware management engine executed by the processor that is configured to both be initialized after an operating system of the protected computer is initialized and to compile the set of behavior rules; and a native scanner executed by the processor that is initialized during the period in the boot sequence of the protected computer when the computer is configured to run native applications, wherein the native scanner is configured to scan files that are utilized by an operating system of the protected computer; wherein the native scanner is configured to scan a registry of the protected computer for pestware during the period in the boot sequence of the protected computer when the computer is configured to run native applications. 12. The system of claim 10, wherein the kernel-level monitor is configured to prevent, in accordance with the behavior rules, pestware from carrying out pestware-related activities. 14. The system of claim 13, wherein the pestware management engine is configured to read the event log and modify the set of behavior rules based

program instructions for: monitoring events during a boot sequence of the computer; managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occuring before the computer becomes configured to run native applications, before a Win32 subsystem is loaded, and after a kernel is loaded; managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occuring when the computer is configured to run native applications; and scanning a registry of the computer for pestware during the second period in the boot sequence. 14. The non-transitory tangible computer-readable storage medium of claim 13 further comprising program instructions for launching, after an operating system is initiated, a pestware management engine; and wherein the set of behavior rules includes behavior rules compiled by the pestware management engine. 15. The non-transitory tangible computer-readable storage medium of claim 13 further comprising program instructions for: generating, in response to the monitoring, a record of events, the record of events including the pestware-related events; analyzing the record of events so as to identify the pestware-related events; and modifying the set of behavior rules so as to prevent the pestware related events. 16. The non-transitory tangible computer-readable storage medium of claim 15, wherein the record of events includes	upon the information about potential pestware-related events so as to prevent subsequent pestware-related events during the boot sequence.

information selected from the group consisting of: process identification information, file identification information, and hook generation information. 17. The non-transitory tangible computer-readable storage medium of claim 13 further comprising program instructions for: managing pestware-related events after the computer becomes configured to run native applications. 18. The non-transitory tangible computer-readable storage medium of claim 13, wherein the monitoring includes monitoring the boot sequence while boot drivers are initiated.	

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

1.) Claims 1, 2, 5-8, 11-14, 17, and 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 20050114687, Zimmer(filed on 11/21/2003) in view of WO

2006077443, Rothwell(filed on 1/18/2006) and further in view of US 20050216759, Rothman(filed on 3/29/2004)

In regards to claim 1, Zimmer taught a method for managing pestware on a computer comprising:

monitoring events during a boot sequence of the computer**(see US 20050114687, Zimmer, para. 0026, where the secure virtual machine monitor is used to monitor events during a boot process)**; managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications**(see US 20050114687, Zimmer, para. 0050, where protection policies may be established during a boot process)**;

Zimmer did not teach managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occuring before the computer becomes configured to run native applications, before a subsystem of an operating system is loaded, and after a kernel is loaded

However, Rothwell taught managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occuring before the computer becomes configured to run native applications, before a subsystem of an operating system is loaded, and after a kernel is loaded **(see WO 2006077443, Rothwell, pg. 18, lines 30,31 and fig. 3, where the anti-malware is implemented during the boot sequence and wherein the anti-malware HDF filter operates in Kernel mode[fig. 3, item 153] to prevent the operation of malware from subsystems)**;

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teaching of Zimmer with the teaching of Rothwell because a user would have been motivated by methods of protecting a computer from malware**(see Rothwell, pg. 1, lines 1-5)**

and the combination of Zimmer and Rothwell did not teach scanning a registry of the computer for pestware during the second period in the boot sequence

However, Rothman taught scanning a registry of the computer for pestware during the second period in the boot sequence **(see US 20050216759, Rothman, para. 0040, where the virus scanner, that's initialized during a pre-boot sequence, is operable to scan a registry)**.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the modified teaching of the combination of Zimmer and Rothwell with the teaching of Rothman because a user would have been motivated by methods of scanning input/output traffic of a computer system**(see Rothman, para. 0002)**

In regards to claim 2, the combination of Zimmer, Rothwell, and Rothman taught the method of claim 1, further comprising launching, after an operating system is initiated, a pestware management engine**(see Zimmer, para. 0026, lines 5-7, where the secure virtual machine monitor is a pestware management engine that is operable after the boot phase of an operating system)**; and

wherein the set of behavior rules includes behavior rules compiled by the pestware

management engine**(see Zimmer, para. 0026, lines 11-14, where the secure virtual machine monitor establishes protection policies)**.

In regards to claim 5, the combination of Zimmer, Rothwell, and Rothman taught the method of claim 1, further comprising:

managing pestware-related events after the computer becomes configured to run native applications**(see US 20050114687 Zimmer, para. 0026, lines 5-7, where the SVMM is a kernel level monitor that is operable in either a pre or post boot phase of an operating system)**.

In regards to claim 6, the combination of Zimmer, Rothwell, and Rothman taught the method of claim 1, wherein the monitoring includes monitoring the boot sequence while boot drivers are initiated**(see Zimmer, para. 0026, lines 5-7, where the secure virtual machine monitor is initiated in the boot phase)**.

In regards to claim 7, Zimmer taught A system for managing pestware on a computer comprising:

a processor**(see US 20050114687, Zimmer, fig. 8, item 802, processor)**;

a memory**(see US 20050114687, Zimmer, fig. 8, item 806, 808, 810, memory elements)** comprising:

a first instruction set configured to monitor events during a boot sequence of the

computer**(see US 20050114687, Zimmer, para. 0026, where the secure virtual machine monitor is used to monitor events during a boot process)**;

a third instruction set configured to manage pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications**(see US 20050114687, Zimmer, para. 0050, where protection policies may be established during a boot process)**;

Zimmer did not teach a second instruction set configured to manage pestware-related events during first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a Win32 subsystem is loaded, and after a kernel is loaded

However, Rothwell taught a second instruction set configured to manage pestware-related events during first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a Win32 subsystem is loaded, and after a kernel is loaded **(see WO 2006077443, Rothwell, pg. 18, lines 30,31 and fig. 3, where the anti-malware is implemented during the boot sequence and wherein the anti-malware HDF filter operates in Kernel mode[fig. 3, item 153] to prevent the operation of malware from subsystems[e.g. Win32])**;

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teaching of Zimmer with the teaching of Rothwell because a user would have been motivated by methods of protecting a computer from malware**(see Rothwell, pg. 1, lines 1-5)**

And the combination of Zimmer and Rothwell did not teach a fourth instruction set configured to scan a registry of the computer for pestware during the second period in the boot sequence sequence

However, Rothman taught a fourth instruction set configured to scan a registry of the computer for pestware during the second period in the boot sequence sequence **(see Rothman, para. 0040, where the virus scanner, that's initialized during a pre-boot sequence, is operable to scan a registry)**.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the modified teaching of the combination of Zimmer and Rothwell with the teaching of Rothman because a user would have been motivated by methods of scanning input/output traffic of a computer system**(see Rothman, para. 0002)**

In regards to claim 8, The system of claim 7, the combination of Zimmer and Rothwell taught wherein the memory further comprises a fifth instruction set configured to launch, after an operating system is initiated, a pestware management engine**(see Zimmer, para. 0026, lines 5-7, where the secure virtual machine monitor is a pestware management engine that is operable after the boot phase of an operating system)**; and wherein the set of behavior rules includes behavior rules compiled by the pestware management engine**(see Zimmer, para. 0026, lines 11-14, where the secure virtual machine monitor establishes protection policies)**.

In regards to claim 11, the combination of Zimmer and Rothwell taught the system of claim 7, wherein the memory further comprises:
a ninth instruction set configured to manage pestware-related events after the computer becomes configured to run native applications**(see US 20050114687 Zimmer, para. 0026, lines 5-7, where the SVMM is a kernel level monitor that is operable in either a pre or post boot phase of an operating system)**.

In regards to claim 12, the combination of Zimmer and Rothwell taught the system of claim 7, wherein the first instruction set is further configured to monitor the boot sequence while boot drivers are initiated**(see Zimmer, para. 0026, lines 5-7, where the secure virtual machine monitor is initiated in the boot phase)**.

In regards to claim 13, Zimmer taught a non-transitory tangible computer-readable storage medium comprising program instructions for:
monitoring events during a boot sequence of the computer**(see US 20050114687, Zimmer, para. 0026, where the secure virtual machine monitor is used to monitor events during a boot process)**; managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occuring when the computer is configured to run native applications **(see US 20050114687, Zimmer, para. 0050, where protection policies may be established during a boot process)**;
Zimmer did not teach managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before

the computer becomes configured to run native applications, before a Win32 subsystem is loaded, and after a kernel is loaded

However, Rothwell taught managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a Win32 subsystem is loaded, and after a kernel is loaded **(see WO 2006077443, Rothwell, pg. 18, lines 30,31 and fig. 3, where the anti-malware is implemented during the boot sequence and wherein the anti-malware HDF filter operates in Kernel mode[fig. 3, item 153] to prevent the operation of malware from subsystems[e.g. Win32])**;

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teaching of Zimmer with the teaching of Rothwell because a user would have been motivated by methods of protecting a computer from malware**(see Rothwell, pg. 1, lines 1-5)**

and the combination of Zimmer and Rothwell did not teach scanning a registry of the computer for pestware during the second period in the boot sequence

However, Rothman taught scanning a registry of the computer for pestware during the second period in the boot sequence **(see Rothman, para. 0040, where the virus scanner, that's initialized during a pre-boot sequence, is operable to scan a registry)**.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the modified teaching of the combination of Zimmer and Rothwell with the teaching of Rothman because a user would have been motivated by

methods of scanning input/output traffic of a computer system**(see Rothman, para. 0002)**

In regards to claim 14, the combination of Zimmer, Rothwell, and Rothman taught the non-transitory tangible computer-readable storage medium of claim 13 further comprising program instructions for launching, after an operating system is initiated, a pestware management engine**(see Zimmer, para. 0026, lines 5-7, where the secure virtual machine monitor is a pestware management engine that is operable after the boot phase of an operating system)**; and
wherein the set of behavior rules includes behavior rules compiled by the pestware management engine**(see Zimmer, para. 0026, lines 11-14, where the secure virtual machine monitor establishes protection policies)**.

In regards to claim 17, the combination of Zimmer, Rothwell, and Rothman taught the non-transitory tangible computer-readable storage medium of claim 13 further comprising program instructions for:
managing pestware-related events after the computer becomes configured to run native applications**(see US 20050114687 Zimmer, para. 0026, lines 5-7, where the SVMM is a kernel level monitor that is operable in either a pre or post boot phase of an operating system)**.

In regards to claim 18, the combination of Zimmer, Rothwell, and Rothman taught The non-transitory tangible computer-readable storage medium of claim 13, wherein the monitoring includes monitoring the boot sequence while boot drivers are initiated**(see Zimmer, para. 0026, lines 5-7, where the secure virtual machine monitor is initiated in the boot phase)**.

2.) Claims 3, 9, and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 20050114687, Zimmer(filed on 11/21/2003) in view of WO 2006077443, Rothwell(filed on 1/18/2006) and further in view of US 20050216759, Rothman(filed on 3/29/2004) and further in view of US 20050125687 Townsend(filed on 12/5/2003)

In regards to claim 3, the combination of Zimmer, Rothwell, and Rothman taught the method of claim 1. The combination of Zimmer, Rothwell, and Rothman did not teach further comprising:

generating, in response to the monitoring, a record of events, the record of events including the pestware-related events; analyzing the record of events so as to identify the pestware-related events; and

modifying the set of behavior rules so as to prevent the pestware related events.

However, Townsend taught further comprising:

generating, in response to the monitoring, a record of events, the record of events including the pestware-related events**(see US 20050125687 Townsend, para. 0059, lines 1-24, where a log file is generated when a rule has been violated)**; analyzing

the record of events so as to identify the pestware-related events**(see Townsend, para. 0029, where the security engines collect pest-ware related events and forwards the information to the event manager for analysis)**; and

modifying the set of behavior rules so as to prevent the pestware related events**(see Townsend, para. 0006, where the replacement of an existent security policy with a new security policy is considered modifying a set of behavioral rules)**

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teaching of the combination of Zimmer, Rothwell, and Rothman with the teaching of Townsend because a user's computer system would benefit from the protective services provided against malicious software attacks. **(see Townsend, para. 0004)**

In regards to claim 9, the combination of Zimmer, Rothwell, and Rothman taught the system of claim 7. The combination of Zimmer, Rothwell, and Rothman did not teach wherein the memory further comprises:

a sixth instruction set configured to generate, in response to the monitoring, a record of events, the record of events including the pestware-related events;

a seventh instruction set configured to analyze the record of events so as to identify the pestware-related events; and

an eighth instruction set configured to modify the set of behavior rules so as to prevent the pestware related events.

However, Townsend taught wherein the memory further comprises:

a sixth instruction set configured to generate, in response to the monitoring, a record of events, the record of events including the pestware-related events**(see US 20050125687 Townsend, para. 0059, lines 1-24, where a log file is generated when a rule has been violated)**;

a seventh instruction set configured to analyze the record of events so as to identify the pestware-related events**(see Townsend, para. 0029, where the security engines collect pest-ware related events and forwards the information to the event manager for analysis)**; and

an eighth instruction set configured to modify the set of behavior rules so as to prevent the pestware related events**(see Townsend, para. 0006, where the replacement of an existent security policy with a new security policy is considered modifying a set of behavioral rules)**.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teaching of the combination of Zimmer, Rothwell, and Rothman with the teaching of Townsend because a user's computer system would benefit from the protective services provided against malicious software attacks. **(see Townsend, para. 0004)**

In regards to claim 15, the combination of Zimmer, Rothwell, and Rothman taught the non-transitory tangible computer-readable storage medium of claim 13. The combination of Zimmer, Rothwell, and Rothman did not teach further comprising

program instructions for:

generating, in response to the monitoring, a record of events, the record of events including the pestware-related events; analyzing the record of events so as to identify the pestware-related events; and modifying the set of behavior rules so as to prevent the pestware related events.

However, Townsend taught further comprising program instructions for: generating, in response to the monitoring, a record of events, the record of events including the pestware-related events**(see US 20050125687 Townsend, para. 0059, lines 1-24, where a log file is generated when a rule has been violated)**; analyzing the record of events so as to identify the pestware-related events**(see Townsend, para. 0029, where the security engines collect pest-ware related events and forwards the information to the event manager for analysis)**; and modifying the set of behavior rules so as to prevent the pestware related events**(see Townsend, para. 0006, where the replacement of an existent security policy with a new security policy is considered modifying a set of behavioral rules)**.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teaching of the combination of Zimmer, Rothwell, and Rothman with the teaching of Townsend because a user's computer system would benefit from the protective services provided against malicious software attacks. **(see Townsend, para. 0004)**

3.) Claims 4, 10, and 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 20050114687, Zimmer(filed on 11/21/2003) in view of WO 2006077443, Rothwell(filed on 1/18/2006) and further in view of US 20050216759, Rothman(filed on 3/29/2004) and further in view of US 20050125687 Townsend(filed on 12/5/2003) and further in view of US 20030135791, Natvig(filed on 9/25/2002)

In regards to claim 4, the combination of Zimmer, Rothwell, Rothman, and Townsend taught the method of claim 3. The combination of Zimmer, Rothwell, Rothman, and Townsend did not teach wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information information

However, Natvig taught wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information information **(see Natvig, para. 0025, where the logging of antivirus information is considered process identification information)**.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the modified teaching of the combination of Zimmer, Rothwell, and Rothman and Townsend with the teaching of Natvig because a user's computer system exposure to malicious code could be eliminated or substantially reduced**.(see Natvig, para. 0025)**.

In regards to claim 10, the combination of Zimmer, Rothwell, Rothman, and Townsend taught the system of claim 9. The combination of Zimmer, Rothwell, Rothman, and Townsend did not teach wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information

However, Natvig taught wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information **(see Natvig, para. 0025, where the logging of antivirus information is considered process identification information)**.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the modified teaching of the combination of Zimmer, Rothwell, and Rothman and Townsend with the teaching of Natvig because a user's computer system exposure to malicious code could be eliminated or substantially reduced**.(see Natvig, para. 0025)**.

In regards to claim 16, the combination of Zimmer, Rothwell, Rothman, and Townsend taught the non-transitory tangible computer-readable storage medium of claim 15. The combination of Zimmer, Rothwell, Rothman, and Townsend did not teach wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information

However, Natvig taught wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information **(see Natvig, para. 0025, where the logging of antivirus information is considered process identification information)**.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the modified teaching of the combination of Zimmer, Rothwell, and Rothman and Townsend with the teaching of Natvig because a user's computer system exposure to malicious code could be eliminated or substantially reduced**.(see Natvig, para. 0025)**.

CONCLUSION

Any inquiry concerning this communication or earlier communications from the examiner should be directed to GREGORY LANE whose telephone number is (571)270-7469. The examiner can normally be reached on 571 270 7469 from 8:00 AM to 6:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Taghi Arani, can be reached on 571 272 3787. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for

published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/GREGORY LANE/ Examiner, Art Unit 2438

/Taghi T. Arani/
Supervisory Patent Examiner, Art Unit 2438

Notice of References Cited	Application/Control No. 13/460,655	Applicant(s)/Patent Under Reexamination SCHNEIDER, JEROME L.	
	Examiner GREGORY LANE	Art Unit 2438	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2005/0114687	05-2005	Zimmer et al.	713/193
*	B	US-2005/0125687	06-2005	Townsend et al.	713/200
*	C	US-2005/0216759	09-2005	Rothman et al.	713/200
*	D	US-2003/0135791	07-2003	Natvig, Kurt	714/38
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N	WO2006077443	07-2006	GB	Rothwell	
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Trend Micro, Inc.
EX1002-182

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 July 2006 (27.07.2006)

PCT

(10) International Publication Number
WO 2006/077443 A2

(51) International Patent Classification:
G06F 21/00 (2006.01)

(21) International Application Number:
PCT/GB2006/050013

(22) International Filing Date: 18 January 2006 (18.01.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
05001120.4 20 January 2005 (20.01.2005) EP

(71) Applicant and
(72) Inventor: **ROTHWELL, William, Grant** [GB/CH]; Lehnmattstrasse 3, CH-4573 Lohn-ammannsegg (CH).

(74) Agent: **BREWER, Michael**; Marks & Clerk, 4220 Nash Court, Oxford Business Park South, Oxford, Oxfordshire OX4 2RU (GB).

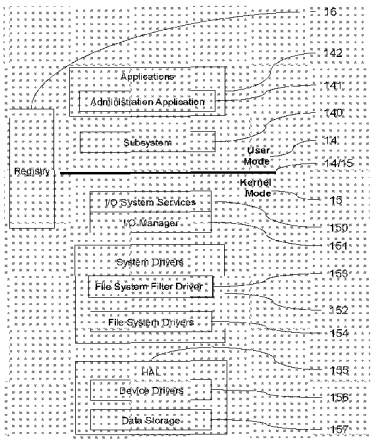
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: COMPUTER PROTECTION AGAINST MALWARE AFFECTION



(57) Abstract: A method is provided of protecting a computer (1) against malware affection. The computer (1) has a data storage (157, 12, 2) and an operating system for managing the data storage (157, 12, 2). The method comprises providing a filter module in the operating system which operates to detect an attempt to store data in the data storage, to determine a data format of the data to be stored in the data storage (157, 12, 2), and to prevent storage of the data if the data format is determined to relate to a predefined type. The filter module may be provided as a file system filter driver (153) in a kernel (15) of the operating system. The filter module may be arranged to operate between an input/output manager (151) of the operating system and a driver associated with the data storage (157, 12, 2). The input/output manager (151) and driver associated with the data storage may form part of the kernel (15) of the operating system.

WO 2006/077443 A2

Computer protection against malware affection

The present invention relates generally to computer protection against malware affection and more particularly to a computer program product and a method for protecting a computer against malware affection and with a computer protected against malware affection.

Malware is a short term used in the art for malicious software summarizing computer programs performing actions on computer systems without the consent of a user, often developed for the purpose of doing harm to the computer system or the user, and comprises for example so-called viruses, worms, Trojan horses, backdoors and spyware. The intended actions of malware widely varies and include stealing users' personal information, stealing other data, abusing the computer system or simply vandalize.

Usually a target computer is first attacked by malware in a warhead phase exploiting weaknesses of the computer systems, as for example producing a buffer overrun, for enabling a propagation phase. The warhead phase is executed by a very small program code portion of the malware. In the propagation phase the body program code of the malware is transferred to the target computer. Typically several phases follow the propagation phase by executing the body program code as for example a target selection phase, in which the malware is seeking for other computers accessible from the target computer, a scanning phase, in which accessible other computers are checked to be suitable target computers for further propagation, and a payload phase, in which the intended purposes of the malware are performed.

To prevent malware affecting a computer system several protections are known in the art as for example virus filters running on computers or firewalls controlling the traffic between connected computer systems. These protections recognize malware using malware descriptions which define specific characteristics of certain malware. They usually analyze data, check if the data shows some of the specific characteristics and take action if the data is identified as being malware. For example, the data is deleted or quarantined. A common technique for analyzing the data is pattern matching, wherein the data is scanned for predefined specific bit patterns defined to be malware bit patterns in the malware descriptions. The analysis usually is quite processing-intensive and can reduce the performance of the

computer system. The effectiveness of the above described protections is highly dependent on the up-to-dateness of the malware descriptions since malware is only recognized after the malware descriptions have been updated. Nevertheless, in the time period between the deployment of malware and the update of the malware descriptions in the computer system the computer system is vulnerable. It is therefore a delicate, difficult, labour-intensive and cost-intensive task to update the descriptions as fast as possible, which is usually done by specialized service providers.

Schmid et al "Protecting data from malicious software", Proc. 18th Annual Computer Security Applications Conference, 2002, 9-13 December 2002, Piscataway, NJ, USA, IEEE, describes an application called FileMonster®. The FileMonster® application uses a driver that works at the imaginary boundary between the user mode and protected kernel space; it intercepts at the point where user mode function calls are translated into kernel API calls, i.e. using service dispatch tables mapping. This is an undocumented approach.

The W32/Swen.A@mm worm that appeared in September 2003 is described in Frisk Software International: "W32/Swen.A@mm" (retrievable from the Internet at http://web.archive.org/web/20040411010214/http://www.f-prot.com/virusinfo/print/descriptions/swena.html). As a way to persist on the victim computer, the worm made auto-start entries in the Registry so that the system would load the worm's executable code on boot up. A random name was used for the worm's image file to avoid detection, i.e. <random_characters>.exe.

Winability Software Corp's "Folder Guard User's Guide – Using the filters" (retrievable from the Internet at: http://web.archive.org/ web/20041023085327/www.winability.com/folderguard/users-guide_filters.htm; see also http://www.winability.com/folderguard/users-guide-printable.htm) describes a system that stops other users from opening or seeing a user's personal files; it can also protect sensitive system files from modification or destruction. Folder Guard allows and disallows program execution using a white list (explicitly allowing certain applications to run) and a black list (explicitly disallowing certain applications to run). This implies the program files already exist on the computer, and Folder Guard performs check to allow or disallow when the program is able to run.

The McAfee® Security "Product Guide – VirusScan Enterprise version 7.0" (7 March 2003) document describes a virus scanning, detection and removal program that can use of file extensions to allow a user to configure the program scanning behaviours. Based on the user's selection, the program can scan the file on 'read' access, 'write access' or 'on network'. The application uses conventional scanning and pattern matching techniques to detect viruses. Similarly, the "Sophos® Antivirus – Windows NT/2000/XP single user installation guide" describes an application that scans a file for virus detection on a read access, or on write and other accesses.

EP-A-1,429,247 (Microsoft Corporation) relates to a new approach as to how a file system filter driver should be implemented.

In view of the disadvantages of the hitherto-known, above described protections against malware affection, it is desirable to provide an improved, effective, fast and easy to maintain computer protection against malware.

According to a first aspect of the present invention, there is provided a method of protecting a computer against malware affection, the computer having a data storage and an operating system for managing the data storage, the method comprising providing a filter module in the operating system which operates to detect an attempt to store data in the data storage, to determine a data format of the data to be stored in the data storage, and to prevent storage of the data if the data format is determined to relate to a predefined type.

The filter module may be provided as a file system filter driver in a kernel of the operating system.

The filter module may be arranged to operate between an input/output (I/O) manager of the operating system and a driver associated with the data storage.

The input/output (I/O) manager may form part of a kernel of the operating system.

The driver associated with the data storage may form part of a kernel of the operating system.

The method may comprise the file system filter driver receiving at least write access calls from the input/output (I/O) manager intended for the kernel driver associated with the data storage, blocking such write access calls relating to data of the format of the predefined type, and such passing write access calls not so relating.

The method may comprise informing the input/output (I/O) manager about the blocking and optionally passing of such write access calls.

The method may comprise arranging for I/O Request Packets that are sent from the input/output (I/O) manager and intended for the driver associated with the storage to be re-routed via the filter module, the re-routed I/O Request Packets being at least those of a type enabling the filter module to detect an attempt to store data in the data storage.

The method may comprise querying a function code of a re-routed I/O Request Packet at the filter module to detect an attempt to store data in the data storage

The re-routed I/O Request Packets may comprise those relating to the creation of a new file in the data storage.

The re-routed I/O Request Packets may comprise those having or associated with an IRP_MJ_CREATE function code.

The re-routed I/O Request Packets may comprise those relating to the writing to an existing file on the storage. The existing file may be a recently-created file.

The re-routed I/O Request Packets may comprise those having or associated with an IRP_MJ_WRITE function code.

The method may comprise blocking a re-routed I/O Request Packet at the filter module if it is determined that the format of the data to be stored by way of that I/O Request packet is of the predefined type.

The method may comprise determining the format of the data to be stored by examining the intended file extension of the file to be created.

The method may comprise determining the format of the data to be stored by examining a predetermined portion of the file to be written, for example its file header. The file header may be the Portable Executable File Format (PE) file header.

The method may comprise referring to at least one parameter associated with the filter module preferably stored in a protected manner in the data storage, and determining the behaviour of the filter module at least in part in dependence on the at least one parameter.

The method may comprise running a protected administration module which allows a verified user to adapt the at least one parameter.

The method may comprise storing the at least one parameter in a protected part of a system database of the operating system. The system database may comprise the Registry.

The method may comprise reading the at least one parameter during the booting of the computer.

The at least one parameter may comprise a predefined blocking file extension as the predefined type, and comprising the filter module detecting the format of the data to be stored by comparing a specific file extension of the data to the predefined blocking file extension.

The at least one parameter may comprise a flag, wherein the flag is able to be set to a predefined status, and comprising the filter module checking the flag and preventing storage of the data if the flag is set to the predefined status.

The at least one parameter may comprise a non-block user identification, and comprising the filter module checking the non-block user identification as well as an active user identification and allowing the storage of the data if the non-block user identification matches the active user identification.

The method may comprise providing an administration application to allow a user to alter the behaviour of the filter module by sending the filter module runtime instructions.

The method may comprise using the administration application to turn off blocking temporarily.

The administration application may be operable to alter the behaviour of the filter module by changing, or causing the filter module to change, at least one of the at least one parameter.

The method may comprise authenticating the administration application to the filter module.

The method may comprise the filter module storing a name or identifier of the filter module in a protected manner in the data storage in such a way that only the administration application can reasonably learn of the name or identifier and thereby communicate with the filter module.

The method may comprise registering the name in the operating system.

The name or identifier of the filter module may be stored in a protected manner in the data storage by changing it from time to time during runtime.

The method may comprise authenticating the administration application to the filter module using a cryptographic authentication method.

The method may comprise logging activities of the filter module.

The predefined type of data may comprise executable data.

The filter module may operate to prevent storage of such data to any part of the storage.

The operating system may be one of the Windows® family of operating systems.

According to a second aspect of the present invention, there is provided method of thwarting malware at its propagation phase by performing a method as claimed in any preceding claim so as to block any unauthorised attempt to write any data to the data storage that could potentially constitute malware, thereby preventing the malware from propagating.

According to a third aspect of the present invention, there is provided method of administrating a computer having a kernel driver and an administration application that communicates with the kernel driver during runtime, comprising authenticating the administration application to the kernel driver.

The method may comprise using a cryptographic method to authenticate the administration application to the kernel driver.

According to a fourth aspect of the present invention, there is provided a method of authenticating an administration application to a kernel driver for use in a method according to the third aspect of the present invention, comprising using a dynamic driver identifier that is made to change during runtime in a way such that only the administration application can reasonably learn of the current driver identifier and thereby communicate with the kernel driver.

The kernel driver may be a file system filter driver.

According to a fifth aspect of the present invention, there is provided computer comprising a data storage, an operating system for managing the data storage, and a filter module provided in the operating system for protecting the computer against malware affection, the filter module being adapted to detect an attempt to store data in the data storage, to determine a data format of the data to be stored in the data storage, and to prevent storage of the data if the data format is determined to relate to a predefined type.

According to a sixth aspect of the present invention, there is provided computer program which, when run on a computer, causes the computer to carry out a method according to the first aspect of the present invention.

According to a seventh aspect of the present invention, there is provided a computer program which, when loaded into a computer, causes the computer to become one according to the fifth aspect of the present invention.

The computer program may be carried on a carrier medium. The carrier medium may be a transmission medium. The carrier medium may be a storage medium.

One embodiment of the present invention provides a computer program product for protecting a computer against malware affection, wherein the computer has a data storage and an operating system managing the data storage. The computer program product comprises computer readable program code means for causing the computer to run a filter module. The computer readable program code means cause the computer to run the filter module in the operating system, and the filter module is configured to detect a format of data to be stored or altered in the data storage. The filter module prevents storage or alteration of the data if the format is of a predefined type. By predefining the types of format of malware data, which typically is any format executable by the operating system, the filter module interferes the malware life cycle at the critical point of its propagation phase by preventing the storage of the malware in the data storage. Without propagation the malware is not able to execute its payload and is effectively terminated. The detection of the format can easily happen without any prior knowledge about current malware and without further data analysis such as scanning the data and searching for specific bit patterns. The running filter module protects the computer in an efficient and fast way, e.g. as a pure software implementation, against known and future malware using the predefined data format types, with no need of updating any malware descriptions and irrespective to operating system privileges of caller processes and threads. It also protects the computer against malware using common mechanisms to bypass its detection such as polymorphism and renaming files. Because the computer runs the filter module in the operating system, the protection against malware is effective on all current data storage devices accessible by the computer as well as all future data storage devices accessible by the computer without requiring any adaptation related to the malware protection.

In a preferred embodiment the program code means are adapted to cause the computer to run the filter module as a file system filter driver in a kernel of the operating system. This assures

that all read/write access requests originated from any user mode application or from any system process, which are directed to any data storage device associated to the computer, are treated by the filter module. The filter module running in the kernel does not have any additional impact on the behaviour or the performance of the computer. Since the kernel of the operating system is launched at a very early stage of the booting process of the computer, the malware protection is effective from scratch and especially before system processes are running. Furthermore, running the filter module in the kernel allows a malware protection irrespective of users' operating system access rights or the operating system access rights of a user simulated by the malware.

Preferably the program code means are adapted to cause the computer to run the file system filter driver configured to receive read and write access calls from an input/output manager of the kernel and further configured to block write access calls related to data of the format of the predefined type. Further the program code means are adapted to run the file system filter driver configured to inform the input/output manager about the blocking and finally to transfer non-blocked read and write access calls to further system drivers of the kernel. Because the file system filter driver receives the read and write access calls from the input/output manager and not directly from an application running in a user mode of the computer, the filter driver is independent of service dispatch tables matching user mode access calls to kernel mode access calls. These service dispatch tables usually are not documented and not supported by the operating system manufacturer and may change in future versions of the operating system. By transferring non-blocked read and write access calls to further system drivers, i.e. file system drivers as for example NTFS (New Technology File System - one of the file systems for the Windows NT (New Technology) operating system by Microsoft Corporation), FAT (File Allocation Table - a file system for various operating systems of Microsoft Corporation) or CDFS (CD ROM (Compact Disk Read Only Memory) File System - a driver for CD ROM players used in certain operating systems of Microsoft Corporation), and not directly to hardware drivers of storage media, the filter module does not need to consist of different classes of filter drivers, each fitting to a storage medium present on the computer, for example a local or a remote hard disk, a backup tape, a memory stick or an optical read/write device.

In a preferred embodiment the program code means are adapted to cause the computer to store a parameter of the filter module in a protected manner in the data storage, to run the filter module adjusted by the parameter, and to run a protected administration module configured to allow a verified user to adapt the parameter. Using parameters for the adjustment of the filter module is an easy way to be able to adapt the malware protection to the changing needs during life time of the malware protection even while it is running. The administration module can be used to adapt the parameter and to trigger the adjustment of the filter module using the changed parameter. It is typically run as a protected user application.

Preferably the program code means are adapted to cause the computer to store the parameter in a protected part of a system database of the operating system, further to cause the computer to read the parameter while booting the computer, and finally to cause the computer to run the filter module adjusted by the parameter. System databases, such as for example the Registry of the Windows® operating systems by Microsoft Corporation, are commonly used to store configuration information of a computer system comprising hardware information, user information, application information, and other information. It can easily be used to store the parameters for adjusting the filter module. In particular while booting the computer the parameters can be read and the filter module can be adjusted. This assures that the malware protection is always customized to the needs of the user and does not run in another configuration.

In a preferred embodiment the program code means are adapted to cause the computer to run the filter module in a way to store a name of the filter module in a protected manner in the data storage and in a way to register the name in the operating system. Such a filter module is able to change its name from time to time or on request, e.g. triggered by a user operating the administration module, and to register the changed name, i.e. the file system filter driver name, in the operating system and to store the name, particularly as a parameter, in a protected part of the system database, for making it accessible to the registration of the filter module during the boot process of the computer. Using such a dynamic name of the filter module allows the filter module to be undetectable and unreferable for programs (including malware) other than the administration module, which knows how and where to look up the current name of the filter module.

