



Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss



Malware behavioural detection and vaccine development by using a support vector model classifier



Ping Wang*, Yu-Shih Wang

Department of Information Management, Kun Shan University, Tainan, Taiwan

ARTICLE INFO

Article history:

Received 16 March 2014

Received in revised form 4 August 2014

Accepted 19 August 2014

Available online 18 December 2014

Keywords:

Behavioural detection

Digital vaccine

Malware detection system

Mobile security

Support vector model (SVM)

ABSTRACT

Most existing approaches for detecting viruses involve signature-based analyses to match the precise patterns of malware threats. However, the problem of classification accuracy regarding unspecified malware detection depends on correct extraction and completeness of training signatures. In practice, malware detection system uses the generalization ability of support vector models (SVMs) to guarantee a small classification error by machine learning. This study developed an automatic malware detection system by training an SVM classifier based on behavioural signatures. A cross-validation scheme was used for solving classification accuracy problems by using SVMs associated with 60 families of real malware. The experimental results reveal that the classification error decreases as the sizing of testing data is increased. For different sizing (N) of malware samples, the prediction accuracy of malware detection goes up to 98.7% with $N = 100$. The overall detection accuracy of the SVC is more than 85% for unspecific mobile malware.

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

Effective security defence mechanisms involving threat-analysis techniques in open networks are essential for detecting intruder attacks. In implementing network applications, defence mechanisms against network threats must focus on two fundamental security concerns. First, vulnerabilities that are exploited by malware must be identified, and the exploitability must be compared with that of attack scenarios. Second, established methods for detecting malware must be used for classifying malicious executables to respond promptly to cyber attacks [1,2].

The automatic malware detection system (AMDS) [3] is typically used for detecting and evaluating potential attack profiles by incorporating cyber-threat analysis (CTA) [4,5] techniques to assist defenders in determining effective defences against network threats caused by malware infection. CTA of malware attacks typically focuses on examining threats and their exposure by accumulating information on recognised attacks to identify malware signatures associated with system vulnerabilities to estimate detection accuracy and the impact of malware threats, as described in the Common Vulnerabilities and Exposures dictionary.

Current malware detection schemes, such as signature-based and semantic analyses, provide methods for examining the precise patterns of malware threats. Signature-based detection is the most widely employed technique in antivirus software featuring precise comparison. Studies on malware detection have primarily focused on performing static analyses to inspect the code-structure signature of viruses, rather than dynamic behavioural aspects. In other words, automatic analysis of malware behaviour using machine learning (ML) techniques for determining unidentified classes of malware or variants

* Corresponding author.

E-mail address: pingwang@mail.ksu.edu.tw (P. Wang).

<http://dx.doi.org/10.1016/j.jcss.2014.12.014>

0022-0000/© 2014 Elsevier Inc. All rights reserved.

have generally been disregarded. The complete installation of precise patterns for real-time malware detection programs on mobile devices remains a challenging task, restricted by computing power, battery capacity, and limited storage space.

Support vector machines (SVMs) [6,7] are used for clustering data into two categories according to maximum boundary geometry. Solving classification problems by using an SVM classifier (SVC) guarantees few classification errors caused by maximising the generalisation ability of learning incorporating the Lagrange multiplier optimisation algorithm [8]. Generally, the results from using SVM classification algorithms are more accurate than those from using other ML approaches involving non-optimised search methods, such as artificial neural networks, least squares, k -nearest neighbour, Bayesian probability, and classification and regression trees [9], particularly when defence systems collect only limited training data.

Previous researches [10–12] have indicated that SVM analysis is useful for discriminating malicious behaviour of malware from the normal behaviour of legitimate applications by training a classifier. SVM is not only used for detecting identified malware, but also for predicting the classification of unidentified malware.

Malware variants generally exhibit similar behaviours to those of malware signatures. Thus, behavioural detection schemes can detect new malware or variants based on existing malware. Furthermore, generalisation is a key benefit of using a behavioural approach instead of payload signatures [10]. This study proposes an improved behaviour-based SVC learning algorithm for use in the AMDS to categorise mobile malware according to collected malware behaviours, enabling the defence system to respond promptly to high-risk security concerns. In particular, the learning signature synthesises both the code features of static analysis and the behavioural patterns of dynamic analysis techniques. A grid search algorithm [13] was used to facilitate defence systems to increase training accuracy by considering the selection of SVC parameters. Furthermore, a digital vaccine (DV) [14] against cyber attacks was developed as a defence solution for preventing malware infection in mobile devices. System validation involved a cross-validation scheme to classify and identify the class of malware by using SVCs associated with 60 families of real mobile malware to test their accuracy.

Three crucial steps were considered in developing the proposed model: 1) perform malware classification by using a behaviour-based SVC based on a heuristic approach incorporating both behavioural [10,15] and code analyses [16,17] of malware to accurately determine the signature of mobile devices, 2) propose a digital vaccine that effectively prevents malware infection and treats infected hosts with a backup and restore approach, and 3) comprehensively determine the classification accuracy of unspecified malware or variants.

The remainder of the paper is organised as follows: Section 2 reviews the basic principles of SVM. Section 3 presents the proposed analytical model used to evaluate the detection accuracy of mobile malware and discusses an effective defence solution for malware infection, and Section 4 describes the proposed approach by presenting two cases of mobile malware attacks on a cloud computing security application. Finally, Section 5 concludes the paper.

2. Related work

This section reviews the use of two important issues, namely classification techniques for anomaly-based detection and SVM, in establishing classification rules of malware detection for discriminating the abnormal behaviour from the normal behaviour of the AMDS to solve the threat analysis problem of cyber attacks.

2.1. Classification techniques for anomaly-based detection

The classification problem is most frequently discussed regarding data mining or machine learning (ML) techniques. Many classification approaches incorporate ML algorithms for detecting malware [13,18,19]. Machine learning techniques for classification algorithm, including SVMs, least squares (LR), k -NN (k -nearest neighbours), decision tree (DT), artificial neural networks (ANN), and Bayesian classifiers approaches have been used to facilitate the prediction performance. These classification schemes are summarised in Table 1.

Compared with those of the traditional ML methods such as LR and k -NN, SVMs have produced excellent results, and are generally considered as best classifiers by a clear margin as the feature set gets larger provided the sample size is not too small [22]. It implies that SVM can be used to discriminate the abnormal behaviour from the normal behaviour for AMDS by training a binary classifier.

2.2. Support vector machine

SVMs, developed by Vapnik in 1995 (AT&T Labs), are supervised learning models associated with learning algorithms and used to analyse data and recognise patterns. In an SVM training algorithm, new examples are assigned to one category or another as either nonlinear or linear binary classifiers obtained from a set of training examples. SVMs have been proven a useful tool for conducting clustering and classification analyses. In particular, SVM theory has been developed gradually from linear SVCs to hyperplane classifiers; that is, SVMs can efficiently perform nonlinear classification by using a kernel function, implicitly mapping their inputs into high-dimensional feature spaces by selecting an appropriate kernel function. Furthermore, a favourable classification result is achieved using a hyperplane that has the largest distance from the nearest training data point of any class [4]. Basic SVM theory is discussed as follows [3,8,23,24].

Given a training dataset D , (x_i, y_i) , where x_i denotes n observations of malware signatures, $x_i \in R^N$, $i = 1, \dots, N$; and y_i is the corresponding class label whose value is either 1 or -1 (i.e., malicious or benign), indicating the class to which the

Table 1

Machine learning approaches for classification.

Approaches	Features	Suitable for	Limitations
Naive Bayes models (Bayesian classifiers) [18]	The network structure consists of only two layers, the class variable in the root node and all the other variables in the leaf nodes.	In empirical tests, Naive Bayes classifiers often outperformed more sophisticated classifiers like decision trees or general Bayesian networks, especially with small datasets (up to 1000 rows).	It is assumed that all leaf nodes are conditionally independent which is often unrealistic, but in practice the Naive Bayes model has worked very well.
ANN [19]	ANNs are computational models inspired by an animal's central nervous systems using machine learning technique to facilitate searches for decision boundaries by minimising the error rate.	The ANN is a machine learning approach used to solve a wide variety of optimal problems that are hard to solve using ordinary rule-based programming.	The weights established for training data may not be generalisable to other data sets, even from the same populations. ANN might lead to over-fitting.
k-NN [20]	k-NN classifier is a non-parametric method used for classification and regression.	k-NN algorithm is among the simplest of all machine learning algorithms using instance-based learning technique, where the function is only approximated locally.	It is sensitive to the local structure of the data. Generally, k-NN classifier has large storage requirements.
SVM [21]	The basic idea is to find a set of support vectors which define the widest linear margin between two classes.	Generally, SVMs provide a suitable means of clustering data for small data sets especially, because the classification is based on support vectors, and data dimension–size ratio has no effect on model complexity.	Selecting appropriate kernel function and parameters is difficult, and involves new tests of different settings empirically.
LR [22]	In general, LR estimates empirical values of the parameters in a qualitative response model, i.e., LR is a form of parametric regression.	It is used when the dependent variable is a dichotomy and the independents are continuous variables, categorical variables, or both.	The search and inclusion of variable interaction terms in LR is possible, however, it has to be done manually which is time consuming and usually suboptimal.

point x_i belongs, $y_i \in \{1, -1\}$, assigned to each observation x_i . Each behavioural signature x_i is of dimension d corresponding to the number of propositional variables

$$D = \{(x_i, y_i) \mid x_i \in R^N, y_i \in \{1, -1\}\}_{i=1}^N. \quad (1)$$

A typical clustering problem is identifying the maximum margin of a hyperplane that divides the points exhibiting $y_i = 1$ from those exhibiting $y_i = -1$. Any hyperplane can be written as the set of points, x , satisfying the following formula:

$$w \cdot x_i + b = 0 \quad \forall i \quad (2)$$

where $\cdot$ denotes the dot product and W denotes the normal vector of the hyperplane. The parameter $\frac{b}{\|w\|}$ determines the offset of a hyperplane from the origin along the normal vector W . Generally, a decision function $D(x)$ is defined for clustering as $D(x) = w \cdot x + b$.