The program code means can be adapted to cause the computer to store a predefined blocking file extension as the predefined type of data format in a protected manner in the data storage and to run the filter module in a way to detect the format of the data to be stored or to be altered by comparing a specific file extension of the data to the predefined blocking file extension. The extension of a data file is a common way to define the data format of a data file and consists for example of a dot and few letters attached to the name of the data file, i.e. Windows operating systems by Microsoft Corporation use in general a dot and three letters following the file name to specify the data format e.g. ".doc", ".xls" or ".txt". The file extension can be used as an indication if a data file is potentially executable by the operating system and therefore be potential malware. A list of such potential malware file extensions can be stored in the data storage, in particular as parameters in a protected part of the system database, in order to be compared to the file extension of the data to be stored or to be altered in the data storage. For Windows operating systems by Microsoft Corporation This can for example be a list comprising the extensions ".COM", ".EXE", ".DLL", ".BAT", ".CMD", ".OCX", ".VBS", ".PIF", ".SCR", ".CHM", ".DRV", ".HTA", ".SHS", ".WMA", ".WSF", ".WSH", ".CPL", ".CAB".

Preferably the program code means are adapted to cause the computer to store a flag in a protected manner in the data storage, wherein the flag is able to be set to a predefined status. The filter module is configured to check the flag, and is further configured to detect the format of the data to be stored or to be altered as well as to prevent the storage or alteration of the data if the flag is set to the predefined status. Such a flag, in particular stored as a parameter in a protected part of the system database, can be used to switch the filter module on or off during the boot process of the computer, or triggered by the administration module. For example, there can be the need to switch the filter module off to be able to store data files of a blocked predefined data format in the data storage, e.g. during installation of new software on the computer.

The program code means can be adapted to cause the computer to store a non-block user identification in a protected manner in the data storage, and to run the filter module in a manner to check the non-block user identification as well as an active user identification, and to intercept the prevention of the storage or alteration of the data if the non-block user identification matches the active user identification. The storage of such non-block user

identifications, in particular as parameters in a protected part of the system database, can be useful to ease the administration of a computer on which often data of the blocked predefined format is regularly saved, e.g. by installation of software. A user who is identified as a non-block user bypasses the filter module in a secure controlled way.

Preferably the program code means are adapted to cause the computer to log activities of the filter module. The log of the activities of the filter module can e.g. be used to analyze the effectiveness of the malware protection, to find unwanted data file filtration, or just for statistical reasons.

Another embodiment of the present invention relates to a computer protected against malware affection having a data storage and an operating system managing the data storage. The computer is configured to run a filter module in the operating system, wherein the filter module is configured to detect a format of data to be stored or altered in the data storage and to prevent storage or alteration of the data if the format is of a predefined type.

The advantages of the computer protected against malware affection and of its preferred embodiments are the same as the advantages of the corresponding computer program product and the corresponding preferred embodiments of the computer program product. They are therefore not explicitly repeated and can be found above.

In a preferred embodiment the computer is adapted to run the filter module as a file system filter driver in a kernel of the operating system.

Preferably the file system filter driver is configured to receive read and write access calls from an input/output manager of the kernel. Further, it is configured to block write access calls related to data of the format of the predefined type, to inform the input/output manager about the blocking and to transfer non-blocked read and write access calls to further system drivers of the kernel.

In a preferred embodiment the computer is adapted to store a parameter of the filter module in a protected manner in the data storage. The filter module is configured to run adjusted by the

parameter, and the computer is adapted to run a protected administration module allowing a verified user to adapt the parameter.

Preferably the computer is adapted to store the parameter in a protected part of a system database of the operating system, further to read the parameter while booting, and finally to run the filter module adjusted by the parameter.

In a preferred embodiment the filter module is configured to store a name of the filter module in a protected manner in the data storage and to register the name in the operating system.

The computer can be adapted to store a predefined blocking file extension as the predefined type in a protected manner in the data storage, wherein the filter module is configured to detect the format of the data to be stored or to be altered by comparing a specific file extension of the data to the predefined blocking file extension.

Preferably the computer is adapted to store a flag in a protected manner in the data storage. The flag is able to be set to a predefined status, and the filter module is configured to check the flag and to detect the format of the data to be stored or to be altered as well as to prevent the storage or alteration of the data if the flag is set to the predefined status.

The computer can be adapted to store a non-block user identification in a protected manner in the data storage. The filter module is configured to check the non-block user identification as well as an active user identification and to intercept the prevention of the storage or of the alteration of the data if the non-block user identification matches the active user identification.

Preferably the computer is adapted to log activities of the filter module.

Another embodiment of the present invention relates to a method for protecting a computer against malware affection, the computer having a data storage and an operating system managing the data storage. The method comprises the steps of running a filter module in the operating system, with the filter module detecting a data format of data to be stored or altered in the data storage and preventing the storage or alteration of the data if the format is of a predefined type.

The advantages of the method and of its preferred embodiments are the same as the advantages of the corresponding computer program product and the corresponding preferred embodiments of the computer program product. They are therefore not explicitly repeated and can be found above.

In a preferred embodiment the filter module is run as a file system filter driver in a kernel of the operating system.

Preferably the method comprises the steps of the file system filter driver receiving read and write access calls from an input/output manager of the kernel, blocking write access calls related to data of the format of the predefined type, informing the input/output manager about the blocking and transferring non-blocked read and write access calls to further system drivers of the kernel.

In a preferred embodiment the method comprises the steps of storing a parameter of the filter module in a protected manner in the data storage, adjusting the filter module by the parameter, and running a protected administration module which allows a verified user to adapt the parameter.

Preferably the method comprises the steps of storing the parameter in a protected part of a system database of the operating system, reading the parameter while booting the computer and adjusting the filter module using the parameter.

In a preferred embodiment the method comprises the steps of the filter module storing a name of the filter module in a protected manner in the data storage and registering the name in the operating system.

The method can comprise the steps of storing a predefined blocking file extension as the predefined type in a protected manner in the data storage and detecting the format of the data to be stored or to be altered by comparing a specific file extension of the data to the predefined blocking file extension by the filter module.

Preferably the method comprises the steps of storing a flag in a protected manner in the data storage, wherein the flag is able to be set to predefined status, the filter module checking the flag, detecting the format of the data to be stored or to be altered, and preventing the storage or alteration of the data if the flag is set to the predefined status.

The method can comprise the steps of storing a non-block user identification in a protected manner in the data storage, the filter module checking the non-block user identification as well as an active user identification and the filter module intercepting the prevention of the storage or alteration of the data if the non-block user identification matches the active user identification.

Preferably the method comprises the step of logging activities of the filter module.

Reference will now be made, by way of example, to the accompanying drawings, in which:

Figure 1 shows a block diagram schematically illustrating components of a computer system which are of relevance for a protection against malware affection according to the invention;

Figure 2 shows a block diagram illustrating the architecture of an implementation of the inventive computer protection against malware affection;

Figure 3 is a block diagram illustrating operation of an embodiment of the present invention to block the writing of a certain type of file while allowing the writing of another type;

Figure 4 is an diagram for illustrating in more detail the processing of an I/O request in the kernel within the context of the present invention; and

Figure 5 shows a flow diagram of a program logic flow of an inventive computer protection against malware affection.

Before proceeding to a description of a specific embodiment of the present invention, an overview will first be provided. An embodiment of the present invention is founded on the basis that:

- All running processes on a computer are executable code, including malware;
- Executable code must be accessible on a computer to load and run, i.e. stored on disk; and therefore
- Malware cannot run or spread to other computers without execution code on a computer.

As mentioned above, the typical infection cycle of malware is (1) to cause its executable code to be stored on a computer using various means usually not in a user control or knowledge, (2) execute its code and (3) continue to spread its code to other computers, i.e. infection spreading. An embodiment of the present invention breaks malware infection by denying malware the ability to write its execution code to computers, which is critical in the malware life cycle. Malware typically exists in computer systems as a runnable program which is identified by its file extension. A few samples of executable file extensions on Microsoft® Windows® platforms are *.exe, *.dll, *.ocx *.vbs, and there are others. Robustly blocking these program files from being saved to a computer effectively stops malware infection.

The following uses a worm attack life cycle as an example to illustrate this in more detail.

1. A typical attack begins with what virus researchers called "warhead attack". A warhead is very small part of the malware code that exploits identified system vulnerabilities, e.g. buffer overrun attacks, to enable it to run on a computer. The warhead code is necessarily small with very limited functions; the infamous SQL Slammer has a warhead of 376 bytes.

2. The next step is to transfer the worm execution body code to the victim computer, i.e. saving a copy of the malware program file, and execute the code. This is known as the malware's "propagation phase". If successful, the malware seeks out other victims and continues its infection spread.

3. If a worm is not stopped, the rest of its cycle can be categorised as "target selection", where it locates other networked computers, "vulnerability scanning" to

identify other vulnerable computers, and "payload attack" when it performs the intended malicious actions on the victim computer.

An embodiment of the present invention intercepts malware's life cycle at the critical point of propagation, i.e. saving its execution code in a program file to permanent storage. Without propagation, the malware cannot run its payload and is effectively terminated. Under the Windows® operating system, all file saving operations require system kernel services and no programs, including malware, can write to hardware directly, because of Windows® operating system's design and internal protection.

An embodiment of the present invention proactively protects computer from malware infection without using conventional methods. In one mode of operating an embodiment of the present invention, all executable files are blocked from being written to a computer that have not had the user's explicit approval. The term "malware" is used to describe virus, worm, Trojan, spyware, adware, rootkits and all unwanted software applications.

On Microsoft® Windows® platforms, all application programs require its executable code (image file) to reside on the hard disk (storage media) in order that the system can load the image file to memory and run. From the operating system perspective, malware is just another form of application program, which must have its execution code residing on the system disk to carry out its 'unwanted action' from the user's perspective. Malware spreads infection by transferring its execution code to a user computer by tricks and exploiting vulnerabilities on the system, e.g. buffer overrun attacks or email attachment.

Without execution code on a computer, malware will not be able to infect a computer nor spread to other computers. An embodiment of the present invention indiscriminately denies writing of executable files to hardware storage, at the file system level such as NTFS and FAT. An embodiment of the present invention scrutinizes kernel instructions (rather than API calls) to the file system to write files. It has a deny-write-by-default policy, and a feature is built in to allow writing of executables if a user explicitly approves them. Such a feature is necessary for the user to install new software and patching system files.

An embodiment of the present invention differs from anti-virus, anti-spyware, rootkit detection tools in protection methods in the following respects:

- No pattern scanning – Anti-virus and anti-spyware products rely on scanning existing and incoming files against known malware signatures to detect virus/malware, e.g. McAfee®, Sophos®, and other major anti-virus products use this approach. Shortcomings are that this only works against known malware in the pattern database, and also degraded system performance.
- No behaviour anomaly detection – Used by most anti-spyware tools and some newer anti-virus products. They monitor and intercept 'unusual' API calls, and 'unexpected' application behaviours, and flagged detection when anomaly is found. False-positive ratio is high and it is relatively easy to bypass by malware.
- No system calls interception – Some tools intercept and manipulates system calls. They route API calls to their interceptor functions and manipulate the call by changing parameters or returning an execution failure status to the caller and not calling the requested API call, e.g. the FileMonster® application [see reference above and http://www.cigital.com/labs/projects/1021/]. This is not a Microsoft® approved approach.
- No system memory scanning – Some solutions scan system memory similar to file-based scan and monitor memory access trying to detect unusual usage, e.g. buffer overrun detection. This is not proven effective against malware and is not effective to "zero day" malware variants.

Another difference between an embodiment of the present invention and existing anti-malware technology is that an embodiment of the present invention is a malware prevention technology and not a detection and clean tool like typical anti-virus and anti-spyware software. Another technical difference, which will become more readily understandable following the description below, is that the filter driver used in an embodiment of the present invention cannot be shut down by malicious software, as seen in many anti-malware applications. An embodiment of the present invention is integrated into the file system and loaded automatically during system boot up sequence.

Interception in an embodiment of the present invention is possible by taking advantage of Windows self-protection architecture design. Briefly, Windows® enforces privilege-checking mechanisms to guarantee system integrity, and the concept of kernel mode (Ring 0), and user mode (Ring 3) on Intel x86 CPU. The operating system (OS) prevents direct access to all hardware devices, such as a hard disk, by a thin layer of system software called the Hardware Abstraction Layer (HAL). When Windows® is running, all 'write' accesses are provided by Kernel service calls. By integrating into the kernel, an embodiment of the present invention scrutinizes what files are allowed to be saved. This prevents email-borne, Internet-borne and user-induced virus files from reaching the protected computer.

Figure 1 provides an overview of components of a computer system that are of relevance for protection against malware affection according to an embodiment of the invention. A computer 1, e.g. a personal computer, a server, a personal digital assistant (PDA) or a cellular phone, has a CPU 10 (Central Processing Unit) and RAM 11 (Random Access Memory - a type of volatile computer memory that can be accessed randomly, meaning any byte of memory can be accessed without touching the preceding bytes) which are involved in all processing of the computer 1. It further has a data storage 157 (see Figure 2) comprising an internal data storage 12 and an external data storage 2 and RW Devices 13 (Read Write Devices). The internal data storage 12 usually comprises at least a hard disk on which data can be stored, changed and deleted. The external data storage 2 comprises on one hand removable data media such as for example floppy disks, CD-ROM (Compact Disks-Read-Only Memory - a type of optical disk), DVD (Digital Versatile Disks – another type of optical disk) memory token as e.g. memory sticks, or multi media devices, e.g. cameras or music players usually containing removable data storage media or hard disks themselves. On the other hand the external data storage 2 comprises remote data storage media such as for example shared network drives. The RW Devices 13 are capable of reading and/or writing to the external data storage 2 and summarize for example on one hand floppy disk drives, CD-ROM drives, DVD drives and ports for the multimedia devices or the memory token such as e.g. USB (Universal Serial Bus) or Firewire, for the use of the removable data storage media, and on the other hand network ports for the use of the remote data storage media. Connected to the computer 1 is a network LAN/WAN 3 (Local Area Network / Wide Area Network as e.g. the Internet) by a link 4 which can be either a wired link or a wireless link. The LAN/WAN 3 is usually used by malware to deploy and to get access to computer 1.

Typically the computer 1 is running an OS (Operating System – e.g. Microsoft® Windows®, UNIX, OS/2 or Linux) which is the system program responsible for the direct control and management of hardware including the data storage 157 (see Figure 2) and basic system operations, as well as running application software. The central module of the OS loaded first while booting the computer 1 and remaining in the RAM 11 usually is called kernel. Typically the kernel is responsible for memory management, process and task management, user interfacing, and disk management. Because it stays in RAM 11, it is important for the kernel to be as small as possible while still providing all the essential services required by other parts of the operating system and by application software.

Figure 2 illustrates one preferable architecture of an implementation of the inventive computer protection technique against malware affection in the computer 1. The implementation is separated by a boundary 14/15 in a upper user mode 14 and a lower kernel mode 15. The user mode 14 comprises a user application layer 142 which illustrates application programs accessible by a user. One user application in the user application layer 142 is a protected administration application 141 allowing a verified user to adjust and to administer the computer protection against malware affection. Between the kernel mode 15 and the user application layer 142 a subsystem layer 140 is arranged. In the subsystem layer 140 a protected subsystem such as e.g. Win32 for Windows® by Microsoft® Corporation or POSIX for UNIX and Linux is running providing interfaces to the user application layer 142, i.e. routines, protocols, and tools for building and running applications, and hiding the kernel mode layers to the user of the computer 1. The subsystem layer 140 passes input/output requests of the user application layer 142 to a I/O system service layer 150 (Input / Output) in the kernel mode 15 containing I/O system services. The I/O system services are provided by a I/O manager layer 151 lying under the I/O system service layer 150 which prepares input/output request packets corresponding to the input/output requests received by the subsystem layer 140 and which passes the input/output request packets to a file system filter driver 153 of a system driver layer 152. The system driver layer 152 comprises various registered system drivers containing said file system filter driver 153 as an implementation of a filter module and the file system drivers 154 such as e.g. NTFS, FAT or CDFS drivers. The file system filter driver 153 detects a format of data affected by the input/output request packets, blocks input/output request packets for data of a predefined format and passes

input/output request packets for data other than of the predefined format to the file system drivers 154. Under the system driver layer 152 the HAL 155 (Hardware Abstraction Layer) is situated comprising device drivers 156 which control the data storage 157. The file system drivers 154 pass the input/output request packets to the HAL 155 in which the storage of the data is completed. Besides the user mode 14 and besides the kernel mode 15 lies the registry 16 as a system database used to store configuration information in all layers described above.

For an embodiment of the present invention that relies on Windows® kernel drivers, and in particular file system filter drivers, an overview of these issues, and the concept of I/O services in the kernel in general, will now be provided with reference to Figures 1 and 2. Such a description will aid in an understanding of a Windows®-based embodiment of the present invention, and how it differs from the prior art, but it will be appreciated that the present invention is not limited to a Windows®-based environment (for example, on modern operating systems, e.g. Microsoft® Windows® and Unix®, software runs either in user mode, such as Outlook, or in kernel mode, such as services like network drivers and TCP/IP stacks).

An embodiment of the present invention will sometimes be referred to herein by the term "HDF", which stands for Hardware Disk Firewall, a function that is effectively performed by an embodiment of the present invention.

The term device driver refers to the software that controls the operation of a particular class of hardware devices, e.g. hard disk or tape drive. There are many classes of drivers, e.g. video drivers and network drivers [see http://www.microsoft.com/whdc/driver/foundation/DrvRoadmap.mspx]. The device drivers of interest in context of an embodiment of the present invention are file system drivers such as NTFS and FAT. User mode Win32 application programs do not run in the kernel space.

The Windows® architecture does not allow applications direct access to hardware, using the software protection layer called 'Hardware Abstraction Layer' (HAL), and it defines device-independent interfaces for applications to I/O operations on hardware devices. This abstraction enables applications to request I/O operations without regard to the physical characteristics of the underlying hardware devices.

Typical I/O operations on a hardware storage device are read and write requests. The kernel maintains a data structure called "services dispatch" tables. The tables map user mode function calls to Windows native APIs (Application Programming Interface), which are not fully documented. When an application makes a function call, the user mode function call (API) is mapped to the appropriate kernel native API call(s) and serviced by the kernel. For example, a user mode call to CreateFile(...) is mapped to kernel function call ZwCreateFile(...).

Some applications, for example the FileMonster® application, manipulate these undocumented dispatch tables ["Undocumented Windows 2000 Secrets: A Programmer's Cookbook" by Sven Schreiber, ISBN: 0201721872] and modify the entries to intercept system calls. Since Microsoft® has not documented nor recommended this approach, it is not considered good practice for system stability. For example, it two applications modify the dispatch tables, the system may not work correctly.

Windows provides I/O services by the I/O system services and I/O Manager kernel components. When an I/O request is received from the I/O system services component, the I/O Manager creates a data structure called an I/O Request Packet (IRP). The IRP is used to package and to dispatch the request and control messages to the target device driver via function codes. The function code is embedded in an IRP that a driver receives and acts on.

A driver implements and services function codes such as IRP_MJ_CREATE, IRP_MJ_CLOSE and IRP_MJ_DEVICE_CONTROL etc. and reports back status on completion. For example, when a Win32 application makes a CreateFile(...) call, the I/O Manager calls the target driver's IRP_MJ_CREATE function code. Later, when the application calls ReadFile(...) or WriteFile(...), the I/O Manager dispatches IRP_MJ_READ or IRP_MJ_WRITE calls in the IRP. Likewise, when the application closes the file handle, IRP_MJ_CLOSE is called. The I/O Manager also returns an I/O request result to the caller, e.g. succeeded or failed.

Windows has a layered-drivers architecture that allows other kernel drivers to intercept IRPs destined to a target device driver. The intended purpose is to provide additional services; these intercepting drivers are called file system filter drivers, or just filter drivers. The filter

driver used in an embodiment of the present invention will referred to herein as the "HDF" (filter) driver, or Hardware Disk Firewall filter driver, and is a file system filter driver that intercepts I/O requests to the file systems, e.g. NTFS.

In its initialization routine DriverEntry(...), the HDF driver informs the I/O Manager the I/O requests it wants to intercept, by registering the necessary I/O function codes, and calls IoAttachDeviceByPointer(...). The I/O Manager then dispatches IRPs, that are destined to the target drivers, first to the HDF filter driver for processing. HDF can request to receive only IRPs of interest but not all IRPs as explained here. The I/O Manager, on loading a driver, calls the "entry point" *DriverEntry()*,and passes in a pointer of a data structure (the driver's Device Object) together with a text string of its Registry path, which is not hard-coded so that HDF knows where to find its registry parameters. HDF performs initialization in *DriverEntry()* and fills out some, although not necessarily all, of the Device Object structure's data members before returning to the I/O Manager. One data member is an array of IRPs of interest, for example "*DriverObject->MajorFunction[IRP_MJ_CREATE] = HDFCreate;*". In this case, *HDFCreate()* is called when there is a IRP_MJ_CREATE; one or more of IRP_MJ_WRITE, IRP_MJ_READ, IRP_MJ_FILE_SYSTEM_CONTROL and IRP_MJ_DEVICE_CONTROL are also registered (it is useful to intercept _READ to protect the HDF application image files themselves from unauthorized access so that other processes are prevented from opening application files relating to HDF. The read and write IRPs can be used for the purpose of blocking executables with unconventional extensions as set out elsewhere in this description).

The HDF driver intercepts I/O requests to create a new file (IRP_MJ_CREATE). The interception allows the HDF driver to inspect the file extension of the impending file create operation without the need to scan the file data. To deny the creation of executable files, the HDF driver blocks all write requests of execution code on the system. This is achieved by returning a status 'STATUS_ACCESS_DENIED' to the I/O Manager, which then fails the request and informs the caller application, e.g. Internet Explorer.

This is illustrated in Figure 3, which shows the signal flow for two different attempted save operations: (a) an attempt to save a potentially harmful executable file "keylog.exe" to the storage medium 157; and (b) an attempt to save a harmless data file "business.doc" to the

storage medium 157. As illustrated, the HDF filter driver 153 blocks the write attempt for "keylog.exe", but not that for "business.doc".

(It should be noted that checking file-extension is a common approach to identify file types on Windows platforms. On a Unix-based computer, executable files are identified by file attributes, for example. The present invention is not to be understood as limited to any particular method of determining the file type on any particular operating system.)

The protection of existing executable files is a side-effect of using a technique embodying the present invention, for the following reasons. For a malware to modify an existing executable file, the creation of a new file is still required, since the modified file is written back to the storage as a new file. Therefore, an attempt to modify an existing executable file on the storage will be caught by checking for IRP_MJ_CREATE as set out above. There does not need to be a check as to whether a completely new file is being created and written, or whether it is an existing file that is being modified and then written back as a new file.

For any data that is written to storage, there is always an IRP_MJ_CREATE (to create a new file, whether it be for completely new data or for data modified from an existing file) followed by an IRP_MJ_WRITE (to actually write the data to the storage), arriving in different IRPs. If the file extension of a file to be created or modified is recognised as an executable, then the IRP_MJ_CREATE check will be sufficient to catch the malware at the time of requesting a new file, and this will block any attempt to create or modify the file.

However, in an attack scenario where malware tries to create or modify an executable file without an expected file extension, e.g. keylog.abc, then the file will pass the above IRP_MJ_CREATE check. To handle such a scenario, the HDF driver can inspect the impending file's data for the PE file header (for which, see below). This is desirable because the HDF driver would otherwise not deny the writing or modification of 'keylog.abc' because it has a non-executable file extension. For this purpose, the HDF driver can also intercept the write data request (IRP_MJ_WRITE). If the I/O request is detected as writing execution code, i.e. PE file structure, then the HDF driver would deny the I/O request by returning a status 'STATUS_ACCESS_DENIED' to the I/O Manager which then fails the request and informs the caller application, e.g. and Outlook® email client. With this method it is only

necessary to check the PE header, unlike known anti-virus tool which have to scan the entire file for a malware signature because all executables will have a PE header.

For the reasons mentioned above, an attempt by a trojan or malware to attach itself to an existing executable would be caught, since it needs to write/add itself to the target image file. HDF can block Alternative Data Stream (ADS) as well. If it is DLL-injection type of attack, then the malware does not need to change the target executable but just "injects" itself in the target memory space and piggybacks on the hijacked process, which performs the deeds of the malware. The attack is to make the target process load and run the attacker's DLL. However, the malware needs to transfer itself (execution body, as a DLL mostly) to the victim machine and cannot do it "over the web". HDF would not protect against memory-based attacks as such, but would stop the DLL being installed to the computer, which is essential for this attack to work.

Exceptions to blocking are provided to allow explicitly-approved write requests. Before returning a 'STATUS_ACCESS_DENIED', the HDF driver checks if the user has defined a rule to indicate that either (a) the requesting process is trusted or (b) the system account under which the request made is trusted. If there is an affirmative match, the HDF driver passes the I/O request to the destined device driver, e.g. NTFS, to perform the requested operation, via a call IoCallDriver(...) with the original IRP.

Under Microsoft® Windows® 32-bits operating systems, all native applications are Win32 executables. The operating system loads and runs execution code and does not differentiate if the code is legitimate or not, i.e. whether the user allows running of the specific code. For brevity, access control list (ACL) and user privileges will not be described here, which are largely irrelevant in the context of an embodiment of the present invention. With HDF, write-denial control would not typically be affected by the access control list (ACL) settings nor user privilege on the computer.

To run an application, the OS checks whether the execution code conforms to the Portable Executable File Format specification, e.g. PE format on all Windows 32-bit operation systems. This applies also to interpreted code, e.g. VB script, and Microsoft® specifies the interpreted code 'run-time-engine' must be in PE format.

The PE format is a Microsoft® specification that all PE files have a standard file structure with an "IMAGE_NT_HEADERS" signature, code and data sections. A PE file starts with two signature characters 'MZ' (called the DOS MZ header) and followed, after a DOS stub, by another PE signature with the characters 'PE\0\0'.

The OS recognizes an executable file by its file extension, and the loader ensures it is a valid PE file when loading the file and run the execution code. A requirement for execution code is that the executable file must be accessible on disk, local or remote, i.e. the execution code cannot reside on a remote Web server.

Checking the existence of the PE file format structure is a documented method by which the HDF driver can identify an executable file. This method is also used by other anti-malware applications for different purposes to HDF.

The NT kernel treats its internal entities as objects, these objects represent files, devices and so on. The kernel internal objects can be either named or unnamed. For named device drivers, the kernel maintains two lists of names in its internal object tree database. Each named device driver, e.g. the HDF filter driver, may have two names in the database. One is the driver's NT device name (Microsoft's terminology) intended for a driver to refer to the device it manages, and the other is a 'symbolic link' name. Of particular interest in an embodiment of the present invention is the HDF driver's symbolic link name, which is described as the driver's identifier for brevity here. It is noted that these internal names are not the driver image file name. For example, the HDF driver's NT device name might be "Device\HDFIoCtl", its symbolic link name might be "DosDevice\HDF" (default), and the image file filename might be "HDF.SYS" (which would generally not change).

Both the NT device name and symbolic link name are supplied by the device driver to the operating system kernel, typically during the driver's initialization routine. A device driver creates the NT device name by a kernel API call IoCreateDevice(...), and the symbolic link name by a call to IoCreateSymbolicLink(...).

The symbolic link name is for the sole purpose of making the driver visible to user mode applications. A user mode application requires a driver's symbolic link name in a function call to communicate with the driver. When an application requires to communicate with a device driver, it calls Win32 APIs CreateFile(...) passing in the driver's symbolic link name as a parameter, followed by DeviceIoControl(...) using the returned driver's handle as parameter. An application is not able to use the driver NT device name that is visible in the kernel space only.

To control computer storage devices, HDF is implemented at the operating system kernel level close to the hardware layer drivers. Technically, HDF is a "file system filter driver" in the Microsoft® operating system architecture. HDF intercepts 'write-file' instruction (not API calls) issued by the system to the file system drivers.

In the initialization routine, the HDF driver makes the function calls of IoRegisterFsRegistrationChange(...) and IoAttachDeviceByPointer(...). After the calls, the kernel I/O Manager routes I/O requests that bound for the target device to the HDF filter driver first (as set out above). From that point on, the HDF driver receives I/O requests to all storage devices on a system and performs write-denial control, until the system shuts down.

The processing of an I/O request in the kernel, and the role of HDF, is illustrated in more detail in Figure 4. In brief, the steps shown are:

1. A user mode API call gets mapped to a kernel call.
2. The I/O System Services serves the I/O and…
3. involves I/O Manager.
4. The I/O Manager creates a new IRP for that request.
5. The I/O Manager sends the IRP down the driver stack, where the HDF may intercept it.
6. The destined driver completes the IRP, if not blocked by HDF
7. The I/O request is completed.
8. The I/O status is copied to the subsystem address space.
9. The IRP is freed.
10. A handle and/or NT STATUS is returned.

As can be seen, HDF intercepts close to the hardware level. By contrast, an application such as FileMonster® intercepts much higher up, as marked in Figure 4. If a blocking occurs with HDF, steps 6 to 8 are skipped, with HDF returning the IRP to I/O Manager to perform step 9.

HDF intercepts e.g. the I/O Manager's "write-file" instruction and passes on other I/O instructions not relevant to HDF to the destined file system driver such as NTFS. On receiving a 'write-file' instruction contained in an IRP (described above), HDF checks and acts accordingly:

- If the write-file target is an executable not approved by the user – return a "STATUS_ACCESS_DENIED" status so that the I/O Manager informs the caller of failure. In this case, no executable file will be stored on the target storage device (HDF primary objective).
- If the write-file target is an executable approved by the user – no action by HDF and pass the IRP to the destined file system driver to carry out the writing.
- If the write target is not an executable, e.g. business.doc – no action by HDF and pass the IRP to the destined file system driver to carry out the writing.

HDF can determine if a write-file target is an executable file by checking:

- The file extension – from a Microsoft documented executable file extensions list, and;
- The executable file format signature called PE format, when the file extension is not covered above (as described above).

HDF can determine if a user approves writing of an executable by:

- A list of user-defined trusted processes that are allowed to write executable files – the default being none so that no process is allowed to write an executable file by default.

- A list of user-defined trusted system logon accounts (user and group account scope) that are allowed to write executable files – the default being none so that no account privilege is allowed to write an executable file by default.
- User action, i.e. user turns off HDF blocking via an administration application to install software; in one implementation the administration application is a standard Win32 command line application called HDFControl.exe and its GUI counterpart HDFMonitor.exe.

The HDF driver can obtain the above two lists (trusted processes and trusted accounts) from two sources:

- The Windows specified Registry entry for driver parameters at [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\driver_name\Parameters], where driver_name is, for example, "HDF", or
- Passed in as parameters by the administration applications at run time.

In general, an HDF implementation would not block writing of data or non-executable file types, although it is simple to reconfigure HDF to control writing of any file type in addition to executable file types.

The HDF driver's ability to block writing of executable files would not typically be affected by the requesting process's account privileges. Therefore, malware would not interfere with blocking even it has administrator or Local System elevated privileges obtained by successful attacks. However, to allow a user to install new software and system patching, HDF can provide an administration application for turning off blocking temporarily. An embodiment of the present invention can use a security method to ensure only the intended administration application is able turn off HDF blocking, and such a feature can be an important part of HDF application security.

Three such methods will be described below for authenticating the calling application. The first one is a completely new approach and the others involve using existing cryptography techniques in a new context to resolve authentication problems particularly associated within the context of an embodiment of the present invention.

Before a description of these authentication methods, the method by which a malicious process can attempt an attack will first be described in brief.

The administration application will be sending runtime instructions to the HDF driver, and in a secure approach the HDF driver would ensure that the instructions come from its administration application and not other malicious process. The threat comes from impersonation attacks because any application can send instructions to a driver if it knows the driver's identifier. Technically, the term driver's identifier refers the driver's internal 'symbolic link name', as described above.

The Microsoft® architecture enables an application to communicate with a kernel driver by calling two functions; CreateFile(...) and DeviceIoControl(...). One of the function parameters is the driver's identifier; in an HDF context this is "DosDevices\HDF". The driver provides its identifier to the operating system at runtime via a function call IoCreateSymbolicLink(...). Through this call, the system knows by name which driver to receive a calling application's instruction.

The risk to HDF is that a malicious process could establish an extra illegal communication path in parallel to the administration application and send instructions to the HDF filter driver. For this attack to work, the malicious code needs to know the HDF driver's identifier. Typically, the driver's identifier is hard-coded in the driver execution code and an attacker can discover it with relative ease, e.g. execution code disassembly. A runtime dynamic identifier thwarts this attack vector. Without the driver's identifier, a malicious process cannot establish communication nor send instructions. This forms the basis of the new HDF solution to mitigate the attack risk, and the other methods involving using cryptographic means to authenticate the communication parties. There is no known method for a driver to authenticate the calling application. It will be appreciated that the described methods can be used in combination to further improve application security.

The first approach described herein is to use a changing dynamic driver's identifier. The scheme is that the HDF driver changes its identifier during runtime via IoCreateSymbolicLink(...) calls, and informs its administration application of the newest identifier by a secure means. Since only the administration application knows HDF driver's

identifier at a specific point in time, only the administration application can communicate with and send instructions to the HDF driver. The change can be triggered by every instruction call or using a time-based scheme. If a malicious program uses an 'expired' identifier to send instructions to HDF driver, the system will not able to dispatch and abort the call. In essence, HDF protects its identifier from arbitrary application to authenticate communication.