As shown in Fig. 1, the Lagrange multiplier for dual optimisation theory was used to determine the maximal and minimal optimisation functions, which provided a viable solution. To solve the problem of identifying the maximum margin of a hyperplane, the Lagrange function is expressed as follows:

$$L_P = L(w, l) = \frac{\|w\|^2}{2} - \sum_{i=1}^N l_i [y_i (w \cdot x_i) + b] - \sum_{i=1}^N l_i, \quad (3)$$

where l_i represents a Lagrange parameter. Theoretically, solving the problem of maximising the geometric boundary requires seeking the minimum of a normal $|w^2|$, which can be transferred to minimise the Lagrange optimisation function L_P subject to the constraint $y_i (w \cdot x_i) + b - 1 \geq 0$, expressed as follows:

$$\begin{aligned} \text{MIN} \quad & L_P \quad \forall i \\ \text{Subject to} \quad & y_i (w \cdot x_i) + b - 1 \geq 0 \quad \forall i \\ & l_i \geq 0 \quad \forall i \end{aligned} \quad (4)$$

To solve the feasible solutions by applying Lagrange dual optimisation theory, a set of support vectors x_i satisfying two constraints according to the classification decision function, $D(x) : \text{sgn}[\sum_{i=1}^N l_i (y_i (w \cdot x_i) + b)]$, must be identified. After the training process was performed to select the hyperplane, the classification system could substitute the trained model

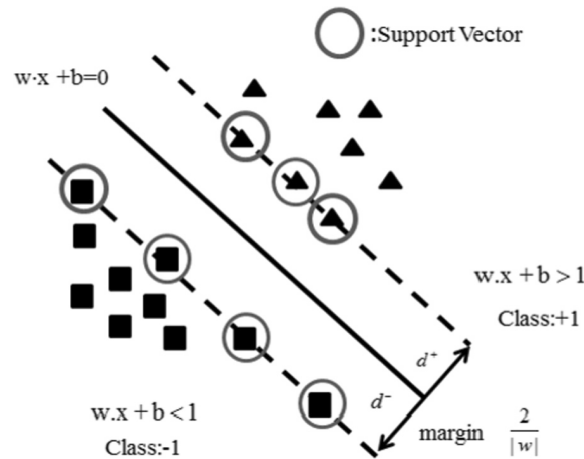


Fig. 1. Find the maximum-margin of hyperplane [7].

Table 2

Selection considerations of kernel function in LIBSVM.

Kernel	Kernel function	Features
Linear, [26]	$u'v$	A special case of RBF Simple, easy to solve
Polynomial, [21]	$(r * u'v + coef0)^{degree}$	Suitable for classify high-dimension data Might diverge in converge phase of resolution process Require more model parameters than that of RBF
Radial basis function, [27,28]	$\exp(-r * u' - v ^2)$	Suitable for classify high-dimension data Only two model parameters C and γ are needed Have been employed to a wide variety of applications
Sigmoid, [29]	$\tanh(r * u'v + coef0)$	Suitable for classify nonlinear data There is no grantee to dealing with positive semi-definite in converge phase of resolution process

parameters derived from the training data into the SVM to determine the sample class for testing the data; in other words, the positive class (+1) was predicted if $D(x) > 0$, and the negative class (−1) was predicted otherwise.

A traditional linear SVM has a key drawback, i.e., assuming that the training data are linearly separable, it cannot hold true when applied to practical real cases. Bernhard et al. [25] suggested a novel approach to generating nonlinear classifiers by applying a kernel function to maximum-margin hyperplanes. The nonlinear classification algorithm is formally similar to the linear SVM, except that each dot product (i.e., $\varphi(x_i) \bullet \varphi(x_j)$) is replaced by a kernel function. This enables the algorithm to fit the maximum margin hyperplane in a transformed feature space (i.e., $\Phi: R_d \rightarrow F$). However, rationally selecting a suitable mapping function Φ is a research issue that must be discussed.

The effectiveness of the SVM depends on the selection of a kernel and the parameters of the kernel. The selection considerations regarding kernel functions are listed in Table 2. As shown in Table 1, the radial basis function (RBF) was used as the kernel function of the SVC to establish the AMDS after the features and limitations of the four kernel functions were considered. Regarding the RBF, a combination of two model parameters (i.e., soft margin parameter C and gamma γ) was frequently selected using a grid search scheme to determine the optimised parameters of an SVC to improve classification accuracy. However, this produced exponentially growing sequences of C and γ . Each combination of parameters selected, as well as their accuracy, was assessed using a k -fold cross-validation scheme. The final model, which was used for testing and classifying new data, was then trained on the whole training set by using the selected parameters.

3. Behavioural detection model and defence mechanism for mobile malware

The proposed model was designed for identifying the classification of malware, enabling quick responses to new cyber-attack threats by collecting behaviour-based malware signatures through training an SVC by using a learning algorithm for detecting malware, appropriately detecting new malware based on existing malicious behavioural signatures, and developing a defence mechanism [30] (i.e., DV). The steps for constructing the AMDS and a DV for malware immunisation are detailed in the following subsections.

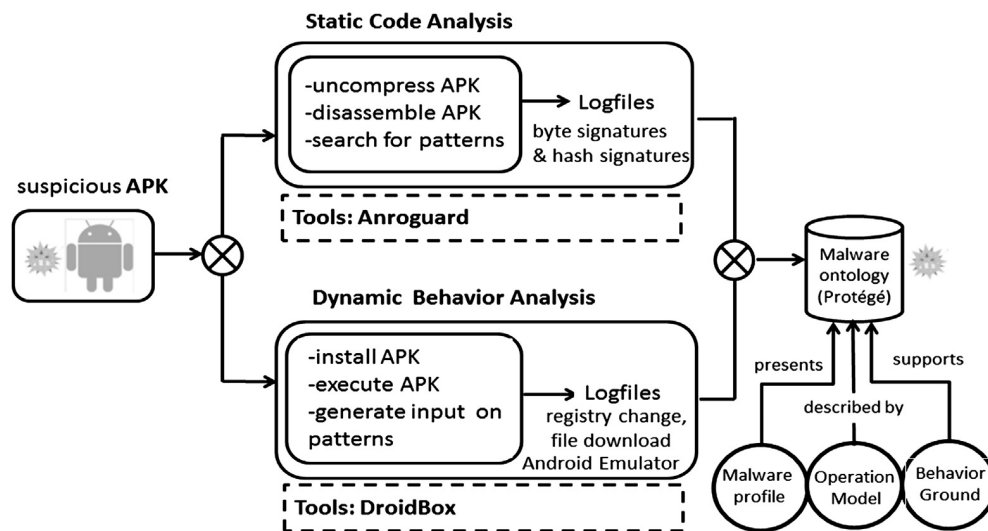


Fig. 2. Experiment environment of virus signature analyses.

3.1. Constructing an automatic malware detection system

SVMs are widely used in speech and pattern recognition and provide insight into possible network attacks with which hackers maliciously attempt to compromise network security by using malware as a tool. Thus, SVMs are effective tools used in the challenging task of automatically classifying malware [31], that is, threat trend analysis of malware infection or spread. Construction of the AMDS was divided into three steps, as described in this section.

Step 1: Malware behavioural analyses. In this study, an AMDS was incorporated to determine the attack actions and states (i.e., identified behaviours) for training an optimal SVC in the subsequent step. Training signatures were derived by aggregating both the security flaws of static code analysis and behavioural patterns of dynamic behavioural analysis with the support of a virtual machine emulator to facilitate in detecting unidentified mobile malware and variants. Dynamic analysis was conducted to obtain the major runtime behaviour regarding access to the network, file, registry, and disk of each application, which was compared with malicious and normal behavioural profiles. Static analysis was conducted to examine the specific source codes and binary strings of malware, and comparison results were recorded in a log file to discriminate the differences in the signatures of malware variants and new viruses. In the AMBAS, two free shareware products, DroidBox [32] and Androguard [17], developed by the HoneyNet Project, were used to generate malware signatures as the source dataset used to train the SVCs. An experiment was conducted using the following four steps (Fig. 2):

- 1) Download the suspected applications (.apk file format) from a mobile phone.
- 2) Perform code analysis on the DroidBox platform.
- 3) Perform behavioural analysis on the Androguard platform.
- 4) Output the synthesis report.

Step 2: Malware detection incorporating a learning algorithm. Difficulties in detecting unknown malware by using signature-based detection can generally be overcome using anomaly-based detection. In general, the classifier's fitness is often measured by a measure of prediction accuracy. Notably, a remarkable issue with the use of anomaly-based detection techniques is to validate that an alert is accurate and not an ineffective alert (i.e., false positive or false negative). Thus, a cross-validation scheme was adopted to examine the predicted accuracy of identified malware for overcoming over-training problem. In further, the present project uses a combination of both the code features of static analysis and behavioural patterns of runtime analysis to facilitate the detection accuracy of unknown mobile malware.

Two phases are typically included in anomaly-based detection: the training phase and detection phase in: 1) the training phase: performing threat analysis to determine the possible behaviour patterns of known viruses by using collected statistical data, and 2) the detection phase: classifying and identifying malware by using behavioural patterns based on malware behavioural analyses, as shown in Fig. 3.

Fig. 3 indicates that three sub-processes were included in the learning process: data analysis, model parameter determination, and system training. Fig. 4 shows that incorporating malware behavioural analysis in the AMDS is suitable for encrypted malware detection by using a virtual machine enumerator, API hooking, or sandbox analysis when the code analysis of the source structure is unavailable. The behavioural patterns were then extracted from malware actions regarding the network traffic, memory, and disk resource accesses for each malware.

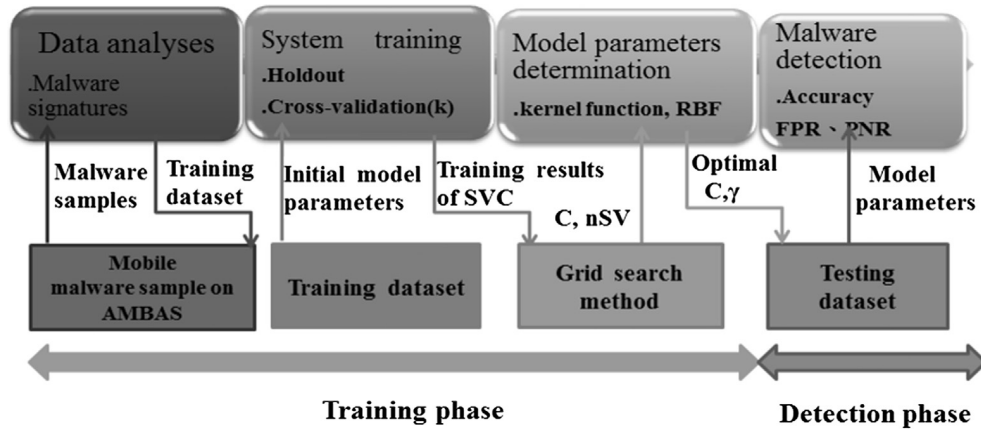


Fig. 3. Classification process of SVM.