By way of further explanation, the HDF driver has a default identifier with which the administration application can communicate for the first time. In the DeviceIoControl() call, the administration application passes to the driver a custom I/O control code, e.g. unblock flag or get configuration settings, and a user-space data buffer. The HDF driver performs the request function and if needed fills up the caller's data buffer. The driver can return the next identifier in the data buffer for the caller to use the next time – this is the "every instruction call" option mentioned above. With the "time-based" option mentioned above, the driver changes its identifier and the administration application makes the same change to the last used identifier independently, but according to some pre-arranged synchronization scheme. For example, the driver may append a number to the end of the identifier's fixed part, e.g. HDF, so that the new identifier would be HDF29 in a scheme where the minutes of the hour are used; 29 minutes passed the hour in this example. In this case the synchronization scheme is a shared secret.

The second and third approaches will now be described. Using cryptographic means to authenticate communication has been used in security applications, but is not known in the context of protecting the sending of instructions from an application to a device driver.

The second approach is to use a standard encryption algorithm such as Advance Encryption Standard (AES - FIP 197) (note: AES is just one of many algorithms that can be used; other examples are RSA, DES etc.) to encrypt the communication data between the administration application and the HDF driver. Because only the two software components know the data decryption key to decrypt data, the communication between the administration application and the driver is secured. To set up the authentication scheme, a standard method called Diffie-Hellman Key Agreement Protocol can be used so that the calling application and the

HDF driver each has a shared-secret key. A malicious process is not able to impersonate the calling application and send false instructions to the driver.

The third approach involves a cryptographic authentication method known as Challenge and Response. To set up the authentication scheme, the Diffie-Hellman Key Agreement Protocol can be used so that the calling application and the HDF driver each has a shared-secret key. As an example, authentication would work by the following steps:

1. The HDF driver sends a random number to the calling application for authentication, the Challenge;
2. The calling application uses the shared-secret key to encrypt the random number and returns the encrypted result to the HDF driver, the Response;
3. The HDF driver then uses its shared-secret key to encrypt the random number it generated before;
4. The HDF driver compares its own result and the calling application's Response;
5. The authentication succeeds if the two results match, because only the two components know the shared-secret key to compute the encrypted values.

Taking into account the above explanations, the steps performed by an embodiment of the present invention will now be described with reference to Figure 5. The program logic flow of an inventive computer protection against malware affection shown in Figure 5 illustrates in more detail an implementation of the file system filter driver 153 and the administration application 141 as described above. By way of example, the implementation is illustrated for a computer running a Microsoft® Windows® operating system.

In step S1, the file system filter driver 153 is registered with the operating system of computer 1 as a file in a kernel mode system driver. For example, the file system filter driver 153 "name.sys" (e.g. "name" here and below may be "hdf", so that the driver would be "hdf.sys") is registered by copying the program file preferably in a write protected manner to the according system folder, e.g. "WINDOWS\System32\Drivers\" (for Windows® XP), and the corresponding value is set in the registry 16, e.g. "[HKEY_LOCAL_MACHINES\ SYSTEM\CurrentControlSet\Services\name]".

In step S2, parameters used for adjusting the operation of the file system filter driver 153 are stored in a protected key of the registry 16, e.g. in the key "[HKEY_LOCAL_MACHINES\SYSTEM\CurrentControlSet\ Services\name\Parameters]". For example, following values are set in the protected registry key: "Active"=dword:00000001" as a flag with a predefined value for activating and deactivating the file system filter driver 153; "Extensions"=".COM; .EXE; .DLL; .BAT; .CMD; .OCX; .VBS; .PIF; .SCR; .CHM; .DRV; .HTA; .SHS; .WMA; .WSF; .WSH; .CPL; .CAB;" as a file extension list for predefining the data format types to be blocked by the file system filter driver 153; "Names" [or "Protected"]="name.SYS; name.LOG; name.EXE; HDFMONITOR.EXE;" as a blocking list of files which are always protected by the filter system 152 program file, the name of the filter system 152 program file's image files, the name(s) of an administration application 141 program file (there may be two such program names; e.g. hdfcontrol.exe and hdfmonitor.exe as set out elsewhere) and the name of a log file; "Sids"="" as an empty non-block user id list being ready to be adapted by the administration application 141 (to allow users or groups to write legitimately to storage, defaulted to empty, i.e. no user or group is allowed by default; it is possible to separate the user ids from the group ids by specifying the former with a "UserSids" parameter and the latter with the "Sids" parameter); "LogPath"="C:\nameLog" as a path definition where to store the log file reporting at least all blocking activities of the file system filter driver 153; and "AllowedProcesses"="" provides a list of process names that the user allows explicitly to write executable files; this defaults to empty but can be changed by the user with the administration application 141. To provide maximum flexibility, HDF parameters can also be changed during runtime. However, with HDF it is not necessary to write to the Registry during runtime, unlike for some other existing applications.

Step S1 and step S2 are part of an Initialization 5 which is conducted at least once during installation of the computer protection against malware affection either by an installation program of the computer program product according to the invention or by a system administrator.

In step S3, the file system filter driver 153 is launched either on user request by the administration application 141 or during each booting process of the computer 1.

In step S4, the file system filter driver 153 reads the parameters stored in the registry 16 as mentioned above.

In step S5, the file system filter driver 153 creates a randomly generated dynamic name, registers the name with the operating system and stores the name in the registry.

In step S6, the file system filter driver 153 receives input/output request packets from the input/output manager 151 and performs the following:

Step S64, checking if a file name of the data file is part of the blocking list stored in the registry 16. If the file name of the data file matches a name of the blocking list, the file system filter driver 153 continues with step S7, otherwise it continues with step S60.

Step S60, checking whether the flag for activating and deactivating the file system filter driver 153 is set to the predefined value. If the flag is set to the predefined value, the file system filter driver 153 continues with step S61, otherwise it continues with step S8.

Step S61, checking whether input/output request packets received is related to a write access request. For example, the file system filter driver 153 checks whether the input/output request packet includes a function code as e.g. "IRP_MJ_CREATE". If the input/output request packets are related to a write access request, the file system filter driver 153 continues with step S62, otherwise it continues with step S8.

Step S62, checking whether the format of the data file associated with the write access request is of the predefined format type. Preferably, the checking of the format is performed by comparing a file extension of the data file associated with the write access request to the file extension list stored in the registry 16. If the data file associated with the write access request is detected to be of the predefined format type the file system filter driver 153 continues with step S63, otherwise it continues with step S8.

Step S63, comparing the user identification information associated with the write access request to the non-block user id list stored in the registry 16. If the user id matches a user or a

user group of the non-block user id list the file system filter driver 153 continues with step S8, otherwise it continues with step S7.

In step S7, the file system filter driver 153 blocks the write access request, informs the I/O manager 151 about the blocking, and logs the blocking activity in the log file.

In step S8, the file system filter driver 153 passes the input/output request packets to a subsequent system driver such as the file system driver 154.

Other alternative embodiments of the computer protection against malware affection according to the invention are realizable. Explicitly mentioned in this context are:

- The file system filter driver can be placed in the kernel below the mounted file system divers, and above the lower level hardware driver for the storage media.
- The data format of a data file can be detected by the file system filter driver reading the file header of the data file.
- Preferably the file extension list for predefining the data format is stored on one hand in the registry and on the other hand in a protected binary data file. The adaptations made by the administration application affect the file extension list in both locations. The file system filter driver reads the file extensions from both locations and produces an error if they don't match. Like this additional security can be implemented into the computer protection against malware affection.

An embodiment of the present invention has one or more of the following technical benefits:

- It can load and run as part of the operating system such that it cannot be disabled, i.e. during system boot up and before any user logging in.
- It can resist hacking and attack attempts to shut down operation, e.g. by hostile malware or user mistakes.
- It is able to identify and block executable files without pattern matching.
- It can function autonomously irrespective to system privileges of the user account, caller processes and threads.
- It can function transparently to users and applications. All applications can run as

normal. There is no requirement for pop-up dialogs or other user alerts during operation.

- It supports local, networked and removable storage devices, and plug-and-play mounted devices are protected automatically.
- Operations, whether blocking or non-blocking (pass through), can be logged for audit trail purposes.
- It has a small application footprint, and does not degrade performance.
- It is compatible with existing anti-virus programs, personal firewalls and scanning tools.
- It does not require maintenance, and has reliable operation.

An embodiment of the present invention:

- Is able to prevents malware infection against a computer using an innovative concept of blocking all executable code entering a computer. This new approach resolves the shortcomings of current technologies used by antivirus, anti-spyware and rootkit detection solutions. It is a new malware-prevention method, rather than a malware-detection approach.
- Is able to protect against the so called "zero-day" malware and recent rootkit spyware without false-positive, with a single tool. "Zero-day" malware refers to malicious code that anti-malware vendors have not yet identified or not released a pattern signature of the malware. Rootkits are malicious software that renders itself undetectable on infected systems.

Although the above description has focussed on the Windows family of operating systems, it will be appreciated that an embodiment of the present invention applies equally and correspondingly to other types of operating systems that support an architecture having a protected kernel environment, e.g. Unix, Linux and Mac OS. The skilled person will readily appreciate that certain components and methods that are specific to Windows® will have corresponding components and methods in other environments. As one example, using the Registry as set out above to store HDF parameters is Windows-specific; on other environments the parameters would be stored by some other means, e.g. file-based or over the network, e.g. using a service/daemon. In addition, use herein of terms such as "input/output

manager", "file system filter driver" and "I/O Request Packets" (in particular their use in the appended claims) is not intended to imply any limitation to a Windows® type of operating system; instead these terms are to be understood as applying to the equivalent part or parts and the equivalent messages in any other type of operating system or environment, whatever terms are normally used when describing that operating system or environment.

With Linux, for example, user applications generally cannot access the hardware directly. The Linux operating system relies on a kernel software comprising various device driver types to provide user applications input/output services to hardware, e.g. ISO 9660 filesystem for CD-ROMs, SMB filesystem for Microsoft Windows servers. Device drivers are developed to integrate into kernel core software or developed as 'loadable device drivers' modules, which are kernel components but not integrated into the kernel core. 'Loadable device drivers' typically load into memory during boot-time. Device drivers provide input/output access to the hardware to which generally user applications cannot access direct.

An embodiment of the present invention can be developed to a Linux filter device drivers and adapted to intercept input/output requests to hardware from user applications. In the same manner as described above, an embodiment of the present invention when implanted with on a computer under the Linux operating system will therefore follow the policy of denying writing to hardware storage when the data format is determined as execution code, for example, by examining the file's "execute permissions" attributes, because Linux executable code depend not on a file extension but on the file permissions attributes.

The current operating system for the Macintosh is Mac OS X that is built on top of the Berkeley Unix implementation (FreeBSD 4.4) and Darwin. The OS X kernel is called XNU, which is a modern operating system with a user space and kernel core architecture. The kernel provides mechanisms and interfaces for user applications to make input/output requests to hardware devices through a kernel software called "I/O Kit". The I/O Kit component is the kernel device driver framework and includes numerous device driver types for e.g. hard disks and other storage devices.

Mac OS X, like other OSs, uses files and directories to store data and execution code, and identifies file types by file extensions. Executable files are organized into a directory together

with its resource files (unlike Windows® executables which store execution code and resources in the same file, *.EXE), and is called an "application bundle". The I/O Kit framework supports driver stacking which is a similar concept to Windows layered-drivers architecture as described above. The driver stacking features allow a skilled person to develop a filtering module to intercept writing of executables to storage devices following the same principles as set out above in the main description of an embodiment of the present invention to achieve protection from malware infection.

The skilled person will readily appreciate how to adapt the present invention for use with other such types of operating system, and operating systems for other types of computing devices such as handheld devices, cell phones, Smartphones, PocketPCs, Personal Digital Assistants (PDAs), Blackberry® devices, TV set-top boxes and any future such devices having a computing (CPU) unit and storage. The terms "computer" and "computer program" in the appended claims are intended to cover such devices.

Although the above description has referred to a program stored on a computer-readable medium, it will be appreciated that a computer program embodying the present invention need not be stored on a computer-readable medium and could, for example, be embodied in a signal such as a downloadable data signal provided from an Internet website. The appended claims are to be interpreted as covering a computer program by itself, or as a record on a carrier, or as a signal, or in any other form.

Claims

1. A method of protecting a computer (1) against malware affection, the computer (1) having a data storage (157, 12, 2) and an operating system for managing the data storage (157, 12, 2), the method comprising providing a filter module in the operating system which operates to detect an attempt to store data in the data storage, to determine a data format of the data to be stored in the data storage (157, 12, 2), and to prevent storage of the data if the data format is determined to relate to a predefined type.

2. A method as claimed in claim 1, wherein the filter module is provided as a file system filter driver (153) in a kernel (15) of the operating system.

3. A method as claimed in claim 1 or 2, wherein the filter module is arranged to operate between an input/output manager (151) of the operating system and a driver (154, 156) associated with the data storage (157, 12, 2).

4. A method as claimed in claim 3, wherein the input/output manager (151) and driver associated with the data storage form part of a kernel (15) of the operating system.

5. A method as claimed in claim 4, when dependent on claim 2, further comprising the file system filter driver (153) receiving at least write access calls from the input/output manager (151) intended for the kernel driver associated with the data storage, blocking such write access calls relating to data of the format of the predefined type, and such passing write access calls not so relating.

6. A method as claimed in claim 5, further comprising informing the input/output manager (151) about the blocking and optionally passing of such write access calls.

7. A method as claimed in any one of claims 3 to 6, comprising arranging for I/O Request Packets that are sent from the input/output manager (151) and intended for the driver associated with the storage (157, 12, 2) to be re-routed via the filter module, the re-routed I/O Request Packets being at least those of a type enabling the filter module to detect an attempt to store data in the data storage.

8. A method as claimed in claim 7, comprising querying a function code of a re-routed I/O Request Packet at the filter module to detect an attempt to store data in the data storage

9. A method as claimed in claim 7 or 8, wherein the re-routed I/O Request Packets comprise those relating to the creation of a new file in the data storage.

10. A method as claimed in claim 9, wherein the re-routed I/O Request Packets comprise those having or associated with an IRP_MJ_CREATE function code.

11. A method as claimed in any one of claims 7 to 10, wherein the re-routed I/O Request Packets comprise those relating to the writing to an existing file on the storage.

12. A method as claimed in claim 11, wherein the existing file is a recently-created file.

13. A method as claimed in claim 11 or 12, wherein the re-routed I/O Request Packets comprise those having or associated with an IRP_MJ_WRITE function code.

14. A method as claimed in any one of claims 7 to 13, comprising blocking a re-routed I/O Request Packet at the filter module if it is determined that the format of the data to be stored by way of that I/O Request packet is of the predefined type.

15. A method as claimed in claim 14, when dependent on claim 9, comprising determining the format of the data to be stored by examining the intended file extension of the file to be created.

16. A method as claimed in claim 14 or 15, when dependent on claim 11, comprising determining the format of the data to be stored by examining a file header of the file to be written.

17. A method as claimed in claim 16, wherein the file header is the Portable Executable File Format file header.

18. A method as claimed in any preceding claim, comprising referring to at least one parameter associated with the filter module preferably stored in a protected manner in the data

storage (157, 12, 2), and determining the behaviour of the filter module at least in part in dependence on the at least one parameter.

19. A method as claimed in claim 18, comprising running a protected administration module (141) which allows a verified user to adapt the at least one parameter.

20. A method as claimed in claim 18 or 19, further comprising storing the at least one parameter in a protected part of a system database (16) of the operating system.

21. A method as claimed in claim 20, wherein the system database comprises the Registry.

22. A method as claimed in any one of claims 18 to 21, comprising reading the at least one parameter during the booting of the computer (1).

23. A method as claimed in any one of claims 18 to 22, wherein the at least one parameter comprises a predefined blocking file extension as the predefined type, and comprising the filter module detecting the format of the data to be stored by comparing a specific file extension of the data to the predefined blocking file extension.

24. A method as claimed in any one of claims 18 to 23, wherein the at least one parameter comprises a flag, wherein the flag is able to be set to a predefined status, and comprising the filter module checking the flag and preventing storage of the data if the flag is set to the predefined status.

25. A method as claimed in any one of claims 18 to 24, wherein the at least one parameter comprises a non-block user identification, and comprising the filter module checking the non-block user identification as well as an active user identification and allowing the storage of the data if the non-block user identification matches the active user identification.

26. A method as claimed in any preceding claim, comprising providing an administration application (141) to allow a user to alter the behaviour of the filter module by sending the filter module runtime instructions.

27. A method as claimed in claim 26, comprising using the administration application to turn off blocking temporarily.

28. A method as claimed in claim 26 or 27, when dependent on claim 18, wherein the administration application is operable to alter the behaviour of the filter module by changing, or causing the filter module to change, at least one of the at least one parameter.

29. A method as claimed in claim 26, 27 or 28, comprising authenticating the administration application to the filter module.

30. A method as claimed in claim 29, comprising the filter module storing a name or identifier of the filter module in a protected manner in the data storage (157, 12, 2) in such a way that only the administration application can reasonably learn of the name or identifier and thereby communicate with the filter module.

31. A method as claimed in claim 30, comprising registering the name in the operating system.

32. A method as claimed in claim 30 or 31, wherein the name or identifier of the filter module is stored in a protected manner in the data storage by changing it from time to time during runtime.

33. A method as claimed in any one of claims 29 to 32, comprising authenticating the administration application to the filter module using a cryptographic authentication method.

34. A method as claimed in any preceding claim, further comprising logging activities of the filter module.

35. A method as claimed in any preceding claim, wherein the predefined type of data comprises executable data.

36. A method as claimed in any preceding claim, wherein the filter module operates to prevent storage of such data to any part of the storage.

37. A method as claimed in any preceding claim, wherein the operating system is one of the Windows® family of operating systems.

38. A method of thwarting malware at its propagation phase by performing a method as claimed in any preceding claim so as to block any unauthorised attempt to write any data to the data storage that could potentially constitute malware, thereby preventing the malware from propagating.

39. A method of administrating a computer having a kernel driver and an administration application that communicates with the kernel driver during runtime, comprising authenticating the administration application to the kernel driver.

40. A method as claimed in claim 39, comprising using a cryptographic method to authenticate the administration application to the kernel driver.

41. A method of authenticating an administration application to a kernel driver for use in a method as claimed in claim 39 or 40, comprising using a dynamic driver identifier that is made to change during runtime in a way such that only the administration application can reasonably learn of the current driver identifier and thereby communicate with the kernel driver.

42. A method as claimed in claim 39, 40 or 41, wherein the kernel driver is a file system filter driver.

43. A computer comprising a data storage (157, 12, 2), an operating system for managing the data storage (157, 12, 2), and a filter module provided in the operating system for protecting the computer against malware affection, the filter module being adapted to detect an attempt to store data in the data storage, to determine a data format of the data to be stored in the data storage (157, 12, 2), and to prevent storage of the data if the data format is determined to relate to a predefined type.

44. A computer program which, when run on a computer, causes the computer to carry out a method as claimed in any one of claims 1 to 42.

45. A computer program which, when loaded into a computer, causes the computer to become one as claimed in claim 43.

46. A computer program as claimed in claim 44 or 45, carried on a carrier medium.

47. A computer program as claimed in claim 46, wherein the carrier medium is a transmission medium.

48. A computer program as claimed in claim 46, wherein the carrier medium is a storage medium.

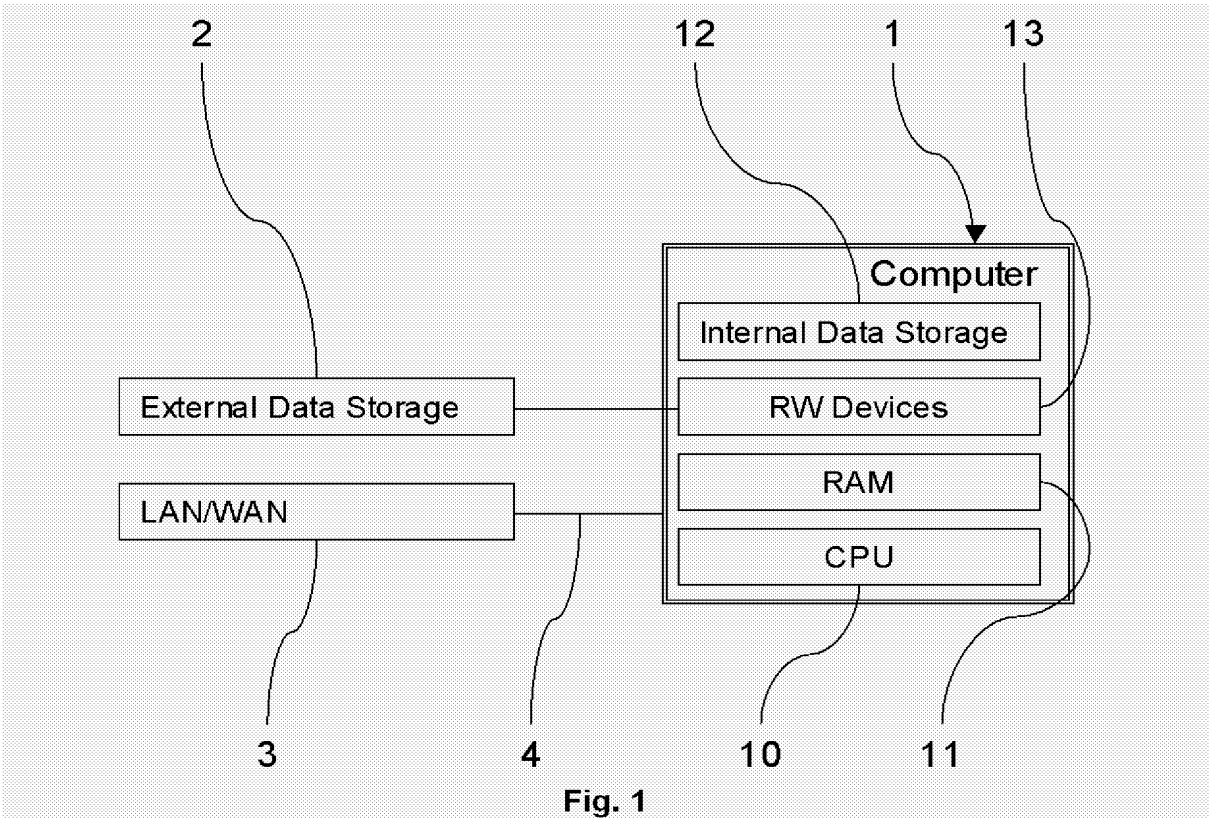

2
12
1
13
Computer
Internal Data Storage
External Data Storage
RW Devices
RAM
LAN/WAN
CPU
3
4
10
11

Fig. 1

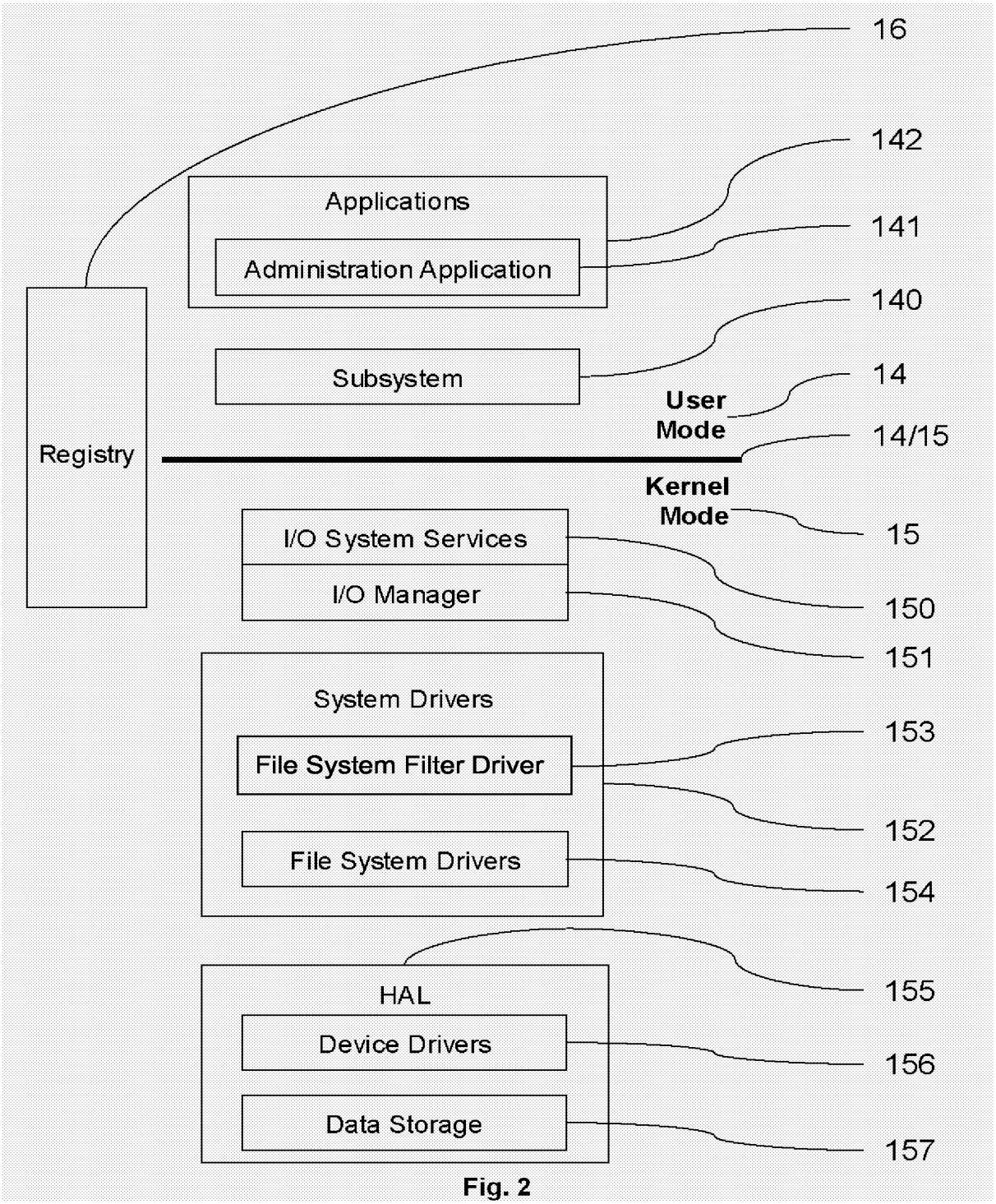

16
142
Applications
141
Administration Application
140
Subsystem
14
User Mode
14/15
Registry
Kernel Mode
15
I/O System Services
150
I/O Manager
151
System Drivers
153
File System Filter Driver
152
File System Drivers
154
155
HAL
Device Drivers
156
Data Storage
157

Fig. 2

FIG. 3

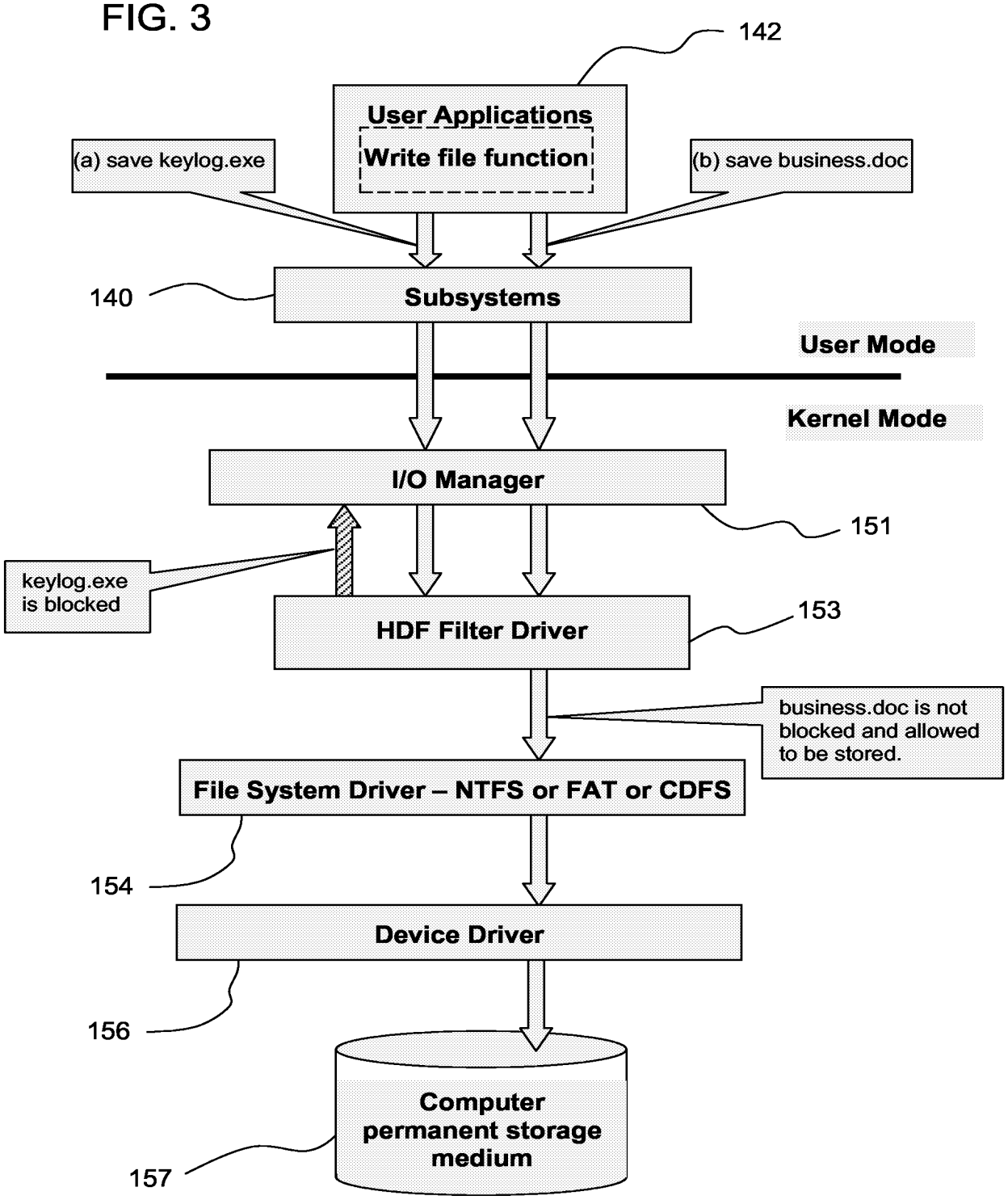
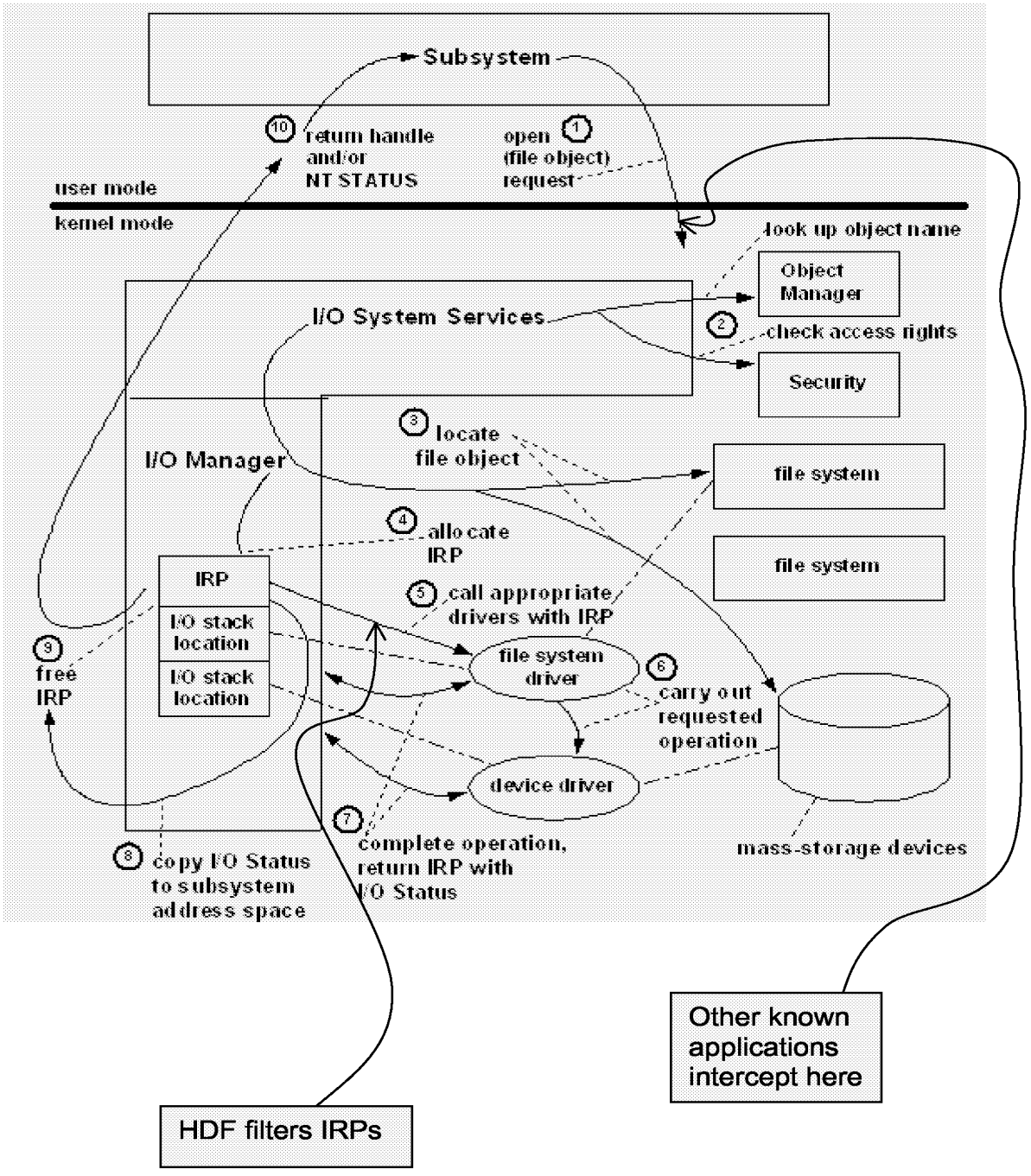