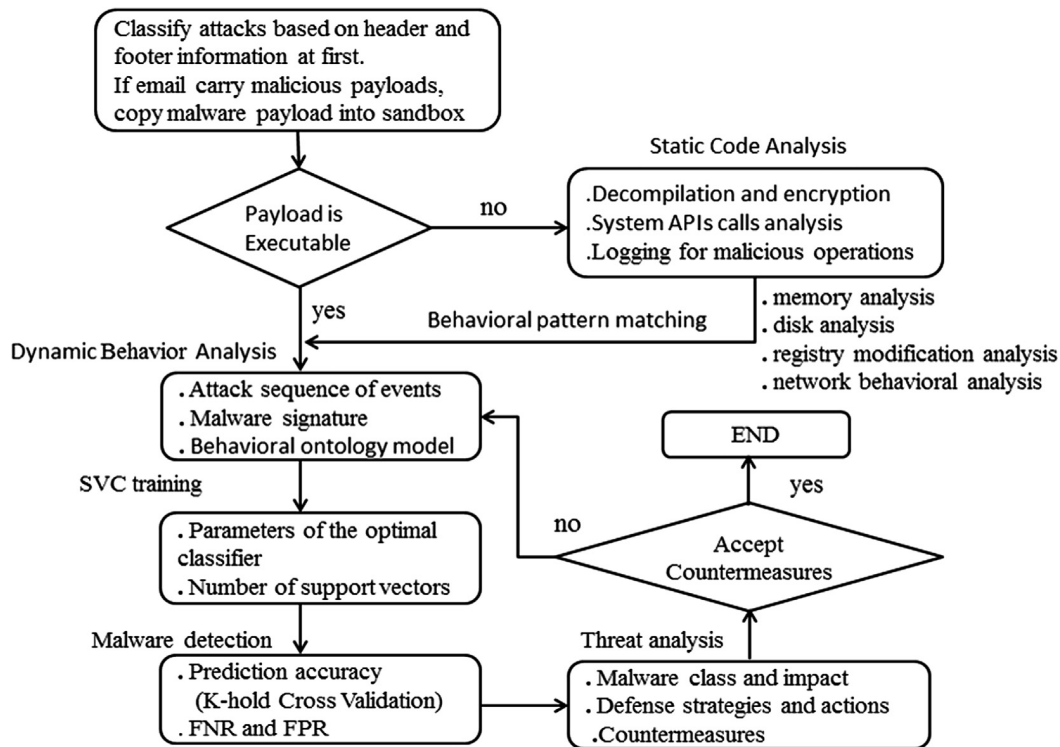


Fig. 4. The execution process of SVM training and malware detection.

A detailed algorithm for identifying the class of malware by using LIBSVM (MATLAB) is described by PDL, as shown in Fig. 5.

3.2. Intelligent defence solution of using vaccine for malware immunisation

In this section, an intelligent defence solution for malware infection that protects host system immunisation and prevents malware propagation by using the proposed DV is discussed.

1) *Fundamental concept:* To solve the problems concerning malware infection, an intelligent DV associated with a network security management platform (NSMP) was developed to provide functions for client hosts in a campus network, including malware detection, infection prevention, rapid antidote, remote control, and risk analysis, as shown in Fig. 6 [33]. The principal benefit of using the DV on an NSMP is that the platform provided an automatic backup and recovery scheme for crucial system files on client hosts and servers and prevented malware infection associated with signatures recorded in a malware ontological database by using Java Agent Development Framework (JADE) agents [34]. When the operating

Input: model parameters of SVC and malware test data.

Output: predict number of support vectors and accuracy of malware classification.

Algorithm SVCAMC: Using support vector classifier algorithm to malware classification.

```

1: Initialise the model parameters of SVC;
2: Select the kernel model as RBF and set the initial values of  $t$ :2;
3: Input the initial values of parameter  $c$ :1000 and gamma  $g$ :0.5;
4: Set the value of the parameter for stopping tolerance  $\varepsilon$  to 0.0001; (default: 0.001)
5: Assign the value of the parameter  $n$ -fold  $v$  for cross-validation to 10;
6: Training loop
7: While (cost_difference < stop condition or) do
8:   svmtrain  $-s$  0  $-t$  2  $-n$  0.1  $-g$  0.5  $-\varepsilon$  0.0001  $-v$  10 test_data
9:   return(model_file);
10: Training results of SVC, model_file include: 1) number of iteration, 2) gamma ( $\nu$ ), 3) final cost of object function ( $C$ ), 4) bias ( $\rho$ ), 5) number of
    support vectors ( $nSV$ ), 6) number of boundary support vectors ( $nBSV$ ), 7) Total  $nSV$ .
11: return svmtrain (output_file);
12: end loop
13: Predict phase
14: svmpredict Android_test_file model_file
15: return svmpredict (output_file);
16: end

```

Fig. 5. Algorithm SVCAMC.

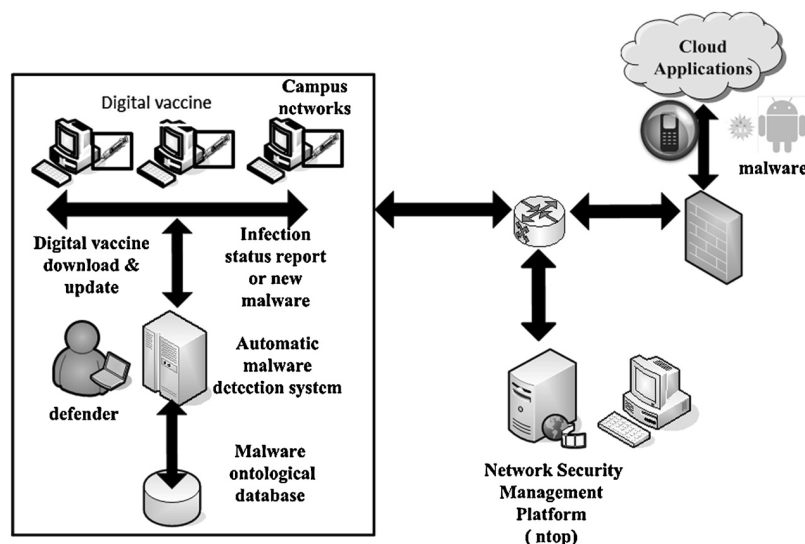


Fig. 6. Concept diagram of intelligent vaccine deployment.

system of a client was infected by identified malware, the agent triggered the DV to repair the system to the previous state, restoring the crucial backup system files. In a situation in which unidentified malware was detected, the agent generated a security log and returned trapped payloads to the NSMP to facilitate the defence system in analysing their behavioural signatures. Furthermore, the agent-based DV was suitable for the cooperative security defence of client hosts, including monitoring and handling illegal network activities, such as reporting a network connection containing numerous packets from suspicious IPs, connecting to malicious URLs, and automatically disconnecting network connections when malware attacks occur.

As shown in Fig. 7, the malware vaccine module involves three primary functions: 1) System Backup (SB), 2) System Monitoring (SM), and 3) System Recovery (SR). SB provides a backup scheme for crucial system files on client hosts and servers by implementing JADE agents for recovering the normal status of an operating system by restoring selected system files, including registries altered by malware. SM recovers operating systems and user data by using the former backup data when detecting changes in specified registries and system files infected by malware. SR, the core of DV, performs system recovery according to the malware infection process by using reverse-engineering techniques and repairs the system by referring to each rule of the established malware behavioural signatures.

2) *Producing a digital vaccine:* An exploratory project on malware vaccines implemented the NSMP by using NTOP open source code [35] and C#.Net associated with JADE software agents (refer to the behaviour and actions shown in Table 2). Assuming that the attack behaviour of malware could be analysed, the DV was constructed using the following three-phase procedure (Fig. 8): 1) produce a behaviour and actions lookup table in XML format for each malware; 2) generate antidote

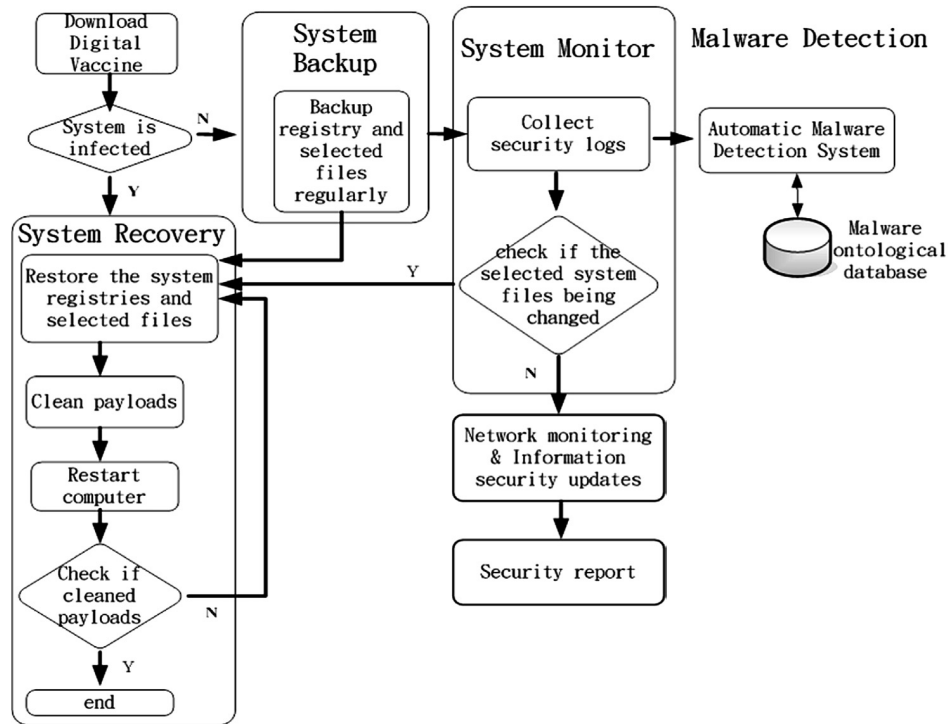


Fig. 7. Operation process of digital vaccine.

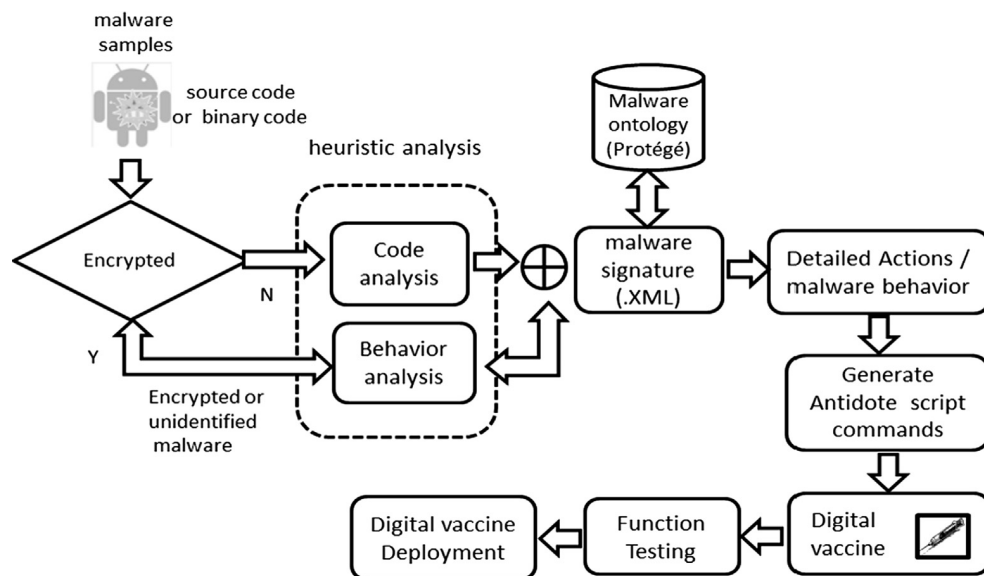


Fig. 8. Production process of digital vaccine for malware.

script commands for the DV by using reverse-engineering techniques, as listed in Table 3; and 3) recover the infected operating system by using the DV when detecting malware infection.