142
User Applications
Write file function
(a) save keylog.exe
(b) save business.doc
140
Subsystems
User Mode
Kernel Mode
I/O Manager
151
keylog.exe
is blocked
HDF Filter Driver
153
business.doc is not
blocked and allowed
to be stored.
File System Driver – NTFS or FAT or CDFS
154
Device Driver
156
Computer
permanent storage
medium
157

FIG. 4


Subsystem
10 return handle and/or NT STATUS
open (file object) request 1
user mode
kernel mode
look up object name
Object Manager
I/O System Services
2 check access rights
Security
3 locate file object
I/O Manager
file system
4 allocate IRP
file system
IRP
I/O stack location
I/O stack location
5 call appropriate drivers with IRP
9 free IRP
file system driver
6 carry out requested operation
device driver
7 complete operation, return IRP with I/O Status
8 copy I/O Status to subsystem address space
mass-storage devices
Other known applications intercept here
HDF filters IRPs

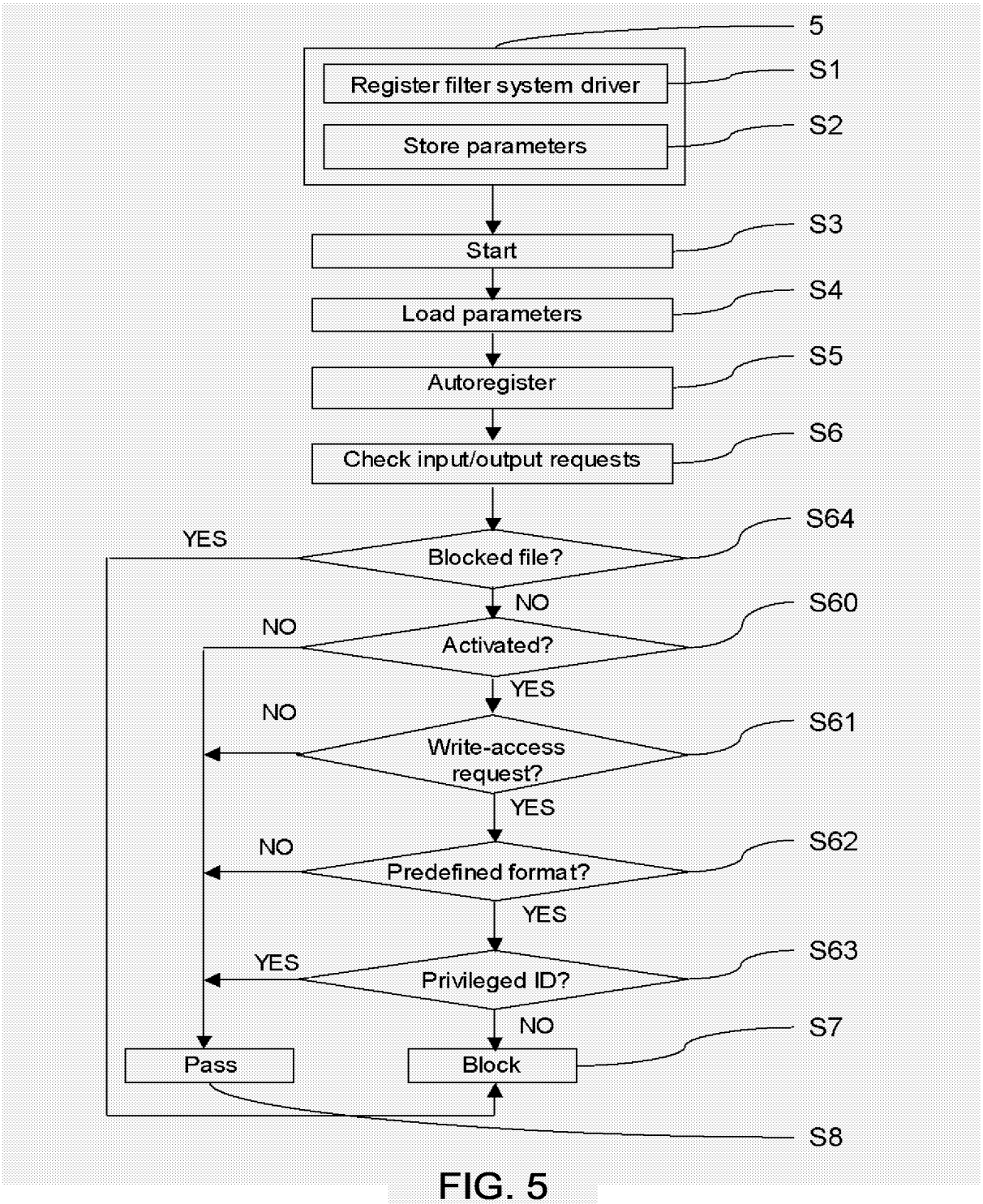

5
Register filter system driver
S1
Store parameters
S2
Start
S3
Load parameters
S4
Autoregister
S5
Check input/output requests
S6
Blocked file?
S64
YES
NO
Activated?
S60
NO
YES
Write-access request?
S61
NO
YES
Predefined format?
S62
NO
YES
Privileged ID?
S63
YES
NO
Pass
Block
S7
S8

FIG. 5

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 July 2006 (27.07.2006)

PCT

(10) International Publication Number
WO 2006/077443 A2

(51) International Patent Classification:
G06F 21/00 (2006.01)

(21) International Application Number:
PCT/GB2006/050013

(22) International Filing Date: 18 January 2006 (18.01.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
05001120.4 20 January 2005 (20.01.2005) EP

(71) Applicant and
(72) Inventor: **ROTHWELL, William, Grant** [GB/CH]; Lehnmattstrasse 3, CH-4573 Lohn-ammannsegg (CH).

(74) Agent: **BREWER, Michael**; Marks & Clerk, 4220 Nash Court, Oxford Business Park South, Oxford, Oxfordshire OX4 2RU (GB).

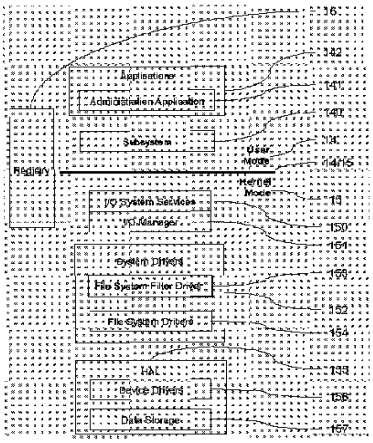
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: COMPUTER PROTECTION AGAINST MALWARE AFFECTION



(57) Abstract: A method is provided of protecting a computer (1) against malware affection. The computer (1) has a data storage (157, 12, 2) and an operating system for managing the data storage (157, 12, 2). The method comprises providing a filter module in the operating system which operates to detect an attempt to store data in the data storage, to determine a data format of the data to be stored in the data storage (157, 12, 2), and to prevent storage of the data if the data format is determined to relate to a predefined type. The filter module may be provided as a file system filter driver (153) in a kernel (15) of the operating system. The filter module may be arranged to operate between an input/output manager (151) of the operating system and a driver associated with the data storage (157, 12, 2). The input/output manager (151) and driver associated with the data storage may form part of the kernel (15) of the operating system.

WO 2006/077443 A2

Computer protection against malware affection

The present invention relates generally to computer protection against malware affection and more particularly to a computer program product and a method for protecting a computer against malware affection and with a computer protected against malware affection.

Malware is a short term used in the art for malicious software summarizing computer programs performing actions on computer systems without the consent of a user, often developed for the purpose of doing harm to the computer system or the user, and comprises for example so-called viruses, worms, Trojan horses, backdoors and spyware. The intended actions of malware widely varies and include stealing users' personal information, stealing other data, abusing the computer system or simply vandalize.

Usually a target computer is first attacked by malware in a warhead phase exploiting weaknesses of the computer systems, as for example producing a buffer overrun, for enabling a propagation phase. The warhead phase is executed by a very small program code portion of the malware. In the propagation phase the body program code of the malware is transferred to the target computer. Typically several phases follow the propagation phase by executing the body program code as for example a target selection phase, in which the malware is seeking for other computers accessible from the target computer, a scanning phase, in which accessible other computers are checked to be suitable target computers for further propagation, and a payload phase, in which the intended purposes of the malware are performed.

To prevent malware affecting a computer system several protections are known in the art as for example virus filters running on computers or firewalls controlling the traffic between connected computer systems. These protections recognize malware using malware descriptions which define specific characteristics of certain malware. They usually analyze data, check if the data shows some of the specific characteristics and take action if the data is identified as being malware. For example, the data is deleted or quarantined. A common technique for analyzing the data is pattern matching, wherein the data is scanned for predefined specific bit patterns defined to be malware bit patterns in the malware descriptions. The analysis usually is quite processing-intensive and can reduce the performance of the

computer system. The effectiveness of the above described protections is highly dependent on the up-to-dateness of the malware descriptions since malware is only recognized after the malware descriptions have been updated. Nevertheless, in the time period between the deployment of malware and the update of the malware descriptions in the computer system the computer system is vulnerable. It is therefore a delicate, difficult, labour-intensive and cost-intensive task to update the descriptions as fast as possible, which is usually done by specialized service providers.

Schmid et al "Protecting data from malicious software", Proc. 18th Annual Computer Security Applications Conference, 2002, 9-13 December 2002, Piscataway, NJ, USA, IEEE, describes an application called FileMonster®. The FileMonster® application uses a driver that works at the imaginary boundary between the user mode and protected kernel space; it intercepts at the point where user mode function calls are translated into kernel API calls, i.e. using service dispatch tables mapping. This is an undocumented approach.

The W32/Swen.A@mm worm that appeared in September 2003 is described in Frisk Software International: "W32/Swen.A@mm" (retrievable from the Internet at http://web.archive.org/web/20040411010214/http://www.f-prot.com/virusinfo/print/descriptions/swena.html). As a way to persist on the victim computer, the worm made auto-start entries in the Registry so that the system would load the worm's executable code on boot up. A random name was used for the worm's image file to avoid detection, i.e. <random_characters>.exe.

Winability Software Corp's "Folder Guard User's Guide – Using the filters" (retrievable from the Internet at: http://web.archive.org/ web/20041023085327/www.winability.com/folderguard/users-guide_filters.htm; see also http://www.winability.com/folderguard/users-guide-printable.htm) describes a system that stops other users from opening or seeing a user's personal files; it can also protect sensitive system files from modification or destruction. Folder Guard allows and disallows program execution using a white list (explicitly allowing certain applications to run) and a black list (explicitly disallowing certain applications to run). This implies the program files already exist on the computer, and Folder Guard performs check to allow or disallow when the program is able to run.

The McAfee® Security "Product Guide – VirusScan Enterprise version 7.0" (7 March 2003) document describes a virus scanning, detection and removal program that can use of file extensions to allow a user to configure the program scanning behaviours. Based on the user's selection, the program can scan the file on 'read' access, 'write access' or 'on network'. The application uses conventional scanning and pattern matching techniques to detect viruses. Similarly, the "Sophos® Antivirus – Windows NT/2000/XP single user installation guide" describes an application that scans a file for virus detection on a read access, or on write and other accesses.

EP-A-1,429,247 (Microsoft Corporation) relates to a new approach as to how a file system filter driver should be implemented.

In view of the disadvantages of the hitherto-known, above described protections against malware affection, it is desirable to provide an improved, effective, fast and easy to maintain computer protection against malware.

According to a first aspect of the present invention, there is provided a method of protecting a computer against malware affection, the computer having a data storage and an operating system for managing the data storage, the method comprising providing a filter module in the operating system which operates to detect an attempt to store data in the data storage, to determine a data format of the data to be stored in the data storage, and to prevent storage of the data if the data format is determined to relate to a predefined type.

The filter module may be provided as a file system filter driver in a kernel of the operating system.

The filter module may be arranged to operate between an input/output (I/O) manager of the operating system and a driver associated with the data storage.

The input/output (I/O) manager may form part of a kernel of the operating system.

The driver associated with the data storage may form part of a kernel of the operating system.

The method may comprise the file system filter driver receiving at least write access calls from the input/output (I/O) manager intended for the kernel driver associated with the data storage, blocking such write access calls relating to data of the format of the predefined type, and such passing write access calls not so relating.

The method may comprise informing the input/output (I/O) manager about the blocking and optionally passing of such write access calls.

The method may comprise arranging for I/O Request Packets that are sent from the input/output (I/O) manager and intended for the driver associated with the storage to be re-routed via the filter module, the re-routed I/O Request Packets being at least those of a type enabling the filter module to detect an attempt to store data in the data storage.

The method may comprise querying a function code of a re-routed I/O Request Packet at the filter module to detect an attempt to store data in the data storage

The re-routed I/O Request Packets may comprise those relating to the creation of a new file in the data storage.

The re-routed I/O Request Packets may comprise those having or associated with an IRP_MJ_CREATE function code.

The re-routed I/O Request Packets may comprise those relating to the writing to an existing file on the storage. The existing file may be a recently-created file.

The re-routed I/O Request Packets may comprise those having or associated with an IRP_MJ_WRITE function code.

The method may comprise blocking a re-routed I/O Request Packet at the filter module if it is determined that the format of the data to be stored by way of that I/O Request packet is of the predefined type.

The method may comprise determining the format of the data to be stored by examining the intended file extension of the file to be created.

The method may comprise determining the format of the data to be stored by examining a predetermined portion of the file to be written, for example its file header. The file header may be the Portable Executable File Format (PE) file header.

The method may comprise referring to at least one parameter associated with the filter module preferably stored in a protected manner in the data storage, and determining the behaviour of the filter module at least in part in dependence on the at least one parameter.

The method may comprise running a protected administration module which allows a verified user to adapt the at least one parameter.

The method may comprise storing the at least one parameter in a protected part of a system database of the operating system. The system database may comprise the Registry.

The method may comprise reading the at least one parameter during the booting of the computer.

The at least one parameter may comprise a predefined blocking file extension as the predefined type, and comprising the filter module detecting the format of the data to be stored by comparing a specific file extension of the data to the predefined blocking file extension.

The at least one parameter may comprise a flag, wherein the flag is able to be set to a predefined status, and comprising the filter module checking the flag and preventing storage of the data if the flag is set to the predefined status.

The at least one parameter may comprise a non-block user identification, and comprising the filter module checking the non-block user identification as well as an active user identification and allowing the storage of the data if the non-block user identification matches the active user identification.

The method may comprise providing an administration application to allow a user to alter the behaviour of the filter module by sending the filter module runtime instructions.

The method may comprise using the administration application to turn off blocking temporarily.

The administration application may be operable to alter the behaviour of the filter module by changing, or causing the filter module to change, at least one of the at least one parameter.

The method may comprise authenticating the administration application to the filter module.

The method may comprise the filter module storing a name or identifier of the filter module in a protected manner in the data storage in such a way that only the administration application can reasonably learn of the name or identifier and thereby communicate with the filter module.

The method may comprise registering the name in the operating system.

The name or identifier of the filter module may be stored in a protected manner in the data storage by changing it from time to time during runtime.

The method may comprise authenticating the administration application to the filter module using a cryptographic authentication method.

The method may comprise logging activities of the filter module.

The predefined type of data may comprise executable data.

The filter module may operate to prevent storage of such data to any part of the storage.

The operating system may be one of the Windows® family of operating systems.

According to a second aspect of the present invention, there is provided method of thwarting malware at its propagation phase by performing a method as claimed in any preceding claim so as to block any unauthorised attempt to write any data to the data storage that could potentially constitute malware, thereby preventing the malware from propagating.

According to a third aspect of the present invention, there is provided method of administrating a computer having a kernel driver and an administration application that communicates with the kernel driver during runtime, comprising authenticating the administration application to the kernel driver.

The method may comprise using a cryptographic method to authenticate the administration application to the kernel driver.

According to a fourth aspect of the present invention, there is provided a method of authenticating an administration application to a kernel driver for use in a method according to the third aspect of the present invention, comprising using a dynamic driver identifier that is made to change during runtime in a way such that only the administration application can reasonably learn of the current driver identifier and thereby communicate with the kernel driver.

The kernel driver may be a file system filter driver.

According to a fifth aspect of the present invention, there is provided computer comprising a data storage, an operating system for managing the data storage, and a filter module provided in the operating system for protecting the computer against malware affection, the filter module being adapted to detect an attempt to store data in the data storage, to determine a data format of the data to be stored in the data storage, and to prevent storage of the data if the data format is determined to relate to a predefined type.

According to a sixth aspect of the present invention, there is provided computer program which, when run on a computer, causes the computer to carry out a method according to the first aspect of the present invention.

According to a seventh aspect of the present invention, there is provided a computer program which, when loaded into a computer, causes the computer to become one according to the fifth aspect of the present invention.

The computer program may be carried on a carrier medium. The carrier medium may be a transmission medium. The carrier medium may be a storage medium.

One embodiment of the present invention provides a computer program product for protecting a computer against malware affection, wherein the computer has a data storage and an operating system managing the data storage. The computer program product comprises computer readable program code means for causing the computer to run a filter module. The computer readable program code means cause the computer to run the filter module in the operating system, and the filter module is configured to detect a format of data to be stored or altered in the data storage. The filter module prevents storage or alteration of the data if the format is of a predefined type. By predefining the types of format of malware data, which typically is any format executable by the operating system, the filter module interferes the malware life cycle at the critical point of its propagation phase by preventing the storage of the malware in the data storage. Without propagation the malware is not able to execute its payload and is effectively terminated. The detection of the format can easily happen without any prior knowledge about current malware and without further data analysis such as scanning the data and searching for specific bit patterns. The running filter module protects the computer in an efficient and fast way, e.g. as a pure software implementation, against known and future malware using the predefined data format types, with no need of updating any malware descriptions and irrespective to operating system privileges of caller processes and threads. It also protects the computer against malware using common mechanisms to bypass its detection such as polymorphism and renaming files. Because the computer runs the filter module in the operating system, the protection against malware is effective on all current data storage devices accessible by the computer as well as all future data storage devices accessible by the computer without requiring any adaptation related to the malware protection.

In a preferred embodiment the program code means are adapted to cause the computer to run the filter module as a file system filter driver in a kernel of the operating system. This assures

that all read/write access requests originated from any user mode application or from any system process, which are directed to any data storage device associated to the computer, are treated by the filter module. The filter module running in the kernel does not have any additional impact on the behaviour or the performance of the computer. Since the kernel of the operating system is launched at a very early stage of the booting process of the computer, the malware protection is effective from scratch and especially before system processes are running. Furthermore, running the filter module in the kernel allows a malware protection irrespective of users' operating system access rights or the operating system access rights of a user simulated by the malware.

Preferably the program code means are adapted to cause the computer to run the file system filter driver configured to receive read and write access calls from an input/output manager of the kernel and further configured to block write access calls related to data of the format of the predefined type. Further the program code means are adapted to run the file system filter driver configured to inform the input/output manager about the blocking and finally to transfer non-blocked read and write access calls to further system drivers of the kernel. Because the file system filter driver receives the read and write access calls from the input/output manager and not directly from an application running in a user mode of the computer, the filter driver is independent of service dispatch tables matching user mode access calls to kernel mode access calls. These service dispatch tables usually are not documented and not supported by the operating system manufacturer and may change in future versions of the operating system. By transferring non-blocked read and write access calls to further system drivers, i.e. file system drivers as for example NTFS (New Technology File System - one of the file systems for the Windows NT (New Technology) operating system by Microsoft Corporation), FAT (File Allocation Table - a file system for various operating systems of Microsoft Corporation) or CDFS (CD ROM (Compact Disk Read Only Memory) File System - a driver for CD ROM players used in certain operating systems of Microsoft Corporation), and not directly to hardware drivers of storage media, the filter module does not need to consist of different classes of filter drivers, each fitting to a storage medium present on the computer, for example a local or a remote hard disk, a backup tape, a memory stick or an optical read/write device.

In a preferred embodiment the program code means are adapted to cause the computer to store a parameter of the filter module in a protected manner in the data storage, to run the filter module adjusted by the parameter, and to run a protected administration module configured to allow a verified user to adapt the parameter. Using parameters for the adjustment of the filter module is an easy way to be able to adapt the malware protection to the changing needs during life time of the malware protection even while it is running. The administration module can be used to adapt the parameter and to trigger the adjustment of the filter module using the changed parameter. It is typically run as a protected user application.

Preferably the program code means are adapted to cause the computer to store the parameter in a protected part of a system database of the operating system, further to cause the computer to read the parameter while booting the computer, and finally to cause the computer to run the filter module adjusted by the parameter. System databases, such as for example the Registry of the Windows® operating systems by Microsoft Corporation, are commonly used to store configuration information of a computer system comprising hardware information, user information, application information, and other information. It can easily be used to store the parameters for adjusting the filter module. In particular while booting the computer the parameters can be read and the filter module can be adjusted. This assures that the malware protection is always customized to the needs of the user and does not run in another configuration.

In a preferred embodiment the program code means are adapted to cause the computer to run the filter module in a way to store a name of the filter module in a protected manner in the data storage and in a way to register the name in the operating system. Such a filter module is able to change its name from time to time or on request, e.g. triggered by a user operating the administration module, and to register the changed name, i.e. the file system filter driver name, in the operating system and to store the name, particularly as a parameter, in a protected part of the system database, for making it accessible to the registration of the filter module during the boot process of the computer. Using such a dynamic name of the filter module allows the filter module to be undetectable and unreferable for programs (including malware) other than the administration module, which knows how and where to look up the current name of the filter module.

The program code means can be adapted to cause the computer to store a predefined blocking file extension as the predefined type of data format in a protected manner in the data storage and to run the filter module in a way to detect the format of the data to be stored or to be altered by comparing a specific file extension of the data to the predefined blocking file extension. The extension of a data file is a common way to define the data format of a data file and consists for example of a dot and few letters attached to the name of the data file, i.e. Windows operating systems by Microsoft Corporation use in general a dot and three letters following the file name to specify the data format e.g. ".doc", ".xls" or ".txt". The file extension can be used as an indication if a data file is potentially executable by the operating system and therefore be potential malware. A list of such potential malware file extensions can be stored in the data storage, in particular as parameters in a protected part of the system database, in order to be compared to the file extension of the data to be stored or to be altered in the data storage. For Windows operating systems by Microsoft Corporation This can for example be a list comprising the extensions ".COM", ".EXE", ".DLL", ".BAT", ".CMD", ".OCX", ".VBS", ".PIF", ".SCR", ".CHM", ".DRV", ".HTA", ".SHS", ".WMA", ".WSF", ".WSH", ".CPL", ".CAB".

Preferably the program code means are adapted to cause the computer to store a flag in a protected manner in the data storage, wherein the flag is able to be set to a predefined status. The filter module is configured to check the flag, and is further configured to detect the format of the data to be stored or to be altered as well as to prevent the storage or alteration of the data if the flag is set to the predefined status. Such a flag, in particular stored as a parameter in a protected part of the system database, can be used to switch the filter module on or off during the boot process of the computer, or triggered by the administration module. For example, there can be the need to switch the filter module off to be able to store data files of a blocked predefined data format in the data storage, e.g. during installation of new software on the computer.

The program code means can be adapted to cause the computer to store a non-block user identification in a protected manner in the data storage, and to run the filter module in a manner to check the non-block user identification as well as an active user identification, and to intercept the prevention of the storage or alteration of the data if the non-block user identification matches the active user identification. The storage of such non-block user

identifications, in particular as parameters in a protected part of the system database, can be useful to ease the administration of a computer on which often data of the blocked predefined format is regularly saved, e.g. by installation of software. A user who is identified as a non-block user bypasses the filter module in a secure controlled way.

Preferably the program code means are adapted to cause the computer to log activities of the filter module. The log of the activities of the filter module can e.g. be used to analyze the effectiveness of the malware protection, to find unwanted data file filtration, or just for statistical reasons.

Another embodiment of the present invention relates to a computer protected against malware affection having a data storage and an operating system managing the data storage. The computer is configured to run a filter module in the operating system, wherein the filter module is configured to detect a format of data to be stored or altered in the data storage and to prevent storage or alteration of the data if the format is of a predefined type.

The advantages of the computer protected against malware affection and of its preferred embodiments are the same as the advantages of the corresponding computer program product and the corresponding preferred embodiments of the computer program product. They are therefore not explicitly repeated and can be found above.

In a preferred embodiment the computer is adapted to run the filter module as a file system filter driver in a kernel of the operating system.

Preferably the file system filter driver is configured to receive read and write access calls from an input/output manager of the kernel. Further, it is configured to block write access calls related to data of the format of the predefined type, to inform the input/output manager about the blocking and to transfer non-blocked read and write access calls to further system drivers of the kernel.

In a preferred embodiment the computer is adapted to store a parameter of the filter module in a protected manner in the data storage. The filter module is configured to run adjusted by the

parameter, and the computer is adapted to run a protected administration module allowing a verified user to adapt the parameter.

Preferably the computer is adapted to store the parameter in a protected part of a system database of the operating system, further to read the parameter while booting, and finally to run the filter module adjusted by the parameter.

In a preferred embodiment the filter module is configured to store a name of the filter module in a protected manner in the data storage and to register the name in the operating system.

The computer can be adapted to store a predefined blocking file extension as the predefined type in a protected manner in the data storage, wherein the filter module is configured to detect the format of the data to be stored or to be altered by comparing a specific file extension of the data to the predefined blocking file extension.

Preferably the computer is adapted to store a flag in a protected manner in the data storage. The flag is able to be set to a predefined status, and the filter module is configured to check the flag and to detect the format of the data to be stored or to be altered as well as to prevent the storage or alteration of the data if the flag is set to the predefined status.

The computer can be adapted to store a non-block user identification in a protected manner in the data storage. The filter module is configured to check the non-block user identification as well as an active user identification and to intercept the prevention of the storage or of the alteration of the data if the non-block user identification matches the active user identification.

Preferably the computer is adapted to log activities of the filter module.

Another embodiment of the present invention relates to a method for protecting a computer against malware affection, the computer having a data storage and an operating system managing the data storage. The method comprises the steps of running a filter module in the operating system, with the filter module detecting a data format of data to be stored or altered in the data storage and preventing the storage or alteration of the data if the format is of a predefined type.

The advantages of the method and of its preferred embodiments are the same as the advantages of the corresponding computer program product and the corresponding preferred embodiments of the computer program product. They are therefore not explicitly repeated and can be found above.

In a preferred embodiment the filter module is run as a file system filter driver in a kernel of the operating system.

Preferably the method comprises the steps of the file system filter driver receiving read and write access calls from an input/output manager of the kernel, blocking write access calls related to data of the format of the predefined type, informing the input/output manager about the blocking and transferring non-blocked read and write access calls to further system drivers of the kernel.

In a preferred embodiment the method comprises the steps of storing a parameter of the filter module in a protected manner in the data storage, adjusting the filter module by the parameter, and running a protected administration module which allows a verified user to adapt the parameter.

Preferably the method comprises the steps of storing the parameter in a protected part of a system database of the operating system, reading the parameter while booting the computer and adjusting the filter module using the parameter.

In a preferred embodiment the method comprises the steps of the filter module storing a name of the filter module in a protected manner in the data storage and registering the name in the operating system.

The method can comprise the steps of storing a predefined blocking file extension as the predefined type in a protected manner in the data storage and detecting the format of the data to be stored or to be altered by comparing a specific file extension of the data to the predefined blocking file extension by the filter module.

Preferably the method comprises the steps of storing a flag in a protected manner in the data storage, wherein the flag is able to be set to predefined status, the filter module checking the flag, detecting the format of the data to be stored or to be altered, and preventing the storage or alteration of the data if the flag is set to the predefined status.

The method can comprise the steps of storing a non-block user identification in a protected manner in the data storage, the filter module checking the non-block user identification as well as an active user identification and the filter module intercepting the prevention of the storage or alteration of the data if the non-block user identification matches the active user identification.

Preferably the method comprises the step of logging activities of the filter module.

Reference will now be made, by way of example, to the accompanying drawings, in which:

Figure 1 shows a block diagram schematically illustrating components of a computer system which are of relevance for a protection against malware affection according to the invention;

Figure 2 shows a block diagram illustrating the architecture of an implementation of the inventive computer protection against malware affection;

Figure 3 is a block diagram illustrating operation of an embodiment of the present invention to block the writing of a certain type of file while allowing the writing of another type;

Figure 4 is an diagram for illustrating in more detail the processing of an I/O request in the kernel within the context of the present invention; and

Figure 5 shows a flow diagram of a program logic flow of an inventive computer protection against malware affection.

Before proceeding to a description of a specific embodiment of the present invention, an overview will first be provided. An embodiment of the present invention is founded on the basis that:

- All running processes on a computer are executable code, including malware;
- Executable code must be accessible on a computer to load and run, i.e. stored on disk; and therefore
- Malware cannot run or spread to other computers without execution code on a computer.

As mentioned above, the typical infection cycle of malware is (1) to cause its executable code to be stored on a computer using various means usually not in a user control or knowledge, (2) execute its code and (3) continue to spread its code to other computers, i.e. infection spreading. An embodiment of the present invention breaks malware infection by denying malware the ability to write its execution code to computers, which is critical in the malware life cycle. Malware typically exists in computer systems as a runnable program which is identified by its file extension. A few samples of executable file extensions on Microsoft® Windows® platforms are *.exe, *.dll, *.ocx *.vbs, and there are others. Robustly blocking these program files from being saved to a computer effectively stops malware infection.

The following uses a worm attack life cycle as an example to illustrate this in more detail.

1. A typical attack begins with what virus researchers called "warhead attack". A warhead is very small part of the malware code that exploits identified system vulnerabilities, e.g. buffer overrun attacks, to enable it to run on a computer. The warhead code is necessarily small with very limited functions; the infamous SQL Slammer has a warhead of 376 bytes.

2. The next step is to transfer the worm execution body code to the victim computer, i.e. saving a copy of the malware program file, and execute the code. This is known as the malware's "propagation phase". If successful, the malware seeks out other victims and continues its infection spread.

3. If a worm is not stopped, the rest of its cycle can be categorised as "target selection", where it locates other networked computers, "vulnerability scanning" to

identify other vulnerable computers, and "payload attack" when it performs the intended malicious actions on the victim computer.

An embodiment of the present invention intercepts malware's life cycle at the critical point of propagation, i.e. saving its execution code in a program file to permanent storage. Without propagation, the malware cannot run its payload and is effectively terminated. Under the Windows® operating system, all file saving operations require system kernel services and no programs, including malware, can write to hardware directly, because of Windows® operating system's design and internal protection.

An embodiment of the present invention proactively protects computer from malware infection without using conventional methods. In one mode of operating an embodiment of the present invention, all executable files are blocked from being written to a computer that have not had the user's explicit approval. The term "malware" is used to describe virus, worm, Trojan, spyware, adware, rootkits and all unwanted software applications.

On Microsoft® Windows® platforms, all application programs require its executable code (image file) to reside on the hard disk (storage media) in order that the system can load the image file to memory and run. From the operating system perspective, malware is just another form of application program, which must have its execution code residing on the system disk to carry out its 'unwanted action' from the user's perspective. Malware spreads infection by transferring its execution code to a user computer by tricks and exploiting vulnerabilities on the system, e.g. buffer overrun attacks or email attachment.

Without execution code on a computer, malware will not be able to infect a computer nor spread to other computers. An embodiment of the present invention indiscriminately denies writing of executable files to hardware storage, at the file system level such as NTFS and FAT. An embodiment of the present invention scrutinizes kernel instructions (rather than API calls) to the file system to write files. It has a deny-write-by-default policy, and a feature is built in to allow writing of executables if a user explicitly approves them. Such a feature is necessary for the user to install new software and patching system files.

An embodiment of the present invention differs from anti-virus, anti-spyware, rootkit detection tools in protection methods in the following respects:

- No pattern scanning – Anti-virus and anti-spyware products rely on scanning existing and incoming files against known malware signatures to detect virus/malware, e.g. McAfee®, Sophos®, and other major anti-virus products use this approach. Shortcomings are that this only works against known malware in the pattern database, and also degraded system performance.
- No behaviour anomaly detection – Used by most anti-spyware tools and some newer anti-virus products. They monitor and intercept 'unusual' API calls, and 'unexpected' application behaviours, and flagged detection when anomaly is found. False-positive ratio is high and it is relatively easy to bypass by malware.
- No system calls interception – Some tools intercept and manipulates system calls. They route API calls to their interceptor functions and manipulate the call by changing parameters or returning an execution failure status to the caller and not calling the requested API call, e.g. the FileMonster® application [see reference above and http://www.cigital.com/labs/projects/1021/]. This is not a Microsoft® approved approach.
- No system memory scanning – Some solutions scan system memory similar to file-based scan and monitor memory access trying to detect unusual usage, e.g. buffer overrun detection. This is not proven effective against malware and is not effective to "zero day" malware variants.

Another difference between an embodiment of the present invention and existing anti-malware technology is that an embodiment of the present invention is a malware prevention technology and not a detection and clean tool like typical anti-virus and anti-spyware software. Another technical difference, which will become more readily understandable following the description below, is that the filter driver used in an embodiment of the present invention cannot be shut down by malicious software, as seen in many anti-malware applications. An embodiment of the present invention is integrated into the file system and loaded automatically during system boot up sequence.