In 2013, two noble functions were additionally enhanced by the JADE agent in the DV: 1) real-time security status reporting, and 2) automatic signature pattern updates for client hosts. When client hosts identified unidentified malware, the agent returned the payload to the NSMP to facilitate defence systems in investigating the behavioural analysis of the malware by 1) priority ranking malware in protected networks, 2) blacklisting infected IPs, and 3) reporting malware in the relevant networks (Fig. 8). System recovery is a procedure involving the use of a DV during the antidote process, as shown in Fig. 9.

Table 3

Actions to malware behaviour for a digital vaccine.

Malicious behaviour	Reactive measures			
	Delete uploaded file	Registry recovery	Disconnect ports	Recover crucial system files
1. Network connections to <IP> <port>			V	V
2. Download payloads to <path>	V			
3. Registry modifications <object>		V		V
4. Delete files <path> <filename>				

2010/9/14 07:44:22	c:\windows\system32\dirfile.com	Set registry value	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Dir Tile	Permitted
2010/9/14 07:44:23	c:\windows\system32\dirfile.com	Access network	TCP [Local host : 1123] -> [63.173.172.98 : 6668]	Permitted
2010/9/14 07:44:35	c:\windows\explorer.exe	Create new process	c:\kda\kda.exe	Permitted
2010/9/14 07:44:42	c:\kda\kda.exe	Create new process	c:\windows\system32\netstat.exe	Permitted
2010/9/14 07:44:43	c:\kda\kda.exe	Delete registry value	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Dir Tile	Permitted
2010/9/14 07:44:44	c:\kda\kda.exe	Set registry value	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Start	Permitted
2010/9/14 07:44:44	c:\kda\kda.exe	Set registry value	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\wuauclt\Start	Permitted
2010/9/14 07:44:44	c:\kda\kda.exe	Set registry value	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\wscnt\Start	Permitted
2010/9/14 07:44:46	c:\kda\kda.exe	Terminate another pr...	c:\windows\system32\dirfile.com	Permitted
2010/9/14 07:44:46	c:\kda\kda.exe	Delete file	C:\WINDOWS\system32\dirfile.com	Permitted
2010/9/14 07:44:48	c:\kda\kda.exe	Access network	TCP [Local host : 1126] -> [72.14.213.109 : 587]	Permitted

Fig. 9. System recovered by digital vaccine.

4. Cyber security application

This section discusses the applicability of the proposed threat analysis model based on malware behavioural detection [28,36] by presenting two cloud security examples. The first example regards a case of automatic classification by using a behaviour-based SVC based on identified mobile malware exhibiting behaviour including false descriptions of applications as well as links that downloaded applications and stole information from the compromised device. The second example regards Android-based unidentified malware and variants from case I that produce false positives (i.e., any normal behaviour of applications running on the handset that may be falsely identified as malicious). In case II, assume that new malware and their variants that exhibited partial differences in behaviour compared with identified malware; in other words, Android-based new malware and variants exhibiting the behaviour are likely to be covered by existing behaviours, including 1) placing a downloaded file in a different directory, 2) using a different filename, and 3) connecting to another infected computer each time. The problem regarding accuracy in detecting malware variants was analysed.

4.1. Numerical case I

This study involved performing experiments on the Android Emulator by using the aforementioned sandbox techniques. The analysis processes involved a frequently used SVM toolbox, LIBSVM, to implement the following process. This experiment involved comparing five types of legitimate application from the Android Market including chat, travel, online game, file manager, and multimedia player applications that exhibited several partial behavioural patterns similar to those of Trojans, bots, and worms. In case I, the AMDS focused on evaluating the threat of intrusion by network malware on Android platform and involved the following three-phase procedure.

Step 1: Malware behavioural analysis. A total of 60 families of malware identified on mobile devices by using active infection manners were included from blacklists published by the Contagio Blogger Web sites [36] and Dr. Web [37] for the experiment conducted from February 2013 to February 2014. The complete execution flow of the behavioural analysis and code analysis of the mobile viruses is shown in Fig. 6. As shown in Fig. 6, a heuristic approach incorporating both behavioural analysis and code analysis of malware to determine their signatures accurately was used and assisted network defenders in rapidly detecting the mobile malware in the wild. A behavioural analysis tool (DroidBox) [32] and code analysis tool (Androguard) [17] were used to examine the behavioural patterns and structural signatures of each malware sample. Recently, mobile malware can detect the behavioural malware analysis environment based on virtual machine emulator such that behaviour analysis could not be performed wholly using the emulator environment in Fig. 2. For these cases, malware samples are to be transferred into a clean operating system (clean OS) in order to extract the behaviour patterns.