Interception in an embodiment of the present invention is possible by taking advantage of Windows self-protection architecture design. Briefly, Windows® enforces privilege-checking mechanisms to guarantee system integrity, and the concept of kernel mode (Ring 0), and user mode (Ring 3) on Intel x86 CPU. The operating system (OS) prevents direct access to all hardware devices, such as a hard disk, by a thin layer of system software called the Hardware Abstraction Layer (HAL). When Windows® is running, all 'write' accesses are provided by Kernel service calls. By integrating into the kernel, an embodiment of the present invention scrutinizes what files are allowed to be saved. This prevents email-borne, Internet-borne and user-induced virus files from reaching the protected computer.

Figure 1 provides an overview of components of a computer system that are of relevance for protection against malware affection according to an embodiment of the invention. A computer 1, e.g. a personal computer, a server, a personal digital assistant (PDA) or a cellular phone, has a CPU 10 (Central Processing Unit) and RAM 11 (Random Access Memory - a type of volatile computer memory that can be accessed randomly, meaning any byte of memory can be accessed without touching the preceding bytes) which are involved in all processing of the computer 1. It further has a data storage 157 (see Figure 2) comprising an internal data storage 12 and an external data storage 2 and RW Devices 13 (Read Write Devices). The internal data storage 12 usually comprises at least a hard disk on which data can be stored, changed and deleted. The external data storage 2 comprises on one hand removable data media such as for example floppy disks, CD-ROM (Compact Disks-Read-Only Memory - a type of optical disk), DVD (Digital Versatile Disks – another type of optical disk) memory token as e.g. memory sticks, or multi media devices, e.g. cameras or music players usually containing removable data storage media or hard disks themselves. On the other hand the external data storage 2 comprises remote data storage media such as for example shared network drives. The RW Devices 13 are capable of reading and/or writing to the external data storage 2 and summarize for example on one hand floppy disk drives, CD-ROM drives, DVD drives and ports for the multimedia devices or the memory token such as e.g. USB (Universal Serial Bus) or Firewire, for the use of the removable data storage media, and on the other hand network ports for the use of the remote data storage media. Connected to the computer 1 is a network LAN/WAN 3 (Local Area Network / Wide Area Network as e.g. the Internet) by a link 4 which can be either a wired link or a wireless link. The LAN/WAN 3 is usually used by malware to deploy and to get access to computer 1.

Typically the computer 1 is running an OS (Operating System – e.g. Microsoft® Windows®, UNIX, OS/2 or Linux) which is the system program responsible for the direct control and management of hardware including the data storage 157 (see Figure 2) and basic system operations, as well as running application software. The central module of the OS loaded first while booting the computer 1 and remaining in the RAM 11 usually is called kernel. Typically the kernel is responsible for memory management, process and task management, user interfacing, and disk management. Because it stays in RAM 11, it is important for the kernel to be as small as possible while still providing all the essential services required by other parts of the operating system and by application software.

Figure 2 illustrates one preferable architecture of an implementation of the inventive computer protection technique against malware affection in the computer 1. The implementation is separated by a boundary 14/15 in a upper user mode 14 and a lower kernel mode 15. The user mode 14 comprises a user application layer 142 which illustrates application programs accessible by a user. One user application in the user application layer 142 is a protected administration application 141 allowing a verified user to adjust and to administer the computer protection against malware affection. Between the kernel mode 15 and the user application layer 142 a subsystem layer 140 is arranged. In the subsystem layer 140 a protected subsystem such as e.g. Win32 for Windows® by Microsoft® Corporation or POSIX for UNIX and Linux is running providing interfaces to the user application layer 142, i.e. routines, protocols, and tools for building and running applications, and hiding the kernel mode layers to the user of the computer 1. The subsystem layer 140 passes input/output requests of the user application layer 142 to a I/O system service layer 150 (Input / Output) in the kernel mode 15 containing I/O system services. The I/O system services are provided by a I/O manager layer 151 lying under the I/O system service layer 150 which prepares input/output request packets corresponding to the input/output requests received by the subsystem layer 140 and which passes the input/output request packets to a file system filter driver 153 of a system driver layer 152. The system driver layer 152 comprises various registered system drivers containing said file system filter driver 153 as an implementation of a filter module and the file system drivers 154 such as e.g. NTFS, FAT or CDFS drivers. The file system filter driver 153 detects a format of data affected by the input/output request packets, blocks input/output request packets for data of a predefined format and passes

input/output request packets for data other than of the predefined format to the file system drivers 154. Under the system driver layer 152 the HAL 155 (Hardware Abstraction Layer) is situated comprising device drivers 156 which control the data storage 157. The file system drivers 154 pass the input/output request packets to the HAL 155 in which the storage of the data is completed. Besides the user mode 14 and besides the kernel mode 15 lies the registry 16 as a system database used to store configuration information in all layers described above.

For an embodiment of the present invention that relies on Windows® kernel drivers, and in particular file system filter drivers, an overview of these issues, and the concept of I/O services in the kernel in general, will now be provided with reference to Figures 1 and 2. Such a description will aid in an understanding of a Windows®-based embodiment of the present invention, and how it differs from the prior art, but it will be appreciated that the present invention is not limited to a Windows®-based environment (for example, on modern operating systems, e.g. Microsoft® Windows® and Unix®, software runs either in user mode, such as Outlook, or in kernel mode, such as services like network drivers and TCP/IP stacks).

An embodiment of the present invention will sometimes be referred to herein by the term "HDF", which stands for Hardware Disk Firewall, a function that is effectively performed by an embodiment of the present invention.

The term device driver refers to the software that controls the operation of a particular class of hardware devices, e.g. hard disk or tape drive. There are many classes of drivers, e.g. video drivers and network drivers [see http://www.microsoft.com/whdc/driver/foundation/DrvRoadmap.mspx]. The device drivers of interest in context of an embodiment of the present invention are file system drivers such as NTFS and FAT. User mode Win32 application programs do not run in the kernel space.

The Windows® architecture does not allow applications direct access to hardware, using the software protection layer called 'Hardware Abstraction Layer' (HAL), and it defines device-independent interfaces for applications to I/O operations on hardware devices. This abstraction enables applications to request I/O operations without regard to the physical characteristics of the underlying hardware devices.

Typical I/O operations on a hardware storage device are read and write requests. The kernel maintains a data structure called "services dispatch" tables. The tables map user mode function calls to Windows native APIs (Application Programming Interface), which are not fully documented. When an application makes a function call, the user mode function call (API) is mapped to the appropriate kernel native API call(s) and serviced by the kernel. For example, a user mode call to CreateFile(...) is mapped to kernel function call ZwCreateFile(...).

Some applications, for example the FileMonster® application, manipulate these undocumented dispatch tables ["Undocumented Windows 2000 Secrets: A Programmer's Cookbook" by Sven Schreiber, ISBN: 0201721872] and modify the entries to intercept system calls. Since Microsoft® has not documented nor recommended this approach, it is not considered good practice for system stability. For example, it two applications modify the dispatch tables, the system may not work correctly.

Windows provides I/O services by the I/O system services and I/O Manager kernel components. When an I/O request is received from the I/O system services component, the I/O Manager creates a data structure called an I/O Request Packet (IRP). The IRP is used to package and to dispatch the request and control messages to the target device driver via function codes. The function code is embedded in an IRP that a driver receives and acts on.

A driver implements and services function codes such as IRP_MJ_CREATE, IRP_MJ_CLOSE and IRP_MJ_DEVICE_CONTROL etc. and reports back status on completion. For example, when a Win32 application makes a CreateFile(...) call, the I/O Manager calls the target driver's IRP_MJ_CREATE function code. Later, when the application calls ReadFile(...) or WriteFile(...), the I/O Manager dispatches IRP_MJ_READ or IRP_MJ_WRITE calls in the IRP. Likewise, when the application closes the file handle, IRP_MJ_CLOSE is called. The I/O Manager also returns an I/O request result to the caller, e.g. succeeded or failed.

Windows has a layered-drivers architecture that allows other kernel drivers to intercept IRPs destined to a target device driver. The intended purpose is to provide additional services; these intercepting drivers are called file system filter drivers, or just filter drivers. The filter

driver used in an embodiment of the present invention will referred to herein as the "HDF" (filter) driver, or Hardware Disk Firewall filter driver, and is a file system filter driver that intercepts I/O requests to the file systems, e.g. NTFS.

In its initialization routine DriverEntry(...), the HDF driver informs the I/O Manager the I/O requests it wants to intercept, by registering the necessary I/O function codes, and calls IoAttachDeviceByPointer(...). The I/O Manager then dispatches IRPs, that are destined to the target drivers, first to the HDF filter driver for processing. HDF can request to receive only IRPs of interest but not all IRPs as explained here. The I/O Manager, on loading a driver, calls the "entry point" *DriverEntry()*,and passes in a pointer of a data structure (the driver's Device Object) together with a text string of its Registry path, which is not hard-coded so that HDF knows where to find its registry parameters. HDF performs initialization in *DriverEntry()* and fills out some, although not necessarily all, of the Device Object structure's data members before returning to the I/O Manager. One data member is an array of IRPs of interest, for example "*DriverObject->MajorFunction[IRP_MJ_CREATE] = HDFCreate;*". In this case, *HDFCreate()* is called when there is a IRP_MJ_CREATE; one or more of IRP_MJ_WRITE, IRP_MJ_READ, IRP_MJ_FILE_SYSTEM_CONTROL and IRP_MJ_DEVICE_CONTROL are also registered (it is useful to intercept _READ to protect the HDF application image files themselves from unauthorized access so that other processes are prevented from opening application files relating to HDF. The read and write IRPs can be used for the purpose of blocking executables with unconventional extensions as set out elsewhere in this description).

The HDF driver intercepts I/O requests to create a new file (IRP_MJ_CREATE). The interception allows the HDF driver to inspect the file extension of the impending file create operation without the need to scan the file data. To deny the creation of executable files, the HDF driver blocks all write requests of execution code on the system. This is achieved by returning a status 'STATUS_ACCESS_DENIED' to the I/O Manager, which then fails the request and informs the caller application, e.g. Internet Explorer.

This is illustrated in Figure 3, which shows the signal flow for two different attempted save operations: (a) an attempt to save a potentially harmful executable file "keylog.exe" to the storage medium 157; and (b) an attempt to save a harmless data file "business.doc" to the

storage medium 157. As illustrated, the HDF filter driver 153 blocks the write attempt for "keylog.exe", but not that for "business.doc".

(It should be noted that checking file-extension is a common approach to identify file types on Windows platforms. On a Unix-based computer, executable files are identified by file attributes, for example. The present invention is not to be understood as limited to any particular method of determining the file type on any particular operating system.)

The protection of existing executable files is a side-effect of using a technique embodying the present invention, for the following reasons. For a malware to modify an existing executable file, the creation of a new file is still required, since the modified file is written back to the storage as a new file. Therefore, an attempt to modify an existing executable file on the storage will be caught by checking for IRP_MJ_CREATE as set out above. There does not need to be a check as to whether a completely new file is being created and written, or whether it is an existing file that is being modified and then written back as a new file.

For any data that is written to storage, there is always an IRP_MJ_CREATE (to create a new file, whether it be for completely new data or for data modified from an existing file) followed by an IRP_MJ_WRITE (to actually write the data to the storage), arriving in different IRPs. If the file extension of a file to be created or modified is recognised as an executable, then the IRP_MJ_CREATE check will be sufficient to catch the malware at the time of requesting a new file, and this will block any attempt to create or modify the file.

However, in an attack scenario where malware tries to create or modify an executable file without an expected file extension, e.g. keylog.abc, then the file will pass the above IRP_MJ_CREATE check. To handle such a scenario, the HDF driver can inspect the impending file's data for the PE file header (for which, see below). This is desirable because the HDF driver would otherwise not deny the writing or modification of 'keylog.abc' because it has a non-executable file extension. For this purpose, the HDF driver can also intercept the write data request (IRP_MJ_WRITE). If the I/O request is detected as writing execution code, i.e. PE file structure, then the HDF driver would deny the I/O request by returning a status 'STATUS_ACCESS_DENIED' to the I/O Manager which then fails the request and informs the caller application, e.g. and Outlook® email client. With this method it is only

necessary to check the PE header, unlike known anti-virus tool which have to scan the entire file for a malware signature because all executables will have a PE header.

For the reasons mentioned above, an attempt by a trojan or malware to attach itself to an existing executable would be caught, since it needs to write/add itself to the target image file. HDF can block Alternative Data Stream (ADS) as well. If it is DLL-injection type of attack, then the malware does not need to change the target executable but just "injects" itself in the target memory space and piggybacks on the hijacked process, which performs the deeds of the malware. The attack is to make the target process load and run the attacker's DLL. However, the malware needs to transfer itself (execution body, as a DLL mostly) to the victim machine and cannot do it "over the web". HDF would not protect against memory-based attacks as such, but would stop the DLL being installed to the computer, which is essential for this attack to work.

Exceptions to blocking are provided to allow explicitly-approved write requests. Before returning a 'STATUS_ACCESS_DENIED', the HDF driver checks if the user has defined a rule to indicate that either (a) the requesting process is trusted or (b) the system account under which the request made is trusted. If there is an affirmative match, the HDF driver passes the I/O request to the destined device driver, e.g. NTFS, to perform the requested operation, via a call IoCallDriver(...) with the original IRP.

Under Microsoft® Windows® 32-bits operating systems, all native applications are Win32 executables. The operating system loads and runs execution code and does not differentiate if the code is legitimate or not, i.e. whether the user allows running of the specific code. For brevity, access control list (ACL) and user privileges will not be described here, which are largely irrelevant in the context of an embodiment of the present invention. With HDF, write-denial control would not typically be affected by the access control list (ACL) settings nor user privilege on the computer.

To run an application, the OS checks whether the execution code conforms to the Portable Executable File Format specification, e.g. PE format on all Windows 32-bit operation systems. This applies also to interpreted code, e.g. VB script, and Microsoft® specifies the interpreted code 'run-time-engine' must be in PE format.

The PE format is a Microsoft® specification that all PE files have a standard file structure with an "IMAGE_NT_HEADERS" signature, code and data sections. A PE file starts with two signature characters 'MZ' (called the DOS MZ header) and followed, after a DOS stub, by another PE signature with the characters 'PE\0\0'.

The OS recognizes an executable file by its file extension, and the loader ensures it is a valid PE file when loading the file and run the execution code. A requirement for execution code is that the executable file must be accessible on disk, local or remote, i.e. the execution code cannot reside on a remote Web server.

Checking the existence of the PE file format structure is a documented method by which the HDF driver can identify an executable file. This method is also used by other anti-malware applications for different purposes to HDF.

The NT kernel treats its internal entities as objects, these objects represent files, devices and so on. The kernel internal objects can be either named or unnamed. For named device drivers, the kernel maintains two lists of names in its internal object tree database. Each named device driver, e.g. the HDF filter driver, may have two names in the database. One is the driver's NT device name (Microsoft's terminology) intended for a driver to refer to the device it manages, and the other is a 'symbolic link' name. Of particular interest in an embodiment of the present invention is the HDF driver's symbolic link name, which is described as the driver's identifier for brevity here. It is noted that these internal names are not the driver image file name. For example, the HDF driver's NT device name might be "Device\HDFIoCtl", its symbolic link name might be "DosDevice\HDF" (default), and the image file filename might be "HDF.SYS" (which would generally not change).

Both the NT device name and symbolic link name are supplied by the device driver to the operating system kernel, typically during the driver's initialization routine. A device driver creates the NT device name by a kernel API call IoCreateDevice(...), and the symbolic link name by a call to IoCreateSymbolicLink(...).

The symbolic link name is for the sole purpose of making the driver visible to user mode applications. A user mode application requires a driver's symbolic link name in a function call to communicate with the driver. When an application requires to communicate with a device driver, it calls Win32 APIs CreateFile(...) passing in the driver's symbolic link name as a parameter, followed by DeviceIoControl(...) using the returned driver's handle as parameter. An application is not able to use the driver NT device name that is visible in the kernel space only.

To control computer storage devices, HDF is implemented at the operating system kernel level close to the hardware layer drivers. Technically, HDF is a "file system filter driver" in the Microsoft® operating system architecture. HDF intercepts 'write-file' instruction (not API calls) issued by the system to the file system drivers.

In the initialization routine, the HDF driver makes the function calls of IoRegisterFsRegistrationChange(...) and IoAttachDeviceByPointer(...). After the calls, the kernel I/O Manager routes I/O requests that bound for the target device to the HDF filter driver first (as set out above). From that point on, the HDF driver receives I/O requests to all storage devices on a system and performs write-denial control, until the system shuts down.

The processing of an I/O request in the kernel, and the role of HDF, is illustrated in more detail in Figure 4. In brief, the steps shown are:

1. A user mode API call gets mapped to a kernel call.
2. The I/O System Services serves the I/O and…
3. involves I/O Manager.
4. The I/O Manager creates a new IRP for that request.
5. The I/O Manager sends the IRP down the driver stack, where the HDF may intercept it.
6. The destined driver completes the IRP, if not blocked by HDF
7. The I/O request is completed.
8. The I/O status is copied to the subsystem address space.
9. The IRP is freed.
10. A handle and/or NT STATUS is returned.

As can be seen, HDF intercepts close to the hardware level. By contrast, an application such as FileMonster® intercepts much higher up, as marked in Figure 4. If a blocking occurs with HDF, steps 6 to 8 are skipped, with HDF returning the IRP to I/O Manager to perform step 9.

HDF intercepts e.g. the I/O Manager's "write-file" instruction and passes on other I/O instructions not relevant to HDF to the destined file system driver such as NTFS. On receiving a 'write-file' instruction contained in an IRP (described above), HDF checks and acts accordingly:

- If the write-file target is an executable not approved by the user – return a "STATUS_ACCESS_DENIED" status so that the I/O Manager informs the caller of failure. In this case, no executable file will be stored on the target storage device (HDF primary objective).
- If the write-file target is an executable approved by the user – no action by HDF and pass the IRP to the destined file system driver to carry out the writing.
- If the write target is not an executable, e.g. business.doc – no action by HDF and pass the IRP to the destined file system driver to carry out the writing.

HDF can determine if a write-file target is an executable file by checking:

- The file extension – from a Microsoft documented executable file extensions list, and;
- The executable file format signature called PE format, when the file extension is not covered above (as described above).

HDF can determine if a user approves writing of an executable by:

- A list of user-defined trusted processes that are allowed to write executable files – the default being none so that no process is allowed to write an executable file by default.

- A list of user-defined trusted system logon accounts (user and group account scope) that are allowed to write executable files – the default being none so that no account privilege is allowed to write an executable file by default.
- User action, i.e. user turns off HDF blocking via an administration application to install software; in one implementation the administration application is a standard Win32 command line application called HDFControl.exe and its GUI counterpart HDFMonitor.exe.

The HDF driver can obtain the above two lists (trusted processes and trusted accounts) from two sources:

- The Windows specified Registry entry for driver parameters at [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\driver_name\Parameters], where driver_name is, for example, "HDF", or
- Passed in as parameters by the administration applications at run time.

In general, an HDF implementation would not block writing of data or non-executable file types, although it is simple to reconfigure HDF to control writing of any file type in addition to executable file types.

The HDF driver's ability to block writing of executable files would not typically be affected by the requesting process's account privileges. Therefore, malware would not interfere with blocking even it has administrator or Local System elevated privileges obtained by successful attacks. However, to allow a user to install new software and system patching, HDF can provide an administration application for turning off blocking temporarily. An embodiment of the present invention can use a security method to ensure only the intended administration application is able turn off HDF blocking, and such a feature can be an important part of HDF application security.

Three such methods will be described below for authenticating the calling application. The first one is a completely new approach and the others involve using existing cryptography techniques in a new context to resolve authentication problems particularly associated within the context of an embodiment of the present invention.

Before a description of these authentication methods, the method by which a malicious process can attempt an attack will first be described in brief.

The administration application will be sending runtime instructions to the HDF driver, and in a secure approach the HDF driver would ensure that the instructions come from its administration application and not other malicious process. The threat comes from impersonation attacks because any application can send instructions to a driver if it knows the driver's identifier. Technically, the term driver's identifier refers the driver's internal 'symbolic link name', as described above.

The Microsoft® architecture enables an application to communicate with a kernel driver by calling two functions; CreateFile(...) and DeviceIoControl(...). One of the function parameters is the driver's identifier; in an HDF context this is "DosDevices\HDF". The driver provides its identifier to the operating system at runtime via a function call IoCreateSymbolicLink(...). Through this call, the system knows by name which driver to receive a calling application's instruction.

The risk to HDF is that a malicious process could establish an extra illegal communication path in parallel to the administration application and send instructions to the HDF filter driver. For this attack to work, the malicious code needs to know the HDF driver's identifier. Typically, the driver's identifier is hard-coded in the driver execution code and an attacker can discover it with relative ease, e.g. execution code disassembly. A runtime dynamic identifier thwarts this attack vector. Without the driver's identifier, a malicious process cannot establish communication nor send instructions. This forms the basis of the new HDF solution to mitigate the attack risk, and the other methods involving using cryptographic means to authenticate the communication parties. There is no known method for a driver to authenticate the calling application. It will be appreciated that the described methods can be used in combination to further improve application security.

The first approach described herein is to use a changing dynamic driver's identifier. The scheme is that the HDF driver changes its identifier during runtime via IoCreateSymbolicLink(...) calls, and informs its administration application of the newest identifier by a secure means. Since only the administration application knows HDF driver's

identifier at a specific point in time, only the administration application can communicate with and send instructions to the HDF driver. The change can be triggered by every instruction call or using a time-based scheme. If a malicious program uses an 'expired' identifier to send instructions to HDF driver, the system will not able to dispatch and abort the call. In essence, HDF protects its identifier from arbitrary application to authenticate communication.

By way of further explanation, the HDF driver has a default identifier with which the administration application can communicate for the first time. In the DeviceIoControl() call, the administration application passes to the driver a custom I/O control code, e.g. unblock flag or get configuration settings, and a user-space data buffer. The HDF driver performs the request function and if needed fills up the caller's data buffer. The driver can return the next identifier in the data buffer for the caller to use the next time – this is the "every instruction call" option mentioned above. With the "time-based" option mentioned above, the driver changes its identifier and the administration application makes the same change to the last used identifier independently, but according to some pre-arranged synchronization scheme. For example, the driver may append a number to the end of the identifier's fixed part, e.g. HDF, so that the new identifier would be HDF29 in a scheme where the minutes of the hour are used; 29 minutes passed the hour in this example. In this case the synchronization scheme is a shared secret.

The second and third approaches will now be described. Using cryptographic means to authenticate communication has been used in security applications, but is not known in the context of protecting the sending of instructions from an application to a device driver.

The second approach is to use a standard encryption algorithm such as Advance Encryption Standard (AES - FIP 197) (note: AES is just one of many algorithms that can be used; other examples are RSA, DES etc.) to encrypt the communication data between the administration application and the HDF driver. Because only the two software components know the data decryption key to decrypt data, the communication between the administration application and the driver is secured. To set up the authentication scheme, a standard method called Diffie-Hellman Key Agreement Protocol can be used so that the calling application and the

HDF driver each has a shared-secret key. A malicious process is not able to impersonate the calling application and send false instructions to the driver.

The third approach involves a cryptographic authentication method known as Challenge and Response. To set up the authentication scheme, the Diffie-Hellman Key Agreement Protocol can be used so that the calling application and the HDF driver each has a shared-secret key. As an example, authentication would work by the following steps:

1. The HDF driver sends a random number to the calling application for authentication, the Challenge;
2. The calling application uses the shared-secret key to encrypt the random number and returns the encrypted result to the HDF driver, the Response;
3. The HDF driver then uses its shared-secret key to encrypt the random number it generated before;
4. The HDF driver compares its own result and the calling application's Response;
5. The authentication succeeds if the two results match, because only the two components know the shared-secret key to compute the encrypted values.

Taking into account the above explanations, the steps performed by an embodiment of the present invention will now be described with reference to Figure 5. The program logic flow of an inventive computer protection against malware affection shown in Figure 5 illustrates in more detail an implementation of the file system filter driver 153 and the administration application 141 as described above. By way of example, the implementation is illustrated for a computer running a Microsoft® Windows® operating system.

In step S1, the file system filter driver 153 is registered with the operating system of computer 1 as a file in a kernel mode system driver. For example, the file system filter driver 153 "name.sys" (e.g. "name" here and below may be "hdf", so that the driver would be "hdf.sys") is registered by copying the program file preferably in a write protected manner to the according system folder, e.g. "WINDOWS\System32\Drivers\" (for Windows® XP), and the corresponding value is set in the registry 16, e.g. "[HKEY_LOCAL_MACHINES\ SYSTEM\CurrentControlSet\Services\name]".

In step S2, parameters used for adjusting the operation of the file system filter driver 153 are stored in a protected key of the registry 16, e.g. in the key "[HKEY_LOCAL_MACHINES\SYSTEM\CurrentControlSet\ Services\name\Parameters]". For example, following values are set in the protected registry key: "Active"=dword:00000001" as a flag with a predefined value for activating and deactivating the file system filter driver 153; "Extensions"=".COM; .EXE; .DLL; .BAT; .CMD; .OCX; .VBS; .PIF; .SCR; .CHM; .DRV; .HTA; .SHS; .WMA; .WSF; .WSH; .CPL; .CAB;" as a file extension list for predefining the data format types to be blocked by the file system filter driver 153; "Names" [or "Protected"]="name.SYS; name.LOG; name.EXE; HDFMONITOR.EXE;" as a blocking list of files which are always protected by the filter system 152 program file, the name of the filter system 152 program file's image files, the name(s) of an administration application 141 program file (there may be two such program names; e.g. hdfcontrol.exe and hdfmonitor.exe as set out elsewhere) and the name of a log file; "Sids"="" as an empty non-block user id list being ready to be adapted by the administration application 141 (to allow users or groups to write legitimately to storage, defaulted to empty, i.e. no user or group is allowed by default; it is possible to separate the user ids from the group ids by specifying the former with a "UserSids" parameter and the latter with the "Sids" parameter); "LogPath"="C:\nameLog" as a path definition where to store the log file reporting at least all blocking activities of the file system filter driver 153; and "AllowedProcesses"="" provides a list of process names that the user allows explicitly to write executable files; this defaults to empty but can be changed by the user with the administration application 141. To provide maximum flexibility, HDF parameters can also be changed during runtime. However, with HDF it is not necessary to write to the Registry during runtime, unlike for some other existing applications.

Step S1 and step S2 are part of an Initialization 5 which is conducted at least once during installation of the computer protection against malware affection either by an installation program of the computer program product according to the invention or by a system administrator.

In step S3, the file system filter driver 153 is launched either on user request by the administration application 141 or during each booting process of the computer 1.

In step S4, the file system filter driver 153 reads the parameters stored in the registry 16 as mentioned above.

In step S5, the file system filter driver 153 creates a randomly generated dynamic name, registers the name with the operating system and stores the name in the registry.

In step S6, the file system filter driver 153 receives input/output request packets from the input/output manager 151 and performs the following:

Step S64, checking if a file name of the data file is part of the blocking list stored in the registry 16. If the file name of the data file matches a name of the blocking list, the file system filter driver 153 continues with step S7, otherwise it continues with step S60.

Step S60, checking whether the flag for activating and deactivating the file system filter driver 153 is set to the predefined value. If the flag is set to the predefined value, the file system filter driver 153 continues with step S61, otherwise it continues with step S8.

Step S61, checking whether input/output request packets received is related to a write access request. For example, the file system filter driver 153 checks whether the input/output request packet includes a function code as e.g. "IRP_MJ_CREATE". If the input/output request packets are related to a write access request, the file system filter driver 153 continues with step S62, otherwise it continues with step S8.

Step S62, checking whether the format of the data file associated with the write access request is of the predefined format type. Preferably, the checking of the format is performed by comparing a file extension of the data file associated with the write access request to the file extension list stored in the registry 16. If the data file associated with the write access request is detected to be of the predefined format type the file system filter driver 153 continues with step S63, otherwise it continues with step S8.

Step S63, comparing the user identification information associated with the write access request to the non-block user id list stored in the registry 16. If the user id matches a user or a

user group of the non-block user id list the file system filter driver 153 continues with step S8, otherwise it continues with step S7.

In step S7, the file system filter driver 153 blocks the write access request, informs the I/O manager 151 about the blocking, and logs the blocking activity in the log file.

In step S8, the file system filter driver 153 passes the input/output request packets to a subsequent system driver such as the file system driver 154.

Other alternative embodiments of the computer protection against malware affection according to the invention are realizable. Explicitly mentioned in this context are:

- The file system filter driver can be placed in the kernel below the mounted file system divers, and above the lower level hardware driver for the storage media.
- The data format of a data file can be detected by the file system filter driver reading the file header of the data file.
- Preferably the file extension list for predefining the data format is stored on one hand in the registry and on the other hand in a protected binary data file. The adaptations made by the administration application affect the file extension list in both locations. The file system filter driver reads the file extensions from both locations and produces an error if they don't match. Like this additional security can be implemented into the computer protection against malware affection.

An embodiment of the present invention has one or more of the following technical benefits:

- It can load and run as part of the operating system such that it cannot be disabled, i.e. during system boot up and before any user logging in.
- It can resist hacking and attack attempts to shut down operation, e.g. by hostile malware or user mistakes.
- It is able to identify and block executable files without pattern matching.
- It can function autonomously irrespective to system privileges of the user account, caller processes and threads.
- It can function transparently to users and applications. All applications can run as

normal. There is no requirement for pop-up dialogs or other user alerts during operation.

- It supports local, networked and removable storage devices, and plug-and-play mounted devices are protected automatically.
- Operations, whether blocking or non-blocking (pass through), can be logged for audit trail purposes.
- It has a small application footprint, and does not degrade performance.
- It is compatible with existing anti-virus programs, personal firewalls and scanning tools.
- It does not require maintenance, and has reliable operation.

An embodiment of the present invention:

- Is able to prevents malware infection against a computer using an innovative concept of blocking all executable code entering a computer. This new approach resolves the shortcomings of current technologies used by antivirus, anti-spyware and rootkit detection solutions. It is a new malware-prevention method, rather than a malware-detection approach.
- Is able to protect against the so called "zero-day" malware and recent rootkit spyware without false-positive, with a single tool. "Zero-day" malware refers to malicious code that anti-malware vendors have not yet identified or not released a pattern signature of the malware. Rootkits are malicious software that renders itself undetectable on infected systems.

Although the above description has focussed on the Windows family of operating systems, it will be appreciated that an embodiment of the present invention applies equally and correspondingly to other types of operating systems that support an architecture having a protected kernel environment, e.g. Unix, Linux and Mac OS. The skilled person will readily appreciate that certain components and methods that are specific to Windows® will have corresponding components and methods in other environments. As one example, using the Registry as set out above to store HDF parameters is Windows-specific; on other environments the parameters would be stored by some other means, e.g. file-based or over the network, e.g. using a service/daemon. In addition, use herein of terms such as "input/output

manager", "file system filter driver" and "I/O Request Packets" (in particular their use in the appended claims) is not intended to imply any limitation to a Windows® type of operating system; instead these terms are to be understood as applying to the equivalent part or parts and the equivalent messages in any other type of operating system or environment, whatever terms are normally used when describing that operating system or environment.

With Linux, for example, user applications generally cannot access the hardware directly. The Linux operating system relies on a kernel software comprising various device driver types to provide user applications input/output services to hardware, e.g. ISO 9660 filesystem for CD-ROMs, SMB filesystem for Microsoft Windows servers. Device drivers are developed to integrate into kernel core software or developed as 'loadable device drivers' modules, which are kernel components but not integrated into the kernel core. 'Loadable device drivers' typically load into memory during boot-time. Device drivers provide input/output access to the hardware to which generally user applications cannot access direct.

An embodiment of the present invention can be developed to a Linux filter device drivers and adapted to intercept input/output requests to hardware from user applications. In the same manner as described above, an embodiment of the present invention when implanted with on a computer under the Linux operating system will therefore follow the policy of denying writing to hardware storage when the data format is determined as execution code, for example, by examining the file's "execute permissions" attributes, because Linux executable code depend not on a file extension but on the file permissions attributes.

The current operating system for the Macintosh is Mac OS X that is built on top of the Berkeley Unix implementation (FreeBSD 4.4) and Darwin. The OS X kernel is called XNU, which is a modern operating system with a user space and kernel core architecture. The kernel provides mechanisms and interfaces for user applications to make input/output requests to hardware devices through a kernel software called "I/O Kit". The I/O Kit component is the kernel device driver framework and includes numerous device driver types for e.g. hard disks and other storage devices.

Mac OS X, like other OSs, uses files and directories to store data and execution code, and identifies file types by file extensions. Executable files are organized into a directory together

with its resource files (unlike Windows® executables which store execution code and resources in the same file, *.EXE), and is called an "application bundle". The I/O Kit framework supports driver stacking which is a similar concept to Windows layered-drivers architecture as described above. The driver stacking features allow a skilled person to develop a filtering module to intercept writing of executables to storage devices following the same principles as set out above in the main description of an embodiment of the present invention to achieve protection from malware infection.

The skilled person will readily appreciate how to adapt the present invention for use with other such types of operating system, and operating systems for other types of computing devices such as handheld devices, cell phones, Smartphones, PocketPCs, Personal Digital Assistants (PDAs), Blackberry® devices, TV set-top boxes and any future such devices having a computing (CPU) unit and storage. The terms "computer" and "computer program" in the appended claims are intended to cover such devices.

Although the above description has referred to a program stored on a computer-readable medium, it will be appreciated that a computer program embodying the present invention need not be stored on a computer-readable medium and could, for example, be embodied in a signal such as a downloadable data signal provided from an Internet website. The appended claims are to be interpreted as covering a computer program by itself, or as a record on a carrier, or as a signal, or in any other form.

Claims

1. A method of protecting a computer (1) against malware affection, the computer (1) having a data storage (157, 12, 2) and an operating system for managing the data storage (157, 12, 2), the method comprising providing a filter module in the operating system which operates to detect an attempt to store data in the data storage, to determine a data format of the data to be stored in the data storage (157, 12, 2), and to prevent storage of the data if the data format is determined to relate to a predefined type.

2. A method as claimed in claim 1, wherein the filter module is provided as a file system filter driver (153) in a kernel (15) of the operating system.

3. A method as claimed in claim 1 or 2, wherein the filter module is arranged to operate between an input/output manager (151) of the operating system and a driver (154, 156) associated with the data storage (157, 12, 2).

4. A method as claimed in claim 3, wherein the input/output manager (151) and driver associated with the data storage form part of a kernel (15) of the operating system.

5. A method as claimed in claim 4, when dependent on claim 2, further comprising the file system filter driver (153) receiving at least write access calls from the input/output manager (151) intended for the kernel driver associated with the data storage, blocking such write access calls relating to data of the format of the predefined type, and such passing write access calls not so relating.

6. A method as claimed in claim 5, further comprising informing the input/output manager (151) about the blocking and optionally passing of such write access calls.

7. A method as claimed in any one of claims 3 to 6, comprising arranging for I/O Request Packets that are sent from the input/output manager (151) and intended for the driver associated with the storage (157, 12, 2) to be re-routed via the filter module, the re-routed I/O Request Packets being at least those of a type enabling the filter module to detect an attempt to store data in the data storage.

8. A method as claimed in claim 7, comprising querying a function code of a re-routed I/O Request Packet at the filter module to detect an attempt to store data in the data storage

9. A method as claimed in claim 7 or 8, wherein the re-routed I/O Request Packets comprise those relating to the creation of a new file in the data storage.

10. A method as claimed in claim 9, wherein the re-routed I/O Request Packets comprise those having or associated with an IRP_MJ_CREATE function code.

11. A method as claimed in any one of claims 7 to 10, wherein the re-routed I/O Request Packets comprise those relating to the writing to an existing file on the storage.

12. A method as claimed in claim 11, wherein the existing file is a recently-created file.

13. A method as claimed in claim 11 or 12, wherein the re-routed I/O Request Packets comprise those having or associated with an IRP_MJ_WRITE function code.

14. A method as claimed in any one of claims 7 to 13, comprising blocking a re-routed I/O Request Packet at the filter module if it is determined that the format of the data to be stored by way of that I/O Request packet is of the predefined type.

15. A method as claimed in claim 14, when dependent on claim 9, comprising determining the format of the data to be stored by examining the intended file extension of the file to be created.

16. A method as claimed in claim 14 or 15, when dependent on claim 11, comprising determining the format of the data to be stored by examining a file header of the file to be written.

17. A method as claimed in claim 16, wherein the file header is the Portable Executable File Format file header.

18. A method as claimed in any preceding claim, comprising referring to at least one parameter associated with the filter module preferably stored in a protected manner in the data

storage (157, 12, 2), and determining the behaviour of the filter module at least in part in dependence on the at least one parameter.

19. A method as claimed in claim 18, comprising running a protected administration module (141) which allows a verified user to adapt the at least one parameter.

20. A method as claimed in claim 18 or 19, further comprising storing the at least one parameter in a protected part of a system database (16) of the operating system.

21. A method as claimed in claim 20, wherein the system database comprises the Registry.

22. A method as claimed in any one of claims 18 to 21, comprising reading the at least one parameter during the booting of the computer (1).

23. A method as claimed in any one of claims 18 to 22, wherein the at least one parameter comprises a predefined blocking file extension as the predefined type, and comprising the filter module detecting the format of the data to be stored by comparing a specific file extension of the data to the predefined blocking file extension.

24. A method as claimed in any one of claims 18 to 23, wherein the at least one parameter comprises a flag, wherein the flag is able to be set to a predefined status, and comprising the filter module checking the flag and preventing storage of the data if the flag is set to the predefined status.

25. A method as claimed in any one of claims 18 to 24, wherein the at least one parameter comprises a non-block user identification, and comprising the filter module checking the non-block user identification as well as an active user identification and allowing the storage of the data if the non-block user identification matches the active user identification.

26. A method as claimed in any preceding claim, comprising providing an administration application (141) to allow a user to alter the behaviour of the filter module by sending the filter module runtime instructions.

27. A method as claimed in claim 26, comprising using the administration application to turn off blocking temporarily.

28. A method as claimed in claim 26 or 27, when dependent on claim 18, wherein the administration application is operable to alter the behaviour of the filter module by changing, or causing the filter module to change, at least one of the at least one parameter.

29. A method as claimed in claim 26, 27 or 28, comprising authenticating the administration application to the filter module.

30. A method as claimed in claim 29, comprising the filter module storing a name or identifier of the filter module in a protected manner in the data storage (157, 12, 2) in such a way that only the administration application can reasonably learn of the name or identifier and thereby communicate with the filter module.

31. A method as claimed in claim 30, comprising registering the name in the operating system.

32. A method as claimed in claim 30 or 31, wherein the name or identifier of the filter module is stored in a protected manner in the data storage by changing it from time to time during runtime.

33. A method as claimed in any one of claims 29 to 32, comprising authenticating the administration application to the filter module using a cryptographic authentication method.

34. A method as claimed in any preceding claim, further comprising logging activities of the filter module.

35. A method as claimed in any preceding claim, wherein the predefined type of data comprises executable data.

36. A method as claimed in any preceding claim, wherein the filter module operates to prevent storage of such data to any part of the storage.

37. A method as claimed in any preceding claim, wherein the operating system is one of the Windows® family of operating systems.

38. A method of thwarting malware at its propagation phase by performing a method as claimed in any preceding claim so as to block any unauthorised attempt to write any data to the data storage that could potentially constitute malware, thereby preventing the malware from propagating.

39. A method of administrating a computer having a kernel driver and an administration application that communicates with the kernel driver during runtime, comprising authenticating the administration application to the kernel driver.

40. A method as claimed in claim 39, comprising using a cryptographic method to authenticate the administration application to the kernel driver.

41. A method of authenticating an administration application to a kernel driver for use in a method as claimed in claim 39 or 40, comprising using a dynamic driver identifier that is made to change during runtime in a way such that only the administration application can reasonably learn of the current driver identifier and thereby communicate with the kernel driver.

42. A method as claimed in claim 39, 40 or 41, wherein the kernel driver is a file system filter driver.

43. A computer comprising a data storage (157, 12, 2), an operating system for managing the data storage (157, 12, 2), and a filter module provided in the operating system for protecting the computer against malware affection, the filter module being adapted to detect an attempt to store data in the data storage, to determine a data format of the data to be stored in the data storage (157, 12, 2), and to prevent storage of the data if the data format is determined to relate to a predefined type.

44. A computer program which, when run on a computer, causes the computer to carry out a method as claimed in any one of claims 1 to 42.

45. A computer program which, when loaded into a computer, causes the computer to become one as claimed in claim 43.

46. A computer program as claimed in claim 44 or 45, carried on a carrier medium.

47. A computer program as claimed in claim 46, wherein the carrier medium is a transmission medium.

48. A computer program as claimed in claim 46, wherein the carrier medium is a storage medium.

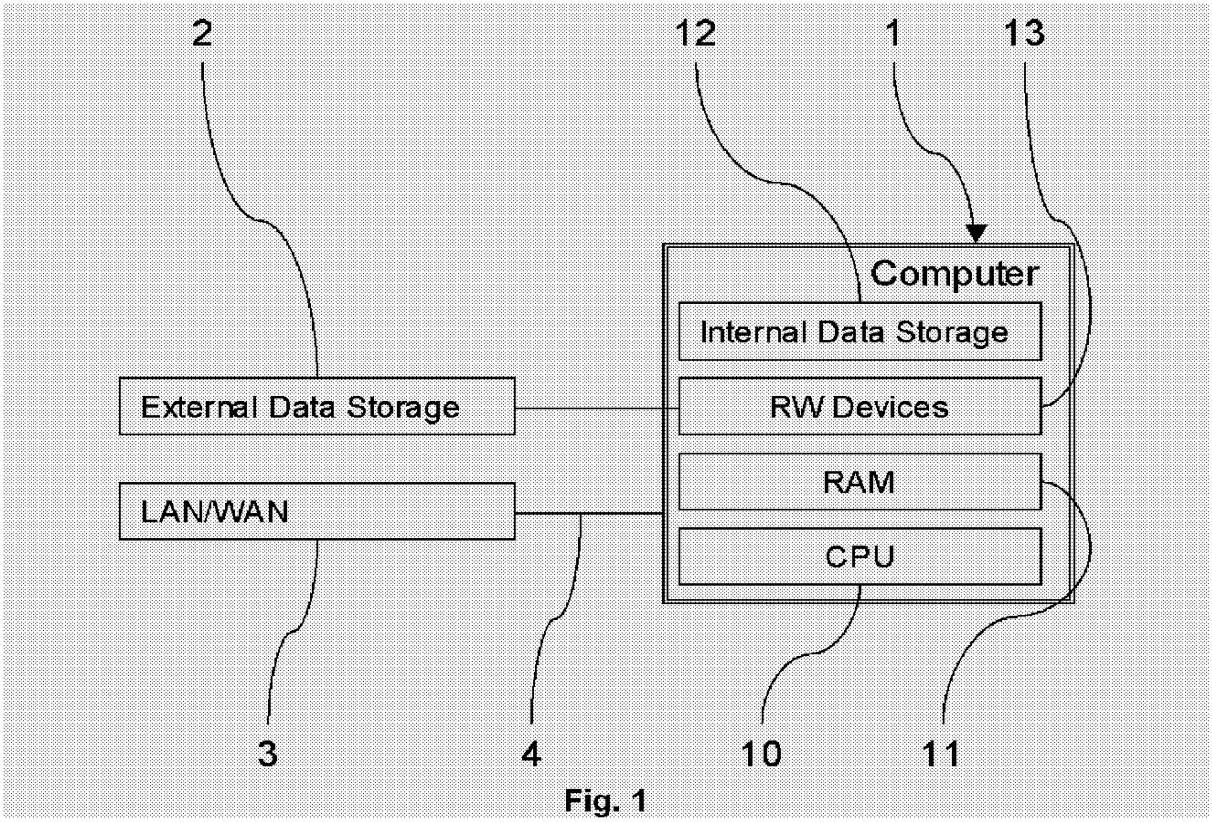

2
12
1
13
Computer
Internal Data Storage
External Data Storage
RW Devices
RAM
LAN/WAN
CPU
3
4
10
11

Fig. 1

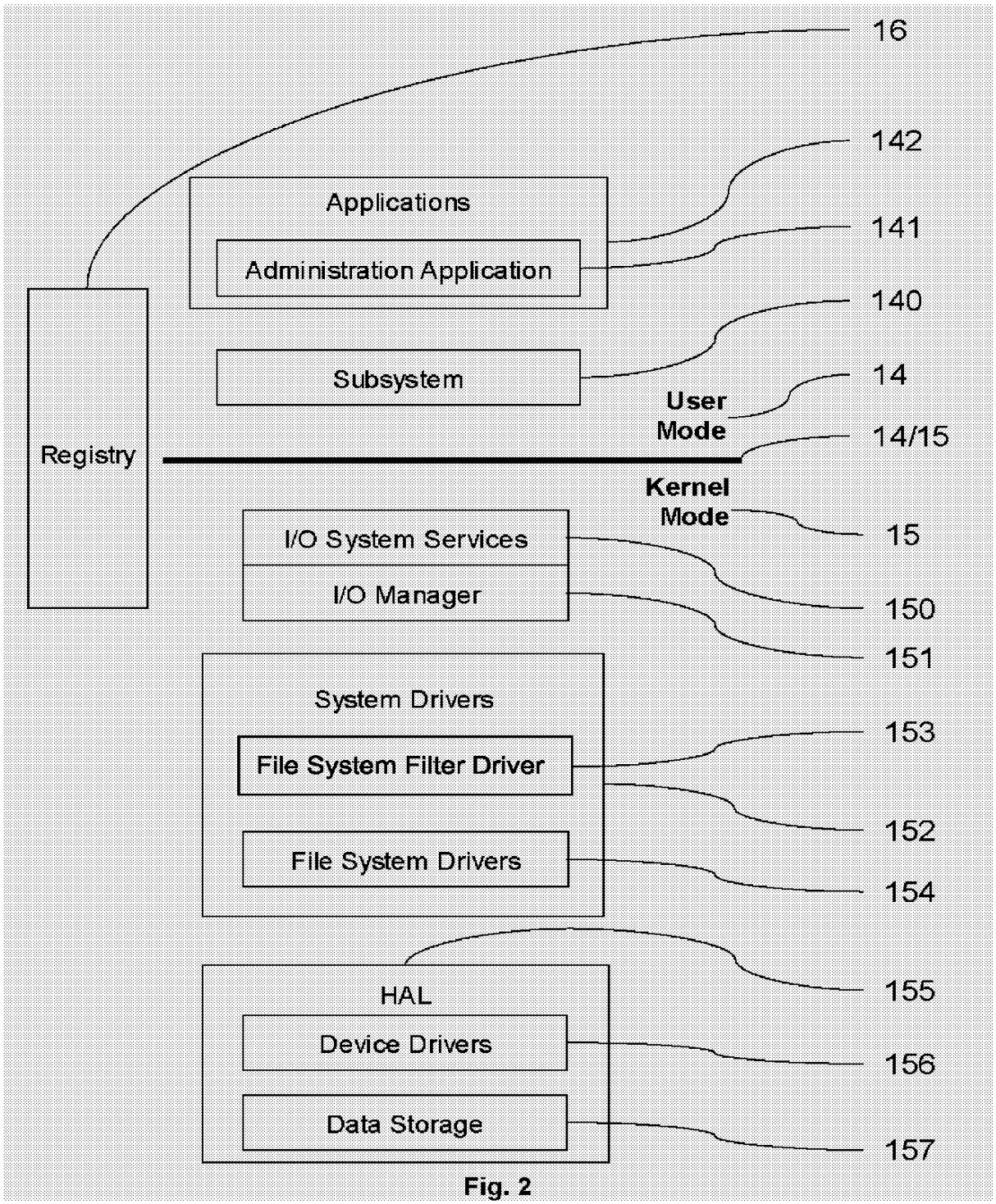

16
142
Applications
141
Administration Application
140
Subsystem
14
User Mode
14/15
Registry
Kernel Mode
15
I/O System Services
150
I/O Manager
151
System Drivers
153
File System Filter Driver
152
File System Drivers
154
155
HAL
Device Drivers
156
Data Storage
157

Fig. 2

FIG. 3

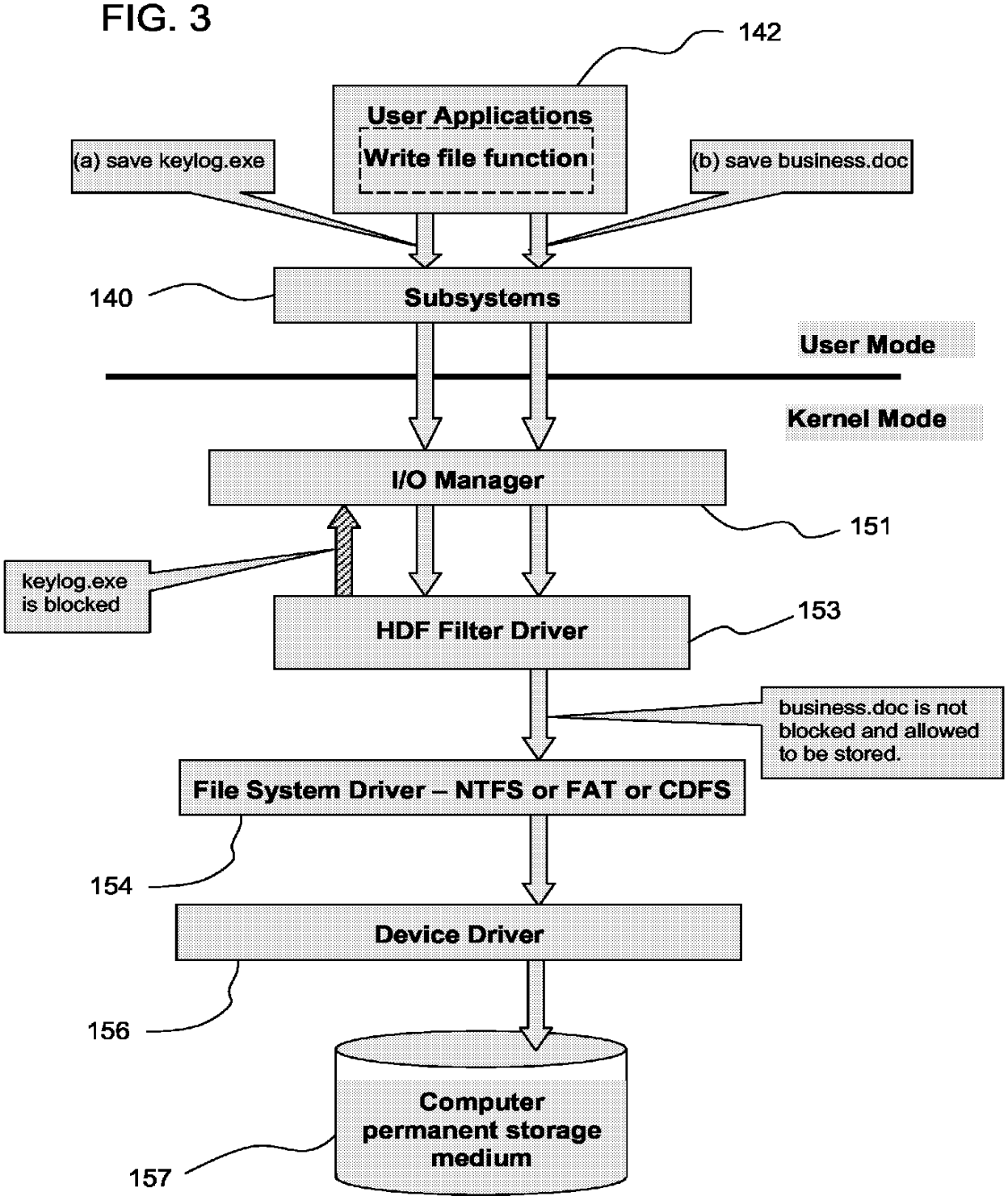
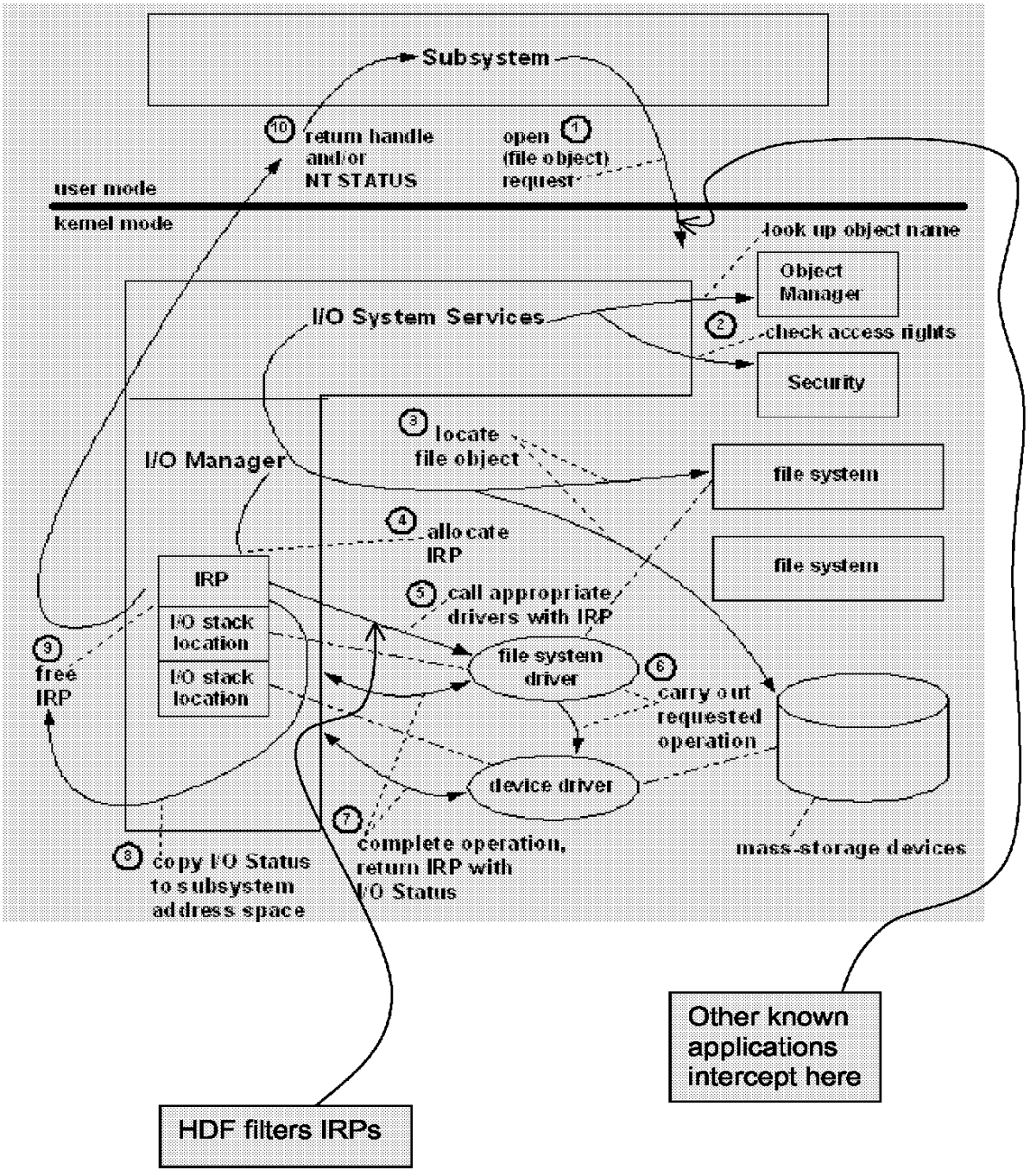

142
User Applications
Write file function
(a) save keylog.exe
(b) save business.doc
140
Subsystems
User Mode
Kernel Mode
I/O Manager
151
keylog.exe
is blocked
HDF Filter Driver
153
business.doc is not
blocked and allowed
to be stored.
File System Driver – NTFS or FAT or CDFS
154
Device Driver
156
Computer
permanent storage
medium
157

FIG. 4


Subsystem
10 return handle and/or NT STATUS
open (file object) request 1
user mode
kernel mode
look up object name
Object Manager
I/O System Services
2 check access rights
Security
3 locate file object
I/O Manager
file system
4 allocate IRP
file system
IRP
I/O stack location
I/O stack location
5 call appropriate drivers with IRP
9 free IRP
file system driver
6 carry out requested operation
device driver
7 complete operation, return IRP with I/O Status
8 copy I/O Status to subsystem address space
mass-storage devices
Other known applications intercept here
HDF filters IRPs

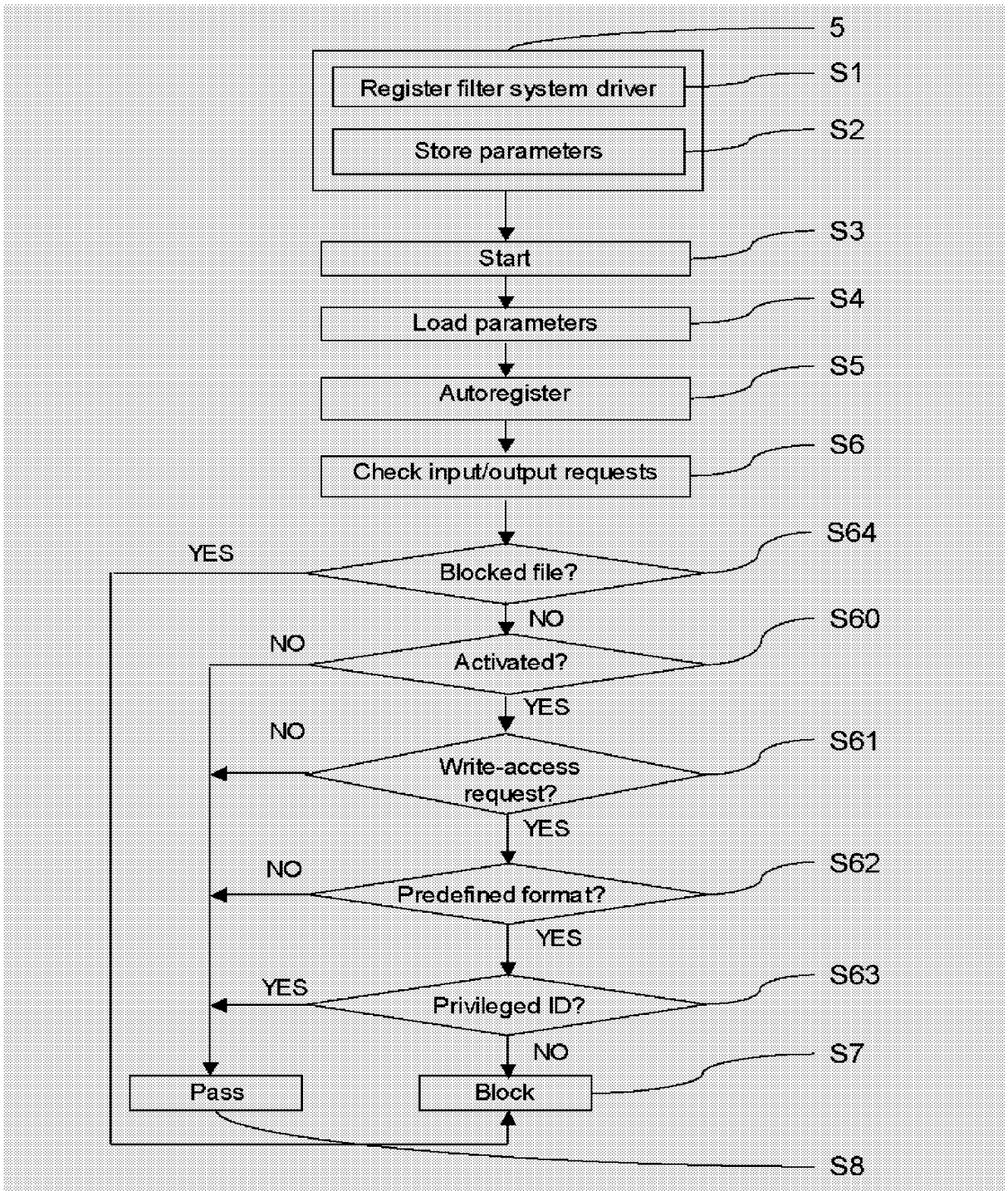

5
Register filter system driver
S1
Store parameters
S2
Start
S3
Load parameters
S4
Autoregister
S5
Check input/output requests
S6
Blocked file?
S64
YES
NO
Activated?
S60
NO
YES
Write-access request?
S61
NO
YES
Predefined format?
S62
NO
YES
Privileged ID?
S63
YES
NO
Pass
Block
S7
S8

FIG. 5

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	105	kernel AND rootkit	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 12:54
S2	3	S1 AND "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 12:55
S3	16	S1 AND @pd < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 12:56
S4	17	kernel AND rootkit SAME signature	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 13:45
S5	68	kernel AND rootkit AND signature	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 13:45
S6	13	S5 AND @pd < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 13:46
S7	36	S5 AND @ad < "20051005"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/02 16:04
S8	0	driver SAME kernel SAME (detector	US-PGPUB;	ADJ	ON	2009/02/03

		OR monitor OR inspec$5) AND rootkey	USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			19:09
S9	690	driver SAME kernel SAME (detector OR monitor OR inspec$5)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/03 19:09
S10	523	S9 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/03 19:10
S11	34	(device driver) AND rootkit	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 11:11
S12	8	S11 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 11:11
S13	8	S11 AND @rlad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:10
S14	3	kernel NEAR2 detector SAME driver AND (malicious OR rootkit OR infect$3) NEAR2 driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:32
S15	4	kernel level detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:38
S16	5	kernel detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	ADJ	ON	2009/02/04 12:38

			IBM_TDB			
S17	2	(detector NEAR3 driver) AND kernel AND (malicious OR infect$4) driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:39
S18	3	(detector) AND kernel AND (malicious OR infect$4) driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:40
S19	4	detector AND kernel AND (malicious OR infect$4) (driver OR dll)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:41
S20	35	detector AND kernel AND ((malicious OR infect$4) (driver OR dll) OR rootkit)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:46
S21	20	S20 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 12:46
S22	145	(kernel level driver) OR (detector NEAR2 driver) AND kernel	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:05
S23	105	S22 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:05
S24	145	(kernel level driver) OR (detector NEAR2 driver) AND kernel AND detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:08
S25	105	S24 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS;	ADJ	ON	2009/02/04 13:08

			EPO; JPO; DERWENT; IBM_TDB			
S26	56	((kernel level driver) OR (detector NEAR2 driver)) AND kernel AND detector	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:10
S27	50	S26 AND @ad < "20051004"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:10
S28	2	S26 AND @ad < "20051004" AND (malware OR virus OR rootkit OR worm)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:14
S29	94	kernel level driver	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 13:49
S30	0	("20070055711")".pn"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 14:41
S31	2	("20070055711").pn.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/04 14:41
S32	1	("20070055711").pn.	US-PGPUB; USPAT	ADJ	ON	2009/02/04 16:44
S33	1	("20070055711").pn. AND signature	US-PGPUB; USPAT	ADJ	ON	2009/02/04 17:49
S34	1	("20040049693").pn.	US-PGPUB; USPAT	ADJ	ON	2009/02/05 09:18
S35	1	("7398389").pn.	US-PGPUB; USPAT	ADJ	ON	2009/02/05 09:20
S36	14844	(380/276,277,278,284,286 OR 713/182,183,184,185,176 OR 726/28,30,33 OR 714/22-30).CCLS.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/02/05 11:30

S37	38	paul NEAR gassoway	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/13 15:28
S38	24	kernel AND (rootkit OR malware OR virus OR worm OR pestware) AND boot SAME (initialize OR start OR bootup) AND kernel WITH monitor	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:02
S39	14	S38 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:02
S40	83	kernel AND (rootkit OR malware OR virus OR worm OR pestware) AND boot$3 SAME (initializ$5 OR start$3 OR bootup) AND kernel WITH monitor$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:09
S41	55	S40 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 10:09
S42	762	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR "20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:39
S43	58	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:39

		"20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433").PN.				
S44	3	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR "20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433").PN. AND kernel AND boot$3 AND (malware OR virus OR worm OR pestware)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:41
S45	6	("5623600" OR "6069628" OR "6073241" OR "6092194" OR "6154844" OR "6167520" OR "6310630" OR "6397264" OR "6460060" OR "6480962" OR "6535931" OR "6611878" OR "6633835" OR "6667751" OR "6701441" OR "6785732" OR "6804780" OR "6813711" OR "6829654" OR "6965968" OR "7058822" OR "20030217287" OR "20040030914" OR "20040034794" OR "20040064736" OR "20040080529" OR "20040143763" OR "20040187023" OR "20040225877" OR "20050138433" OR "20020129277" OR "20030159070" OR "20040225877" OR "20050120242" OR "20050149726" OR "20050172115" OR "20050188272" OR "20050257266" OR "2006026576").PN. AND kernel AND boot$3 AND (malware OR virus OR worm OR pestware)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 13:46
S46	42	S40 AND @ad < "20060801" AND @pd < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:47
S47	37986112	S40 AND @ad < "20060801" OR @pd < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	ADJ	ON	2009/08/20 14:47

			IBM_TDB			
S48	55	S40 AND (@ad < "20060801" OR @pd < "20060801")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:47
S49	8349	S40 AND @ad < "20060801" AND (boot$3 WITH before) OR pre-boot$3 OR (prior WITH boot$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:50
S50	14	S40 AND @ad < "20060801" AND ((boot$3 WITH before) OR pre-boot$3 OR (prior WITH boot$3))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/20 14:51
S51	180	kernel AND boot AND Win32 AND (scan$4 OR monitor$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:09
S52	158	S51 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:10
S53	94	S51 AND @ad < "20060801" AND registry	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:11
S54	12	kernel AND boot AND Win32 AND (scan$4 OR monitor$3) WITH registry	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:13
S55	8	S54 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 10:14
S56	1	("20050125687").pn.	US-PGPUB; USPAT	ADJ	ON	2009/08/24 13:57
S57	1	("20050125687").pn. AND log	US-PGPUB;	ADJ	ON	2009/08/24

			USPAT			13:57
S58	18	kernel AND boot AND (change OR modify OR update) WITH (rule$2 OR polic$3) SAME (read WITH log)	US-PGPUB; USPAT	ADJ	ON	2009/08/24 14:28
S59	18	kernel AND boot AND (change OR modify OR update) WITH (rule$2 OR polic$3) SAME (read WITH log)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 14:32
S60	51	kernel AND boot AND (change OR modify OR update) WITH (rule$2 OR polic$3) AND (read WITH log)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 14:38
S61	26	S60 AND @ad < "20060801"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	ADJ	ON	2009/08/24 14:38
S62	1	("20030135791").pn.	US-PGPUB; USPAT	ADJ	ON	2009/08/25 12:03
S63	71	(713/164 OR 726/2,4,23-25) AND kernel AND (malware OR virus OR worm) AND boot$3	US-PGPUB; USPAT	ADJ	ON	2009/08/25 12:55
S64	7	jerome NEAR schneider	US-PGPUB; USPAT	ADJ	ON	2009/08/25 12:58
S65	331	(system OR device) driver SAME (malware OR pestware OR virus)	US-PGPUB; USPAT	ADJ	ON	2011/04/09 10:44
S66	209	S65 AND @AD < "20060801"	US-PGPUB; USPAT	ADJ	ON	2011/04/09 10:44
S67	85	S65 AND @AD < "20060801" AND kernel	US-PGPUB; USPAT	ADJ	ON	2011/04/09 10:45
S68	41	S65 AND @AD < "20060801" AND kernel AND boot	US-PGPUB; USPAT	ADJ	ON	2011/04/09 11:05
S69	1	native NEAR2 scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT	ADJ	ON	2011/04/09 11:59
S70	1	native NEAR2 scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2011/04/09 12:00
S71	433	scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2011/04/09 12:01
S72	16	boot With scan$4 SAME registry AND (malware OR pestware OR virus)	US-PGPUB; USPAT; FPRS; EPO	ADJ	ON	2011/04/09 12:20
S73	11	S72 AND @AD < "20060801"	US-PGPUB; USPAT	ADJ	ON	2011/04/09 12:20
S76	2	("8190868").PN.	US-PGPUB;	ADJ	ON	2013/10/04

			USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			16:03

EAST Search History (Interference)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S74	0	(713/2,164,193).CCLS. AND ((pestware AND boot AND sequence AND period AND drivers AND initialized AND kernel-level AND monitor AND boot AND managing AND related AND before AND native AND application AND loaded AND subsystem AND scanning AND scanner AND registry AND behavior AND rules) OR (launching AND operating AND system AND pestware AND management AND behavior AND rules AND portion AND compiled AND engine) OR (generating AND response AND monitoring AND record AND events AND pestware-related AND analyzing AND identify AND behavior AND rules AND prevent)).CLM.	USPAT; UPAD	ADJ	ON	2012/01/14 20:55
S75	0	(726/4,22-27).CCLS. AND ((pestware AND boot AND sequence AND period AND drivers AND initialized AND kernel-level AND monitor AND boot AND managing AND related AND before AND native AND application AND loaded AND subsystem AND scanning AND scanner AND registry AND behavior AND rules) OR (launching AND operating AND system AND pestware AND management AND behavior AND rules AND portion AND compiled AND engine) OR (generating AND response AND monitoring AND record AND events AND pestware-related AND analyzing AND identify AND behavior AND rules AND prevent)).CLM.	USPAT; UPAD	ADJ	ON	2012/01/14 20:56

10/7/2013 11:03:58 AM
C:\Users\glane\Documents\EAST\Workspaces\case_13460655.wsp

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 9502

SERIAL NUMBER	FILING or 371(c) DATE RULE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
13/460,655	04/30/2012	713	2438	6890-45-CON

APPLICANTS
Jerome L. Schneider, Boulder, CO;

**** CONTINUING DATA ***************************
This application is a CON of 11/462,827 08/07/2006 PAT 8190868 *
(*)Data provided by applicant is not consistent with PTO records.

**** FOREIGN APPLICATIONS ***************************

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ****
05/16/2012

Foreign Priority claimed ☐ Yes ☑ No 35 USC 119(a-d) conditions met ☐ Yes ☑ No Verified and Acknowledged /GREGORY A LANE/ Examiner's Signature	☐ Met after Allowance Initials	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
		CO	6	18	3

ADDRESS
Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202
UNITED STATES

TITLE
Malware Management Through Kernel Detection During a Boot Sequence

FILING FEE RECEIVED		
1250	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

Trend Micro, Inc.
EX1002-292

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13460655	SCHNEIDER, JEROME L.
	Examiner	Art Unit
	GREGORY LANE	2438

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	10/07/2013								
	1	✓								
	2	✓								
	3	✓								
	4	✓								
	5	✓								
	6	✓								
	7	✓								
	8	✓								
	9	✓								
	10	✓								
	11	✓								
	12	✓								
	13	✓								
	14	✓								
	15	✓								
	16	✓								
	17	✓								
	18	✓								

Trend Micro, Inc.
EX1002-293

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13460655	SCHNEIDER, JEROME L.
	Examiner	Art Unit
	GREGORY LANE	2438

CPC- SEARCHED		
Symbol	Date	Examiner

CPC COMBINATION SETS - SEARCHED		
Symbol	Date	Examiner

US CLASSIFICATION SEARCHED			
Class	Subclass	Date	Examiner
713	2,193	10/7/2013	GL

SEARCH NOTES		
Search Notes	Date	Examiner
searched for the inventive concept of kernel level malware	10/7/2013	GL
searched the inventors' names to check for double patenting violation	10/7/2013	GL
considered 35 USC 101 and 112 violations	10/7/2013	GL
Performed EAST search	10/7/2013	GL

INTERFERENCE SEARCH			
US Class/ CPC Symbol	US Subclass / CPC Group	Date	Examiner

/GREGORY LANE/ Examiner.Art Unit 2438	

Trend Micro, Inc.
EX1002-294

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

Sheridan Ross, PC
1560 Broadway
Suite 1200
Denver, CO 80202

MAILED

NOV 26 2012

OFFICE OF PETITIONS

In re Application of :
Jerome L. Schneider :
Application No. 13/460,655 :
Filed: April 30, 2012 :
Attorney Docket No. 6890-45-CON :

DECISION ON PETITION
TO WITHDRAW FROM
RECORD

This is a decision on the Request to Withdraw as attorney or agent of record under 37 CFR §§ 1.36(b) or 10.40, filed September 20, 2012.

The request is **MOOT**.

A review of the file record indicates that any previous power of attorney was revoked by the assignee of the above application on October 5, 2012. Accordingly, the request to withdraw under 37 CFR §§ 1.36(b) or 10.40 is unnecessary.

All future communications from the Office will be directed to the new address of record unless otherwise instructed.

Telephone inquiries concerning this decision should be directed to the undersigned at (571) 272-3226.

/Andrea Smith/
Andrea Smith
Petitions Examiner
Office of Petitions

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
13/460,655	04/30/2012	Jerome L. Schneider	WEBR-023/01US 303666-2124

CONFIRMATION NO. 9502

22442
Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

POA ACCEPTANCE LETTER

OC000000057062268

Date Mailed: 10/15/2012

NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 10/05/2012.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

/vvan/

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

Trend Micro, Inc.
EX1002-296

PTO/AIA/80 (07-12)
Approved for use through 11/30/2014. OMB 0651-0035
U.S. Patent and Trademark Office; U.S DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

POWER OF ATTORNEY TO PROSECUTE APPLICATIONS BEFORE THE USPTO

I hereby revoke all previous powers of attorney given in the application identified in the attached statement under 37 CFR 3.73(c).

I hereby appoint:

■ Practitioners associated with Customer Number: 22442

OR

☐ Practitioner(s) named below (if more than ten patent practitioners are to be named, then a customer number must be used):

Name	Registration Number

Name	Registration Number

As attorney(s) or agent(s) to represent the undersigned before the United States Patent and Trademark Office (USPTO) in connection with any and all patent applications assigned only to the undersigned according to the USPTO assignment records or assignments documents attached to this form in accordance with 37 CFR 3.73(c).

Please change the correspondence address for the application identified in the attached statement under 37 CFR 3.73(c) to:

■ The address associated with Customer Number: 22442

OR

☐	Firm or Individual Name			
	Address			
	City		State	Zip
	Country			
	Telephone		Email	

Assignee Name and Address: WEBROOT INC.
385 Interlocken Crescent, Ste. 800
Broomfield, Colorado 80021

A copy of this form, together with a statement under 37 CFR 3.73(c) (Form PTO/AIA/96 or equivalent) is required to be Filed in each application in which this form is used. The statement under 37 CFR 3.73(c) may be completed by one of The practitioners appointed in this form, and must identify the application in which this Power of Attorney is to be filed.

SIGNATURE of Assignee of Record

The individual whose signature and title is supplied below is authorized to act on behalf of the assignee

Signature	David Huberman	Date	1 OCT 2012
Name	David Huberman, Esq.	Telephone	720-842-3627
Title	General Counsel		

This collection of information is required by 37 CFR 1.31, 1.32 and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Electronic Acknowledgement Receipt

EFS ID:	13917558
Application Number:	13460655
International Application Number:	
Confirmation Number:	9502
Title of Invention:	Malware Management Through Kernel Detection During a Boot Sequence
First Named Inventor/Applicant Name:	Jerome L. Schneider
Customer Number:	58249
Filer:	Bradley M. Knepper/Erica Picard
Filer Authorized By:	Bradley M. Knepper
Attorney Docket Number:	WEBR-023/01US 303666-2124
Receipt Date:	05-OCT-2012
Filing Date:	30-APR-2012
Time Stamp:	12:03:11
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Assignee showing of ownership per 37 CFR 3.73.	STATEMENT.pdf	119874 c46ed391fc06768dc13ff08b2980cd6700892274	no	3

Warnings:	
Information:	

2	Power of Attorney	POA.pdf	59485 68e3501600681cf461a6a1ad6257bc47dc71e26d	no	1

Warnings:	
Information:	
Total Files Size (in bytes):	179359

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/AIA/96 (08-12)
Approved for use through 01/31/2013. OMB 0651-0031
U.S. Patent and Trademark Office;U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of1995, no persons arerequired to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(c)

Applicant/Patent Owner: Webroot Inc.

Application No./Patent No.: 13/460,655 Filed/Issue Date: 2012-04-30

Titled: MALWARE MANAGEMENT THROUGH KERNEL DETECTION DURING A BOOT SEQUENCE

Webroot Inc., a Corporation

(Name of Assignee) (Type of Assignee, e.g., corporation, partnership, university, government agency, etc.)

states that, for the patent application/patent identified above, it is (choose **one** of options 1, 2, 3 or 4 below):

1. ☑ The assignee of the entire right, title, and interest.

2. ☐ An assignee of less than the entire right, title, and interest (check applicable box):

   ☐ The extent (by percentage) of its ownership interest is _____%. Additional Statement(s) by the owners holding the balance of the interest must be submitted to account for 100% of the ownership interest.

   ☐ There are unspecified percentages of ownership. The other parties, including inventors, who together own the entire right, title and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

3. ☐ The assignee of an undivided interest in the entirety (a complete assignment from one of the joint inventors was made). The other parties, including inventors, who together own the entire right, title, and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

4. ☐ The recipient, via a court proceeding or the like (*e.g.*, bankruptcy, probate), of an undivided interest in the entirety (a complete transfer of ownership interest was made). The certified document(s) showing the transfer is attached.

The interest identified in option 1, 2 or 3 above (not option 4) is evidenced by either (choose **one** of options A or B below):

A. ☐ An assignment from the inventor(s) of the patent application/patent identified above. The assignment was recorded in the United States Patent and Trademark Office at Reel _____, Frame _____, or for which a copy thereof is attached.

B. ☑ A chain of title from the inventor(s), of the patent application/patent identified above, to the current assignee as follows:

1. From: Jerome L. Schneider To: Webroot Software, Inc.

   The document was recorded in the United States Patent and Trademark Office at
   Reel 028201, Frame 0093, or for which a copy thereof is attached.

2. From: Webroot Software, Inc. To: Webroot Inc.

   The document was recorded in the United States Patent and Trademark Office at
   Reel 029066, Frame 0457, or for which a copy thereof is attached.

This collection of information is required by37 CFR3.73(b). The information is required toobtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentialityis governed by35 U.S.C. 122and 37 CFR1.11 and1.14. Thiscollection is estimated to take 12 minutes to complete, including gathering, preparing, and submittingthe completed application form to the USPTO.Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent tothe Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS.**SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/AIA/96 (08-12)
Approved for use through 01/31/2013. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(c)

3. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

4. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

5. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

6. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

☐ Additional documents in the chain of title are listed on a supplemental sheet(s).

☑ As required by 37 CFR 3.73(c)(1)(i), the documentary evidence of the chain of title from the original owner to the assignee was, or concurrently is being, submitted for recordation pursuant to 37 CFR 3.11.

[NOTE: A separate copy (i.e., a true copy of the original assignment document(s)) must be submitted to Assignment Division in accordance with 37 CFR Part 3, to record the assignment in the records of the USPTO. See MPEP 302.08]

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignee.

/Bradley M. Knepper/	2012-10-05
Signature	Date
Bradley M. Knepper	44,189
Printed or Typed Name	Title or Registration Number

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that yoube given certain informationin connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, pleasebe advised that: (1) the general authority forthe collection of thisinformation is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and(3) the principal purpose forwhich the information isused by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent applicationor patent. If you do not furnish the requested information,the U.S. Patent and Trademark Office may not be able to process and/or examineyour submission,which may result in termination of proceedings or abandonment of the applicationor expiration of the patent.

The informationprovided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the informationin order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an InternationalApplication filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, arecord may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from thissystem of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

PTO/SB/83 (11-08)
Approved for use through 11/30/2011. OMB 0651-0035
U.S. Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

REQUEST FOR WITHDRAWAL AS ATTORNEY OR AGENT AND CHANGE OF CORRESPONDENCE ADDRESS

Application Number	SEE SCHEDULE A
Filing Date	SEE SCHEDULE A
First Named Inventor	SEE SCHEDULE A
Art Unit	SEE SCHEDULE A
Examiner Name	SEE SCHEDULE A
Attorney Docket Number	SEE SCHEDULE A

To: Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Please withdraw me as attorney or agent for the above identified patent application, and

☒ all the practitioners of record;

☐ the practitioners (with registration numbers) of record listed on the attached paper(s); or

☐ the practitioners of record associated with Customer Numbers: 58249 and 22903

NOTE: The immediately preceding box should only be marked when the practitioners were appointed using the listed Customer Number.

The reason(s) for this request are those described in 37 CFR :

☐ 10.40(b)(1) ☐ 10.40(b)(2) ☐ 10.40(b)(3) ☒ 10.40(b)(4)

☐ 10.40(c)(1)(i) ☐ 10.40(c)(1)(ii) ☐ 10.40(c)(1)(iii) ☐ 10.40(c)(1)(iv)

☐ 10.40(c)(1)(v) ☐ 10.40(c)(1)(vi) ☐ 10.40(c)(2) ☐ 10.40(c)(3)

☐ 10.40(c)(4) ☐ 10.40(c)(5) ☐ 10.40(c)(6). Please explain below:

Certifications

Check each box below that is factually correct. WARNING: If a box is left unchecked, the request will likely not be approved.

1. ☒ I/We have given reasonable notice to the client, prior to the expiration of the response period, that the practitioner(s) intend to withdraw from employment.

2. ☒ I/We have delivered to the client or a duly authorized representative of the client all papers and property (including funds) to which the client is entitled.

3. ☒ I/We have notified the client of any responses that may be due and the time frame within which the client must respond.

Please provide an explanation, if necessary:
SEE ATTACHED LETTER FROM WEBROOT, INC. REQUESTING TRANSFER OF FILES.

This collection of information is required by 37 CFR 1.36. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

American LegalNet, Inc.
www.FormsWorkflow.com
Trend Micro, Inc.
EX1002-303

PTO/SB/83 (11-08)
Approved for use through 11/30/2011. OMB 0651-0035
U.S. Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

REQUEST FOR WITHDRAWAL AS ATTORNEY OR AGENT AND CHANGE OF CORRESPONDENCE ADDRESS

Complete the following section only when the correspondence address will change. *Changes of address will only be accepted to an inventor or an assignee that has properly made itself of record pursuant to 37 CFR 3.71.*

Change the correspondence address and direct all future correspondence to:

A. ☐ The address of the inventor or assignee associated with Customer Number:

OR

B. ☒ Inventor or Assignee name	Webroot, Inc. c/o Sheridan Ross
Address	1560 Broadway, Suite 1200

City Denver	State CO	Zip 80202	Country US
Telephone 303-863-9700		Email rbrunelli@sheridanross.com	

I am authorized to sign on behalf of myself and all withdrawing practitioners.

Signature		
Name	Mark R. Schafer	Registration No. 65,336
Address	777 6th Street NW, Ste. 1100	

City Washington	State DC	Zip 20001	Country UA
Date 2012-09-20		Telephone No. 720 566-4044	

NOTE: ***Withdrawal is effective when approved rather than when received.***

This collection of information is required by 37 CFR 1.36. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

American LegalNet, Inc.
www.FormsWorkflow.com

Trend Micro, Inc.
EX1002-304

SCHEDULE A

Application No.	Application Date	Inventor	Group Art Unit	Examiner	Cooley Docket No.	Conf. No.
11/031,615	Jan-07-2005	THOMAS	2163	LE, Uyen	WEBR-001/01US 303666-2010	3544
10/956,573	Oct-01-2004	THOMAS	2162	Alam, Shahid	WEBR-003/00US 303666-2004	7086
10/956,274	Oct-01-2004	BERTMAN	2135	Truong, Thanhnga B.	WEBR-004/00US 303666-2008	6861
10/956,574	Oct-01-2004	THOMAS	2162	BULLOCK, Joshua	WEBR-005/00US 303666-2007	7085
11/104,201	Apr-12-2005	BURTSCHER	2167	KIM, Chong	WEBR-010/00US 303666-2014	1307
11/104,202	Apr-12-2005	BURTSCHER	2436	CERVETTI, David G.	WEBR-011/00US 303666-2011	1284
11/462,827	Aug-07-2006	SCHNEIDER	2438	LANE, Gregory A.	WEBR-023/00US 303666-2024	5954
13/460,655	Apr-30-2012	SCHNEIDER	2431		WEBR-023/01US 303666-2124	9502
11/145,592	Jun-06-2005	NICHOLS	2167	UDDIN, Mohammed	WEBR-024/00US 303666-2027	4495
11/258,536	Oct-25-2005	MOOD	2432	PERUNGAVOOR, Venkatanaray	WEBR-026/00US 303666-2028	2648
11/334,318	Jan-18-2006	HORNE	4144	WATSON, Joshua C.	WEBR-033/00US 303666-2038	6611
11/334,307	Jan-18-2006	HORNE	2431	ZIA, Syed	WEBR-034/00US 303666-2039	6605
11/334,317	Jan-18-2006	HORNE	2431	Avery, Jeremiah L.	WEBR-035/00US 303666-2040	6608
11/460,032	Jul-26-2006	BURTSCHER	2439	OLION, Brian L.	WEBR-037/00US 303666-2042	1341
11/386,595	Mar-22-2006	NICHOLS	2439	LE, Canh	WEBR-038/00US 303666-2046	4954
11/386,590	Mar-22-2006	NICHOLS	2195	WAI, Eric Charles	WEBR-039/00US 303666-2045	4944
11/386,594	Mar-22-2006	NICHOLS	2189	Peikari, Behzad	WEBR-040/00US 303666-2044	4951
11/408,146	Apr-20-2006	BONEY	2438	RAHMAN, Mahfuzur	WEBR-044/00US 303666-2049	8009
13/490,294	Jun-06-2012	BONEY	2431		WEBR-044/01US 303666-2126	1064
11/408,145	Apr-20-2006	BONEY	2438	RAHMAN, Mahfuzur	WEBR-045/00US 303666-2050	8007
13/460,648	Apr-30-2012	BONEY	2431		WEBR-045/01US 303666-2125	8780
11/482,903	Jul-07-2006	SPROWLS	2438	Truong, Thanhnga B.	WEBR-057/00US 303666-2069	6773
13/184,925	Jul-18-2011	SPROWLS	2438	Truong, Thanhnga B.	WEBR-057/01US 303666-2121	1361
13/184,931	Jul-18-2011	SPROWLS	2438	Truong, Thanhnga B.	WEBR-057/02US 303666-2122	1371
11/465,680	Aug-18-2006	WANG	2115	Connolly, Mark A.	WEBR-059/00US 303666-2065	1654
12/829,749	Jul-02-2010	WANG	2115	Connolly, Mark A.	WEBR-059/01US 303666-2119	9140
12/830,021	Jul-02-2010	WANG	2115	Connolly, Mark A.	WEBR-059/02US 303666-2120	9668
13/413,391	Mar-06-2012	WANG	2115		WEBR-059/03US 303666-2123	7167
11/462,943	Aug-07-2006	BURTSCHER	2438	LANE, Gregory A.	WEBR-060/00US 303666-2067	6147
11/462,956	Aug-07-2006	BURTSCHER	2197	DAS, Chameli	WEBR-061/00US 303666-2070	6168
11/462,781	Aug-07-2006	MCCLOY	2442		WEBR-063/00US 303666-2066	5887
12/236,419	Sep-23-2008	ADAMS	2491	SHAW, Brian F	WEBR-072/00US 303666-2108	7849

Trend Micro, Inc.
EX1002-305

Webroot
385 Interlocken Crescent Boulevard
Suite 800
Broomfield, CO 80021
Toll-Free: 800.870.8102
www.webroot.com

July 17, 2012

BY EMAIL

Wayne O. Stacy, Esq.
Cooley LLP
380 Interlocken Crescent
Suite 900
Broomfield, CO 80021

RE: Engagement of Sheridan Ross P.C.

Dear Wayne:

As you and I discussed recently, Webroot Inc. has elected to consolidate its intellectual property prosecution work with Sheridan Ross P.C. Accordingly, I write to request that you arrange for the items listed below to be forwarded Sheridan Ross as soon as possible:

- Lists of all patent, trademark, and copyright matters your firm has handled – and is currently handling – for Webroot;
- A docket report for each patent, trademark, and copyright matter detailing upcoming deadlines through December 31, 2012;
- Physical files for each patent, trademark, and copyright matter;
- Electronic files for each patent, trademark, and copyright matter, if available; and
- A master list of all items forwarded to Sheridan Ross.

All items should be sent to the attention of Robert Brunelli, 1560 Broadway, Suite 1200, Denver, Colorado 80202; rbrunelli@sheridanross.com.

Webroot anticipates that the fees/costs associated with providing the foregoing materials to Sheridan Ross will be minimal, and in no event will exceed $500 per docket. If you expect fees/costs greater than that amount, please contact me immediately.

Please note that you need not notify foreign associates of Webroot's engagement of Sheridan Ross. Sheridan Ross will handle those notifications.

Thank you for your attention to this matter.

Sincerely,

David Huberman
General Counsel
Webroot Inc.

Electronic Acknowledgement Receipt

EFS ID:	13799589
Application Number:	13460655
International Application Number:	
Confirmation Number:	9502
Title of Invention:	Malware Management Through Kernel Detection During a Boot Sequence
First Named Inventor/Applicant Name:	Jerome L. Schneider
Customer Number:	58249
Filer:	Mark Randolph Schafer
Filer Authorized By:	
Attorney Docket Number:	WEBR-023/01US 303666-2124
Receipt Date:	20-SEP-2012
Filing Date:	30-APR-2012
Time Stamp:	17:22:30
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Change of Address	WEBR-Req2Withdraw.pdf	282148 8bb7d27bcb066b34eeeca860249160c928194588	no	4

Warnings:	
Information:	

Total Files Size (in bytes):	282148

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
13/460,655	04/30/2012	Jerome L. Schneider	WEBR-023/01US 303666-2124

CONFIRMATION NO. 9502

58249
COOLEY LLP
ATTN: Patent Group
Suite 1100
777 - 6th Street, NW
WASHINGTON, DC 20001

PUBLICATION NOTICE

OC000000056137840

Title: Malware Management Through Kernel Detection During a Boot Sequence

Publication No.US-2012-0216027-A1
Publication Date:08/23/2012

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Office of Public Records. The Office of Public Records can be reached by telephone at (703) 308-9726 or (800) 972-6382, by facsimile at (703) 305-8759, by mail addressed to the United States Patent and Trademark Office, Office of Public Records, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently http://pair.uspto.gov/. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Office of Data Managment, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

Trend Micro, Inc.
EX1002-309

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number: 13/460,655

APPLICATION AS FILED - PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE($)	SMALL ENTITY FEE($)	OR	OTHER THAN SMALL ENTITY RATE($)	OTHER THAN SMALL ENTITY FEE($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	380
SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	620
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	250
TOTAL CLAIMS (37 CFR 1.16(i))	18 minus 20 =	*			OR	x 60 =	0.00
INDEPENDENT CLAIMS (37 CFR 1.16(h))	3 minus 3 =	*				x 250 =	0.00
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $310 ($155 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						0.00
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							0.00
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	1250

APPLICATION AS AMENDED - PART II

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE($)	SMALL ENTITY ADDITIONAL FEE($)	OR	OTHER THAN SMALL ENTITY RATE($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE($)
AMENDMENT A	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE($)	ADDITIONAL FEE($)		RATE($)	ADDITIONAL FEE($)
AMENDMENT B	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest found in the appropriate box in column 1.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
13/460,655	04/30/2012	2431	1250	WEBR-023/01US 303666-2124	18	3

CONFIRMATION NO. 9502

58249
COOLEY LLP
ATTN: Patent Group
Suite 1100
777 - 6th Street, NW
WASHINGTON, DC 20001

FILING RECEIPT

OC000000054318412

Date Mailed: 05/18/2012

Receipt is acknowledged of this non-provisional patent application. The application will be taken up for examination in due course. Applicant will be notified as to the results of the examination. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please submit a written request for a Filing Receipt Correction. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections**

Applicant(s)

Jerome L. Schneider, Boulder, CO;

Assignment For Published Patent Application

WEBROOT, INC., Broomfield, CO

Power of Attorney: None

Domestic Priority data as claimed by applicant

This application is a CON of 11/462,827 08/07/2006 PAT 8190868 *
(*)Data provided by applicant is not consistent with PTO records.

Foreign Applications (You may be eligible to benefit from the **Patent Prosecution Highway** program at the USPTO. Please see http://www.uspto.gov for more information.)

If Required, Foreign Filing License Granted: 05/16/2012

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 13/460,655**

Projected Publication Date: 08/23/2012

Non-Publication Request: No

Early Publication Request: No

Trend Micro, Inc.
EX1002-311

Title

Malware Management Through Kernel Detection During a Boot Sequence

Preliminary Class

726

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4158).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as

Trend Micro, Inc.
EX1002-312

set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

SelectUSA

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation and commercialization of new technologies. The USA offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to encourage, facilitate, and accelerate business investment. To learn more about why the USA is the best country in the world to develop technology, manufacture products, and grow your business, visit SelectUSA.gov.

PTO/SB/14 (11-08)
Approved for use through 01/31/2014. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	WEBR-023/01US 303666-2124
	Application Number	
Title of Invention	Malware Management Through Kernel Detection During a Boot Sequence	

The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76.
This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.

Secrecy Order 37 CFR 5.2

☐ Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)

Applicant Information:

Applicant 1 Remove

Applicant Authority ◉ Inventor ○ Legal Representative under 35 U.S.C. 117 ○ Party of Interest under 35 U.S.C. 118

Prefix	Given Name	Middle Name	Family Name	Suffix
	Jerome	L.	Schneider	

Residence Information (Select One) ◉ US Residency ○ Non US Residency ○ Active US Military Service

City	Boulder	State/Province	CO	Country of Residence [i]	US

Citizenship under 37 CFR 1.41(b) [i] US

Mailing Address of Applicant:

Address 1	1520 Harrison Avenue		
Address 2			
City	Boulder	State/Province	CO
Postal Code	80303	Country [i]	US

All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the **Add** button. Add

Correspondence Information:

Enter either Customer Number or complete the Correspondence Information section below.
For further information see 37 CFR 1.33(a).

☐ **An Address is being provided for the correspondence Information of this application.**

Customer Number	58249		
Email Address	zpatdcdocketing@cooley.com	Add Email	Remove Email

Application Information:

Title of the Invention	Malware Management Through Kernel Detection During a Boot Sequence		
Attorney Docket Number	WEBR-023/01US 303666-2124	Small Entity Status Claimed	☐
Application Type	Nonprovisional		
Subject Matter	Utility		
Suggested Class (if any)		Sub Class (if any)	
Suggested Technology Center (if any)			
Total Number of Drawing Sheets (if any)	6	Suggested Figure for Publication (if any)	

Trend Micro, Inc.
EX1002-314

PTO/SB/14 (11-08)
Approved for use through 01/31/2014. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	WEBR-023/01US 303666-2124
	Application Number	
Title of Invention	Malware Management Through Kernel Detection During a Boot Sequence	

Publication Information:

☐	Request Early Publication (Fee required at time of Request 37 CFR 1.219)
☐	**Request Not to Publish.** I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application **has not and will not** be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32).
Enter either Customer Number or complete the Representative Name section below. If both sections are completed the Customer Number will be used for the Representative Information during processing.

Please Select One:	◉ Customer Number	○ US Patent Practitioner	○ Limited Recognition (37 CFR 11.9)
Customer Number	58249		

Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, or 365(c) or indicate National Stage entry from a PCT application. Providing this information in the application data sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78(a)(2) or CFR 1.78(a)(4), and need not otherwise be made part of the specification.

Prior Application Status	Pending		Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
	Continuation of	11462827	2006-08-07

Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the **Add** button. Add

Foreign Priority Information:

This section allows for the applicant to claim benefit of foreign priority and to identify any prior foreign application for which priority is not claimed. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55(a).

Remove

Application Number	Country [i]	Parent Filing Date (YYYY-MM-DD)	Priority Claimed
			○ Yes ◉ No

Additional Foreign Priority Data may be generated within this form by selecting the **Add** button. Add

Assignee Information:

Providing this information in the application data sheet does not substitute for compliance with any requirement of part 3 of Title 37 of the CFR to have an assignment recorded in the Office.

Assignee 1 Remove

PTO/SB/14 (11-08)
Approved for use through 01/31/2014. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	WEBR-023/01US 303666-2124
	Application Number	
Title of Invention	Malware Management Through Kernel Detection During a Boot Sequence	

If the Assignee is an Organization check here. ☒			
Organization Name	Webroot, Inc.		
Mailing Address Information:			
Address 1	385 Interlocken Crescent, Suite 800		
Address 2			
City	Broomfield	**State/Province**	CO
Country [i]	US	Postal Code	80021
Phone Number		Fax Number	
Email Address			
Additional Assignee Data may be generated within this form by selecting the **Add** button.		Add	

Signature:

A signature of the applicant or representative is required in accordance with 37 CFR 1.33 and 10.18. Please see 37 CFR 1.4(d) for the form of the signature.					
Signature	/Mark R. Schafer/			Date (YYYY-MM-DD)	2012-04-30
First Name	Mark	Last Name	Schafer	Registration Number	65336

This collection of information is required by 37 CFR 1.76. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 23 minutes to complete, including gathering, preparing, and submitting the completed application data sheet form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.

2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.

3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.

4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).

5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.

6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).

7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.

8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.

9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

TITLE: **MALWARE MANAGEMENT THROUGH KERNEL DETECTION DURING A BOOT SEQUENCE**

PRIORITY

[0001] The present application is a continuation of commonly-owned and assigned U.S. Non-Provisioanl Application No. 11/462,827, entitled "Malware Management Through Kernel Detection," filed August 7, 2006, which is incorporated herein by reference in its entirety.

RELATED APPLICATIONS

[0002] The present application is also related to commonly-owned and assigned U.S. Non-Provisional Application No. 10/956,578, entitled "System and Method for Monitoring Network Communications For Pestware," and U.S. Non-Provisional Application No. 11/237,291, entitled "Client Side Exploit Tracking," each of which is incorporated herein by reference in its entirety.

FIELD OF THE INVENTION

[0003] The present invention relates to computer system management. In particular, but not by way of limitation, the present invention relates to systems and methods for controlling pestware or malware.

Trend Micro, Inc.
EX1002-318

BACKGROUND OF THE INVENTION

[0004] Personal computers and business computers are continually attacked by trojans, spyware, and adware, collectively referred to as "malware" or "pestware." These types of programs generally act to gather information about a person or organization—often without the person or organization's knowledge. Some pestware is highly malicious. Other pestware is non-malicious but may cause issues with privacy or system performance. And yet other pestware is actual beneficial or wanted by the user. Wanted pestware is sometimes not characterized as "pestware" or "spyware." But, unless specified otherwise, "pestware" as used herein refers to any program that collects and/or reports information about a person or an organization and any "watcher processes" related to the pestware.

[0005] The design and implementation of current and future pestware incorporates techniques that make the pestware difficult to identify, remove, or even to detect. These techniques, and likely future improvements to them, rely on patches, hooks and yet-to-be-discovered methods for modifying the behavior of a computer operating system itself. Such techniques render current detection tools ineffective by intercepting and altering the results of operating system queries from the tools that must rely on the dependability of operating system calls to return lists of running programs, file system and registry contents, for example.

[0006] Detection of pestware that uses these cloaking techniques is often ineffective by the real-time shields that existing anti-pestware applications utilize because these real-

time shields begin execution after pestware has been able to execute and modify the operating system. FIGURE 3, for example, illustrates the boot and operating system sequences that demonstrate the inability of user-mode services or applications to detect and prevent pestware (e.g., a Rootkit) that loads before them in a boot sequence. Periodic scanning for pestware is particularly ineffective because it leaves a window of time between scans in which pestware can execute and cloak itself. Accordingly, current software is not always able to identify and remove pestware in a convenient manner and will most certainly not be satisfactory in the future.

SUMMARY OF THE INVENTION

[0007] Exemplary embodiments of the present invention that are shown in the drawings are summarized below. These and other embodiments are more fully described in the Detailed Description section. It is to be understood, however, that there is no intention to limit the invention to the forms described in this Summary of the Invention or in the Detailed Description. One skilled in the art can recognize that there are numerous modifications, equivalents and alternative constructions that fall within the spirit and scope of the invention as expressed in the claims.

[0008] In one embodiment, the invention may be characterized as a method for managing pestware on a computer. The method in this embodiment includes monitoring events during a boot sequence of the computer; managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occuring

before the computer becomes configured to run native applications, before a subsystem of an operating system is loaded, and after a kernel is loaded; managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occuring when the computer is configured to run native applications; and scanning a registry of the computer for pestware during the second period in the boot sequence.

[0009] As previously stated, the above-described embodiments and implementations are for illustration purposes only. Numerous other embodiments, implementations, and details of the invention are easily recognized by those of skill in the art from the following descriptions and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Various objects and advantages and a more complete understanding of the present invention are apparent and more readily appreciated by reference to the following Detailed Description and to the appended claims when taken in conjunction with the accompanying Drawings wherein:

FIGURE 1 is a block diagram depicting a pestware management system in accordance with an exemplary embodiment of the present invention;

FIGURE 2 is a flowchart depicting steps carried out in accordance with an exemplary embodiment of the present invention;

FIGURE 3 is a timing diagram depicting a boot sequence in accordance with prior art pestware management techniques;

FIGURE 4 is a timing diagram depicting a boot sequence in accordance with pestware management techniques of several embodiments of the present invention;

FIGURE 5 is a block diagram of a pestware management system in accordance with another embodiment of the present invention; and

FIGURE 6 is a timing diagram depicting another boot sequence in accordance with another embodiment of the present invention.

DETAILED DESCRIPTION

[0011] Referring now to the drawings, where like or similar elements are designated with identical reference numerals throughout the several views, and referring in particular to FIGURE 1, it illustrates a block diagram 100 of one implementation of the present invention. This implementation includes four components: a kernel-level monitor 105, a pestware management engine 110, event storage 115 and behavior rule storage 120.

[0012] The kernel-level monitor 105 and a pestware management engine 110 can be implemented in software or hardware. And if implemented in software, the modules can be designed to operate on any type of computer system including WINDOWS and Linux-based systems Additionally, the software can be configured to operate on personal computers and/or servers. For convenience, embodiments of the present invention are generally described herein with relation to WINDOWS-based systems. Those of skill in

the art can easily adapt these implementations for other types of operating systems or computer systems.

[0013] The event storage 115 and behavior rule storage 120 may be realized by a single magnetic hard drive, distributed hard drives or non-volatile memory. In some variations, the event storage 115 may be implemented with volatile memory.

[0014] Referring first to the kernel-level monitor 105, in the exemplary embodiment it is responsible for detecting pestware or pestware activity on a protected computer or system. (The term "protected computer" is used to refer to any type of computer system, including personal computers, handheld computers, servers, firewalls, etc.) In several embodiments. the kernel-level monitor 105 begins execution early in the boot and operating system loading process on the protected computer, and as discussed further herein, the kernel-level monitor 105 obtains (e.g., in a secure manner), a most recent set of behavior rules from the behavior rules storage 120 that was previously compiled by the pestware management engine 110.

[0015] The kernel-level monitor 105 additionally uses several techniques that allow it to intercept and monitor various operating system behaviors, including attempts by pestware to modify the behavior of the operating system through the use of patches and hooks or by invoking other mechanisms that could be used to alter information passed to or from the operating system.

[0016] Moreover, the kernel-level monitor 105 in the exemplary embodiment also provides mechanisms that can prevent, disable or disallow attempts by pestware to use or create patches, hooks and other methods required for intercepting or altering operating system information. In addition, the kernel-level monitor 105 identifies specific attempts by other software by locating the specific instance within the set of behavior rules 120 generated by the pestware management engine 110.

[0017] Beneficially, the kernel-level monitor 105 in the exemplary embodiment also constructs and stores an event log in the event storage 115 that contains details for each of the intercepted and monitored events detected so that after the pestware management engine 110 executes, it acquires the event log from the event storage module 115.

[0018] The pestware management engine 110 in the present embodiment identifies, by examining the event log created by the kernel-level monitor 105, instances of known pestware, suspicious events by unknown software, and other patterns of events that may be useful in identifying pestware.

[0019] In addition, the pestware management engine 110 can add to or modify the common set of behavior rules that are utilized by kernel-level monitor 105 the next time the kernel-level monitor 105 loads and executes. In this way, the pestware management engine 110 is able to provide updated information about pestware-related events that should be prevented by the kernel-level monitor 105 during future boot operations on the protected computer.

[0020] Advantageously the ability to first observe, using the kernel-level monitor 105, events associated with the loading, execution and behaviors of pestware, followed by the pestware management engines's 110 incorporation of the events into the behavior rules 120 allows, in subsequent boot operations, the kernel-level monitor 105 to prevent or deny the events from happening. Moreover, the pestware management module 110 may receive updated information about events that should be prevented from a remote host (not shown). The above-identified application entitled *Client-side Exploit Tracking* includes details about developing behavior rules at a centralized host location that may be pushed out to protected computers.

[0021] Referring next to FIGURE 2, it is a flowchart of one method for managing pestware. In this method, a boot sequence of the protected computer is initiated (e.g., by a user of the protected computer)(Block 202), and in advance of a period when native applications (e.g., checkdisk) of the protected computer beginning execution, the kernel-level monitor 105 is initialized (Block 204).

[0022] In several embodiments, for example, the kernel-level monitor 105 is loaded with boot drivers and initialized before at least some of the boot drivers. In some embodiments, the kernel-level monitor 105 is initialized before any of the boot drivers so as to be capable of monitoring the initialization of the boot drivers. In one embodiment, for example, a low tag number (e.g., the lowest tag number) is associated with the kernel-level monitor 105 so that the kernel-level monitor 105 is initialized before boot drivers, with higher tag numbers, are initialized. It is contemplated that once the kernel-level

monitor 105 is provided with a low tag number, registry keys of the protected computer may be protected so as to prevent pestware from subsequently altering the tag order.

[0023] In yet another embodiment, a tag number of zero is associated with the kernel-level monitor 105 and the operating system of the protected computer is altered so as to allow a tag number of zero to be initialized first, before any boot drivers are initiated.

[0024] Referring briefly to FIGURE 4, which depicts an exemplary boot and operating system load sequence, the kernel-level monitor 105 in the exemplary embodiment is initialized before a period when native applications may be run. More specifically, as depicted in FIGURE 4, the kernel-level monitor 105 is loaded in the boot sequence when boot drivers are loaded and initialized at a very early (e.g., first) stage during the period in which the boot drivers are initialized. As depicted in FIGURE 4, in an alternative embodiment, the kernel-level monitor 105 is loaded from a BIOS of the protected computer.

[0025] In many embodiments of the invention, the kernel-level monitor 105 is realized by a Windows device driver that can operate in the constraints imposed on the boot drivers, which are needed to operate peripherals (e.g., keyboard, mouse, monitor, hard drive, etc.) during the initial boot process.

[0026] During the time when the kernel-level monitor 105 is loaded and initialized, driver hooks and monitoring mechanisms that are utilized by the kernel-level monitor 105 are placed into the operating system kernel. Referring again to FIGURE 4, this time

period includes the time period depicted as "Event Snapshot and Rule Enforcement." In the exemplary boot sequence depicted in FIGURE 4, the kernel-level monitor 105 monitors events on an ongoing basis after being initialized early during the period in which the boot drivers are loaded.

[0027] Referring again to FIGURE 2, when the kernel-level monitor 105 begins execution, it acquires the most recent set of behavior rules (e.g., in a secure manner) from behavior rule storage 120 (Block 206) and begins monitoring for the events specified in the rules (Block 208). When a monitored event is detected, details of the event (e.g., ProcessID and/or filename) are added to an event record (Block 210) that is added to event storage 115, which may be either an in-memory or a file-based log of events.

[0028] As shown in FIGURE 2, the kernel-level monitor 105 then manages the protected computer in response to the event in accordance with the behavior rules (Block 212). For example, if a detected event is marked for denial or deletion in the behavior rules, the kernel-level monitor 105 either prevents the operating system from receiving the request or it forces the operating system to reject the request.

[0029] In the exemplary embodiment, as the boot sequence continues, the kernel-level monitor 105 continues to run independently, collecting event log data (e.g., during the periods in which additional drivers are initialized and when native applications are capable of executing). As depicted in FIGURE 4, as the boot sequence progresses, it is possible that a rootkit or another variety of pestware loads as a driver or service. In the

context of a Windows operating system, for example, during the period that the native operating system starts the Win32 subsystem, it is possible that a rootkit or other pestware may load as a driver or service. Events that are generated by such a pestware driver or service are monitored by the kernel-level monitor 105, added to the event log, and if the behavior rules include specific instance data indicating the event is a pestware-related event, the kernel-level monitor 105 denies or disables the rootkit's attempts to install.

[0030] As shown in FIGURE 4, in a Windows-based system, as Win32 starts global services, the kernel-level monitor 105 again monitors and possibly prevents spyware and rootkits from installing. Also during this period, the pestware management engine 110 is started as a service (Block 214).

[0031] In several embodiments, one of the tasks of the pestware management engine 110 is to communicate securely with the kernel-level monitor 105 so as to retrieve the event logs for examination and analysis (Block 216). In several embodiments, the event logs are compared against factors known to be associated with pestware so as to identify a likelihood that the events in the event log are pestware related events. In addition, the event log may be sent to a centralized host, which collects information about activities at protected computers and generates weighted factors which are pushed out to the protected computers and utilized to help identify pestware at the protected computer. The above-identified application entitled *Client Side Exploit Tracking* includes details of techniques that may be used to identify pestware-related events from the event logs.

[0032] In the exemplary embodiment, after the pestware management engine 110 is launched, it continues to communicate with the kernel-level monitor 105 so as to receive new events that are detected by the kernel-level monitor 105 and to send new behavior rules to the kernel-level monitor 105 as they are developed. If the pestware management engine 110 should cease execution, the kernel-level monitor 105 can continue to collect event logs and protect against those events covered by the current behavior rules until the pestware management engine 110 restarts and communicates once again with the kernel-level monitor 105.

[0033] Referring next to FIGURE 5, shown a block diagram 500 of another implementation of the present invention. In this embodiment, a native scanner 502 is shown operating at a native user level of the protected computer. In several embodiments, the native scanner 502 is initialized early during the period of the boot sequence during which native applications may be run. In the embodiment depicted in FIGURE 5, for example, the native scanner 502 is the first non-kernel application to initialize so as to be capable of monitoring the protected computer before any other native applications are initialized.

[0034] In operation, the native scanner 502 is configured to examine registry, filesystem and other portions of a protected computer before the Win32 subsystem and most services are loaded and executed. In this way, additional information is generated that the pestware management engine 110 can later examine and utilize in order to generate new behavior rules for the kernel-level monitor 105.

[0035] In some embodiments, the native scanner 502 may be enabled with pestware management functions such as memory and file scanning. The above-identified application entitled: *System and Method for Monitoring Network Communications* includes details of many pestware management functions that may be incorporated into the native scanner 502.

[0036] In conclusion, the present invention provides, among other things, a system and method for managing pestware during a boot sequence of a protected computer. It should be recognized, however, that embodiments of the present invention certainly have applications that extend beyond the boot sequence period of operation when the protected computer is operating in a user mode. Moreover, those skilled in the art can readily recognize that numerous variations and substitutions may be made in the invention, its use and its configuration to achieve substantially the same results as achieved by the embodiments described herein. Accordingly, there is no intention to limit the invention to the disclosed exemplary forms. Many variations, modifications and alternative constructions fall within the scope and spirit of the disclosed invention as expressed in the claims.

Trend Micro, Inc.
EX1002-330

WHAT IS CLAIMED IS:

1. A method for managing pestware on a computer comprising:

monitoring events during a boot sequence of the computer;

managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occuring before the computer becomes configured to run native applications, before a subsystem of an operating system is loaded, and after a kernel is loaded;

managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occuring when the computer is configured to run native applications; and

scanning a registry of the computer for pestware during the second period in the boot sequence.

2. The method of claim 1, further comprising launching, after an operating system is initiated, a pestware management engine; and

wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

3. The method of claim 1, further comprising:

generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;

analyzing the record of events so as to identify the pestware-related events; and

modifying the set of behavior rules so as to prevent the pestware related events.

4. The method of claim 3, wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information.

5. The method of claim 1, further comprising:

managing pestware-related events after the computer becomes configured to run native applications.

6. The method of claim 1, wherein the monitoring includes monitoring the boot sequence while boot drivers are initiated.

7. A system for managing pestware on a computer comprising:

a processor;

a memory comprising:

a first instruction set configured to monitor events during a boot sequence of the computer;

a second instruction set configured to manage pestware-related events during first period in a boot sequence of the computer, the first period in the boot sequence occuring before the computer becomes configured to run native applications, before a Win32 subsystem is loaded, and after a kernel is loaded;

a third instruction set configured to manage pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occuring when the computer is configured to run native applications; and

a fourth instruction set configured to scan a registry of the computer for pestware during the second period in the boot sequence.

8. The system of claim 7, wherein the memory further comprises a fifth instruction set configured to launch, after an operating system is initiated, a pestware management engine; and wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

9. The system of claim 7, wherein the memory further comprises:

a sixth instruction set configured to generate, in response to the monitoring, a record of events, the record of events including the pestware-related events;

a seventh instruction set configured to analyze the record of events so as to identify the pestware-related events; and

an eighth instruction set configured to modify the set of behavior rules so as to prevent the pestware related events.

10. The system of claim 9, wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information.

11. The system of claim 7, wherein the memory further comprises:

a ninth instruction set configured to manage pestware-related events after the computer becomes configured to run native applications.

12. The system of claim 7, wherein the first instruction set is further configured to monitor the boot sequence while boot drivers are initiated.

13. A non-transitory tangible computer-readable storage medium comprising program instructions for:

monitoring events during a boot sequence of the computer;

managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occuring before the computer becomes configured to run native applications, before a Win32 subsystem is loaded, and after a kernel is loaded;

managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occuring when the computer is configured to run native applications; and

scanning a registry of the computer for pestware during the second period in the boot sequence.

14. The non-transitory tangible computer-readable storage medium of claim 13 further comprising program instructions for launching, after an operating system is initiated, a pestware management engine; and

wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

15. The non-transitory tangible computer-readable storage medium of claim 13 further comprising program instructions for:

generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;

analyzing the record of events so as to identify the pestware-related events; and

modifying the set of behavior rules so as to prevent the pestware related events.

16. The non-transitory tangible computer-readable storage medium of claim 15, wherein the record of events includes information selected from the group consisting of: process identification information, file identification information, and hook generation information.

17. The non-transitory tangible computer-readable storage medium of claim 13 further comprising program instructions for:

managing pestware-related events after the computer becomes configured to run native applications.

18. The non-transitory tangible computer-readable storage medium of claim 13, wherein the monitoring includes monitoring the boot sequence while boot drivers are initiated.

Trend Micro, Inc.
EX1002-336

ABSTRACT

A system and method for managing pestware on a protected computer is described. The method in one variation includes monitoring events during a boot sequence of the computer; managing pestware-related events before native applications can run and after a kernel is loaded; managing pestware-related events when native applications can run; and scanning a registry of the computer for pestware when native applications can run. In variations, a pestware management engine is initialized after an operating system of the protected computer is initialized and the pestware management system both receives an event log of the monitored events and compiles the set of behavior rules utilized by kernel-level monitor.

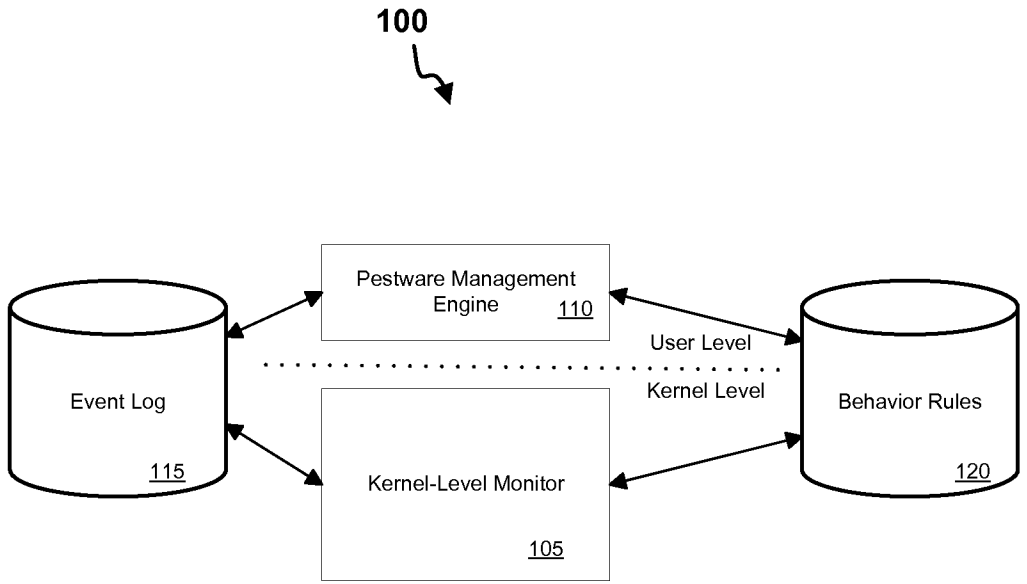


FIG. 1

ATTORNEY DOCKET NO. WEBR-023/01US

Trend Micro, Inc.
EX1002-338

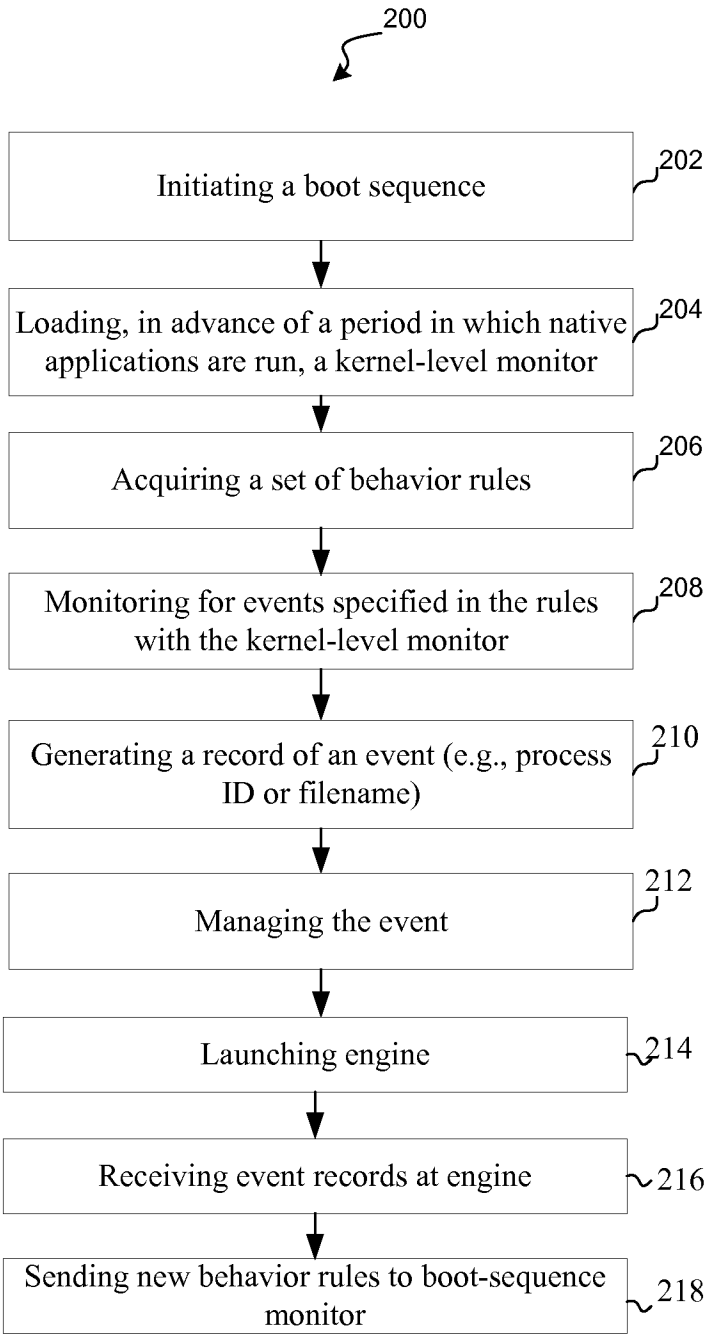
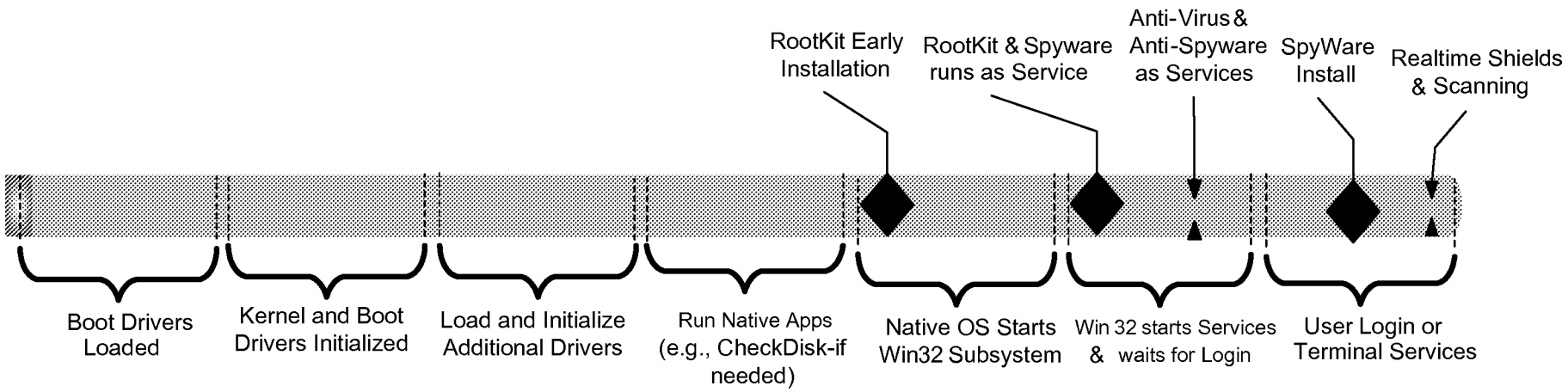


FIG. 2

ATTORNEY DOCKET NO. WEBR-023/01US

Conventional Monitoring



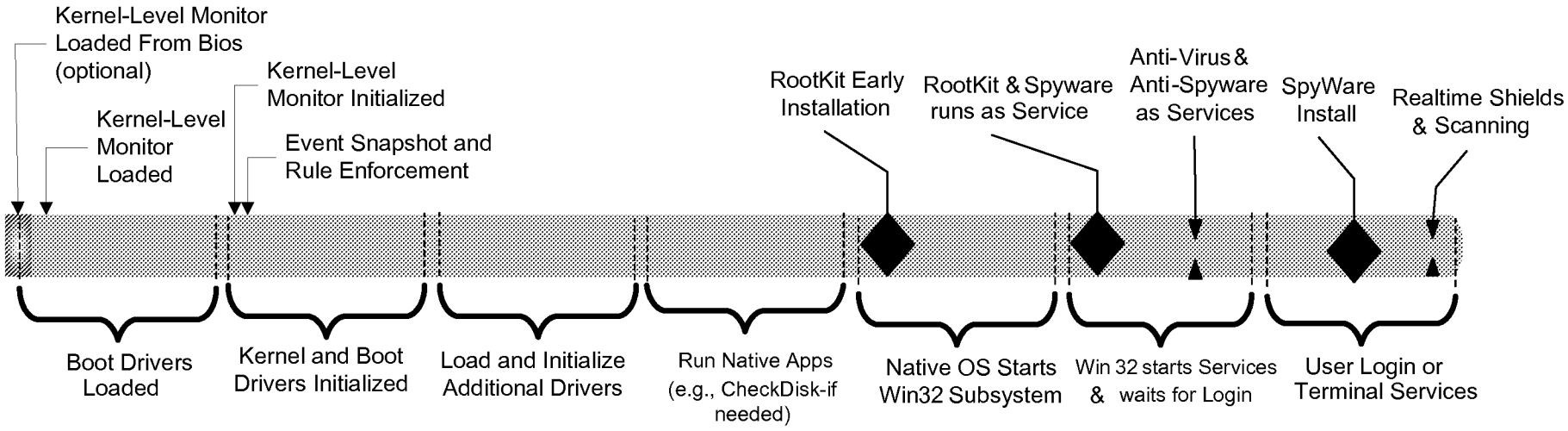
(Prior Art)

FIG. 3

ATTORNEY DOCKET NO. WEBR-023/01US

Trend Micro, Inc.
EX1002-340

Kernel-Level Monitoring



ATTORNEY DOCKET NO. WEBR-023/01US

FIG. 4

Trend Micro, Inc.
EX1002-341

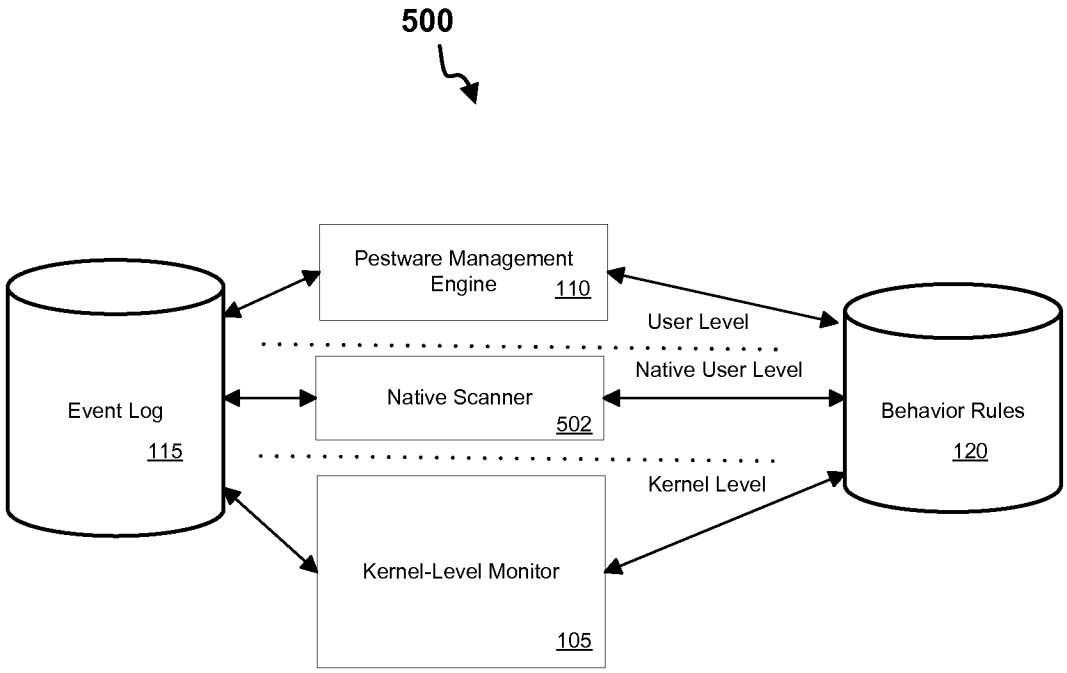
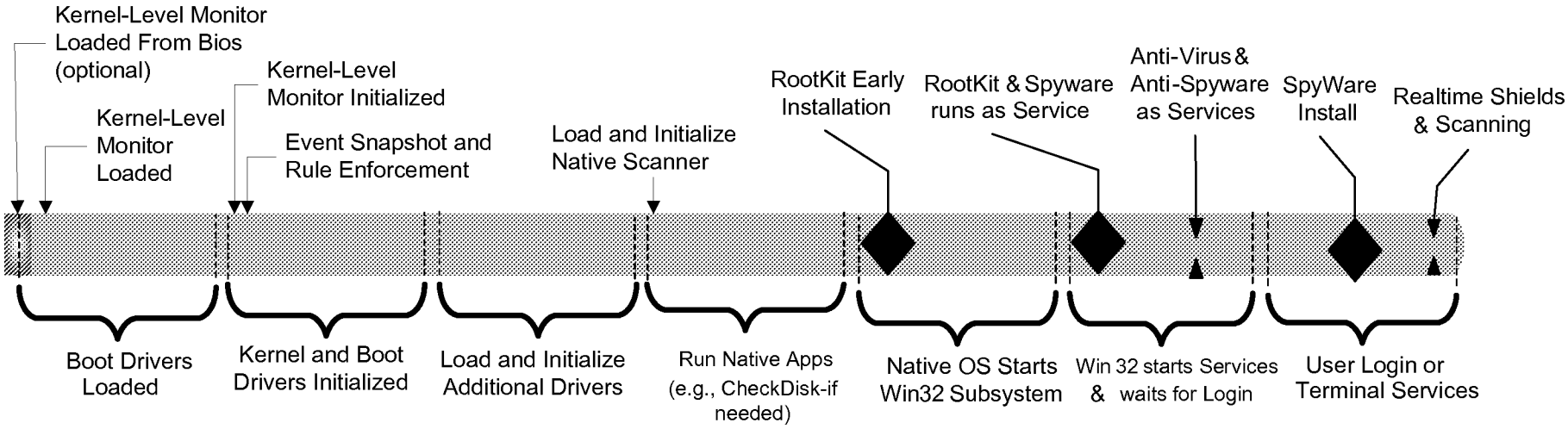


FIG. 5

ATTORNEY DOCKET NO. WEBR-023/01US

Trend Micro, Inc.
EX1002-342

Native Scanning



ATTORNEY DOCKET NO. WEBR-023/01US

FIG. 6

Trend Micro, Inc.
EX1002-343

Attorney Docket No: WEBR-023/00US **PATENT**

DECLARATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

MALWARE MANAGEMENT THROUGH KERNEL DETECTION

the specification of which:

(check one)

[] is attached hereto;

[X] was filed as United States Application Serial No. 11/462,827 on August 7, 2006.

I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above;

I acknowledge the duty to disclose to the U.S. Patent and Trademark Office all information which is known to me to be material to the patentability of said invention in accordance with 37 C.F.R. §1.56;

I hereby claim the benefit under 35 U.S.C. §119(e) of any United States provisional application(s) listed below:

N/A	
(Application Number)	(Filing Date) (day, month, year)
N/A	
(Application Number)	(Filing Date) (day, month, year)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first inventor: Jerome L. Schneider

Inventor's signature Jerome L. Schneider Date December 12, 2006

Residence: City of Boulder, State Of Colorado

Citizen of: United States of America

Post Office Address: 1520 Harrison Ave, Boulder CO 80303

Trend Micro, Inc.
EX1002-345

Electronic Patent Application Fee Transmittal

Application Number:	
Filing Date:	
Title of Invention:	MALWARE MANAGEMENT THROUGH KERNEL DETECTION DURING A BOOT SEQUENCE
First Named Inventor/Applicant Name:	Jerome L. Schneider
Filer:	Mark Randolph Schafer/Sherry Bitler
Attorney Docket Number:	WEBR-023/01US 303666-2124

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Utility application filing	1011	1	380	380
Utility Search Fee	1111	1	620	620
Utility Examination Fee	1311	1	250	250
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Miscellaneous:				
			Total in USD ($)	**1250**

Electronic Acknowledgement Receipt

EFS ID:	12666640
Application Number:	13460655
International Application Number:	
Confirmation Number:	9502
Title of Invention:	MALWARE MANAGEMENT THROUGH KERNEL DETECTION DURING A BOOT SEQUENCE
First Named Inventor/Applicant Name:	Jerome L. Schneider
Customer Number:	58249
Filer:	Mark Randolph Schafer/Sherry Bitler
Filer Authorized By:	Mark Randolph Schafer
Attorney Docket Number:	WEBR-023/01US 303666-2124
Receipt Date:	30-APR-2012
Filing Date:	
Time Stamp:	20:09:12
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$1250
RAM confirmation Number	8510
Deposit Account	501283
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Application Data Sheet	WEBR02301US-ads.pdf	1905766 8d9f6800e9ac4bebf1eb4f1d8a84576d15047c40	no	4
Warnings:					
Information:					
2		WEBR02301US_Nonprovisional_Application.pdf	140947 0613513e6a7ebd61a8cd8c0de63ac758c97fcf94	yes	20

Multipart Description/PDF files in .zip description

Document Description	Start	End
Specification	1	13
Claims	14	19
Abstract	20	20

Warnings:					
Information:					
3	Drawings-only black and white line drawings	WEBR02301US_Drawings.pdf	149106 cafabf5f582c14a9ebb691e4ba06be77d9b0bf91	no	6
Warnings:					
Information:					
4	Oath or Declaration filed	WEBR02301US-declaration.pdf	60913 d94dc541dfa01559c343c77c8142f9408a59a136	no	2
Warnings:					
Information:					
5	Fee Worksheet (SB06)	fee-info.pdf	33191 91a3317dc1f06ede08ba2054bc3cba1882272bcc	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	2289923		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

DocCode - SCORE

SCORE Placeholder Sheet for IFW Content

Application Number: 13460655 Document Date: 4/30/2012

The presence of this form in the IFW record indicates that the following document type was received in electronic format on the date identified above. This content is stored in the SCORE database.

- Drawings – Other than Black and White Line Drawings

Since this was an electronic submission, there is no physical artifact folder, no artifact folder is recorded in PALM, and no paper documents or physical media exist. The TIFF images in the IFW record were created from the original documents that are stored in SCORE.

To access the documents in the SCORE database, refer to instructions developed by SIRA.

At the time of document entry (noted above):

- Examiners may access SCORE content via the eDAN interface.
- Other USPTO employees can bookmark the current SCORE URL (http://es/ScoreAccessWeb/).
- External customers may access SCORE content via the Public and Private PAIR interfaces.

Form Revision Date: February 8, 2006