Twelve behavioural patterns (Table 4) of the virus samples were identified using behavioural analysis, and seven code structural features (Table 5) were identified using code analysis, summarised as the input of SVC training signature, and the outputs of the analyses were merged using the Concept Explorer tool from Protégé [38] to identify the relationship between malware (objects) and behaviour (attributes). Malicious characteristics are listed in Tables 4 and 5 and show the hierarchy of possible behaviours of viruses in mobile devices to assist managers in identifying the behaviour of unidentified malware.

Table 4

Signature patterns derived by behaviour analysis for malware samples.

	Sent SMS	Information leaks	Authority override	Circumvented permissions	Started service	Broadcast receiver	Operation links	Outbound traffic	Inbound traffic	Encrypted API	File read	File write
SMS Zombie	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Chuli.A	-1	-1	-1	-1	1	1	-1	1	-1	-1	-1	1
Enesoluty	-1	1	-1	-1	1	-1	1	-1	-1	-1	-1	1
System Security	-1	1	-1	-1	1	1	1	1	1	-1	-1	-1
FakeGuard	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	1
Sumzand	-1	1	-1	-1	-1	1	1	-1	-1	-1	-1	-1
FakeLookout	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	1
FinFish	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	1
Loozfon	-1	1	-1	-1	1	-1	1	-1	-1	-1	-1	1
Kranxpay	-1	-1	1	-1	1	1	1	1	1	-1	-1	1
Dropdialer	-1	-1	-1	-1	-1	-1	1	1	1	-1	-1	1
Gonfu	-1	1	1	1	1	-1	1	1	-1	-1	-1	-1
FindandCall	-1	-1	-1	-1	-1	1	1	1	-1	-1	-1	1
FakeInst.a	-1	-1	-1	1	-1	1	-1	-1	-1	-1	-1	1
Qicsomos	1	-1	1	-1	-1	-1	-1	-1	-1	-1	1	-1
Uxipp	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1
Tascudap	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1
Faketoken.b	-1	1	-1	-1	1	1	1	1	-1	-1	-1	1
Scipix	-1	-1	-1	-1	1	-1	1	-1	-1	-1	1	1
Gen.2	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	1
Fakeguard	-1	-1	-1	-1	1	1	1	-1	-1	-1	-1	1
Startapp.5.origi	-1	-1	-1	1	1	1	1	1	-1	1	-1	-1
Monad.c	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	1
Andef.b	-1	-1	-1	-1	1	1	1	1	-1	-1	1	-1
FakeMart.a	1	-1	-1	-1	-1	1	1	1	1	-1	-1	1
Fakejob	-1	1	-1	1	1	1	1	1	1	1	1	1
Opfake.bo	1	1	-1	-1	1	1	1	1	-1	1	-1	1
Stels	-1	-1	-1	1	-1	1	-1	-1	-1	-1	-1	1
SmForw.i	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1
SmsCatch.a	-1	1	-1	-1	1	1	1	1	-1	-1	1	1
Uracto	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	1	1
Pikspam	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	1	1
Claco	-1	-1	-1	-1	1	1	1	-1	-1	-1	-1	-1
Mobilespy	-1	-1	-1	-1	1	1	-1	-1	-1	-1	1	-1

Table 5

Signature patterns derived by code analysis for malware samples.

A	N	O	P	Q	R	S	T
	GPS position	Money stealing	Java refectons	SMS sent	High risk	Internet Connection	Phone call
SMS Zombie	-1	-1	-1	-1	-1	-1	-1
Chuli.A	1	-1	-1	1	1	1	-1
Enesoluty	-1	-1	-1	-1	1	1	-1
System Security	-1	1	-1	1	-1	1	-1
FakeGuard	-1	-1	-1	1	1	1	-1
Sumzand	-1	-1	-1	-1	1	1	-1
FakeLookout	1	1	1	1	1	1	-1
FinFish	1	1	1	1	1	1	1
Loozfon	-1	1	-1	1	1	1	-1
Kranxpay	1	1	1	1	1	1	-1
Dropdialer	-1	-1	-1	-1	1	1	-1
Gonfu	1	-1	1	1	1	1	-1
FindandCall	1	-1	-1	1	1	1	-1
FakeInst.a	1	1	1	1	1	1	1
Qicsomos	-1	-1	1	-1	1	1	-1
Uxipp	1	1	1	1	1	1	1
Tascudap	-1	1	-1	1	1	1	-1
Faketoken.b	-1	1	1	1	1	1	1
Scipix	-1	-1	1	-1	1	1	-1
Gen.2	1	-1	-1	-1	1	1	-1
Fakeguard	-1	-1	-1	1	1	1	-1
Startapp.5.origi	1	-1	1	1	1	1	-1
Monad.c	1	1	1	1	1	1	1
Andef.b	-1	1	1	1	1	1	1
FakeMart.a	-1	1	1	1	1	1	-1
Fakejob	1	1	1	1	1	1	-1
Opfake.bo	-1	1	1	1	1	1	-1
Stels	-1	1	1	1	-1	1	1
SmForw.i	-1	1	1	1	1	1	1
SmsCatch.a	-1	1	-1	1	1	1	-1
Uracto	-1	-1	-1	-1	1	1	-1
Pikspam	-1	1	-1	1	1	1	-1
Claco	1	1	-1	1	1	1	-1

The generalization of relationships between malware and their behaviour is formed using the concept context in Table 4 and Table 5 thru FCA (Formal Concept Analysis) which allows the merging of the same attributes of the malware as those in Protégé, the experimental results are shown in Fig. 10.

Step 2: Detecting malware by using a learning algorithm. A malware detection system must typically guarantee few classification errors caused by maximising the generalisation ability of learning in the absence of complete malware samples. First,

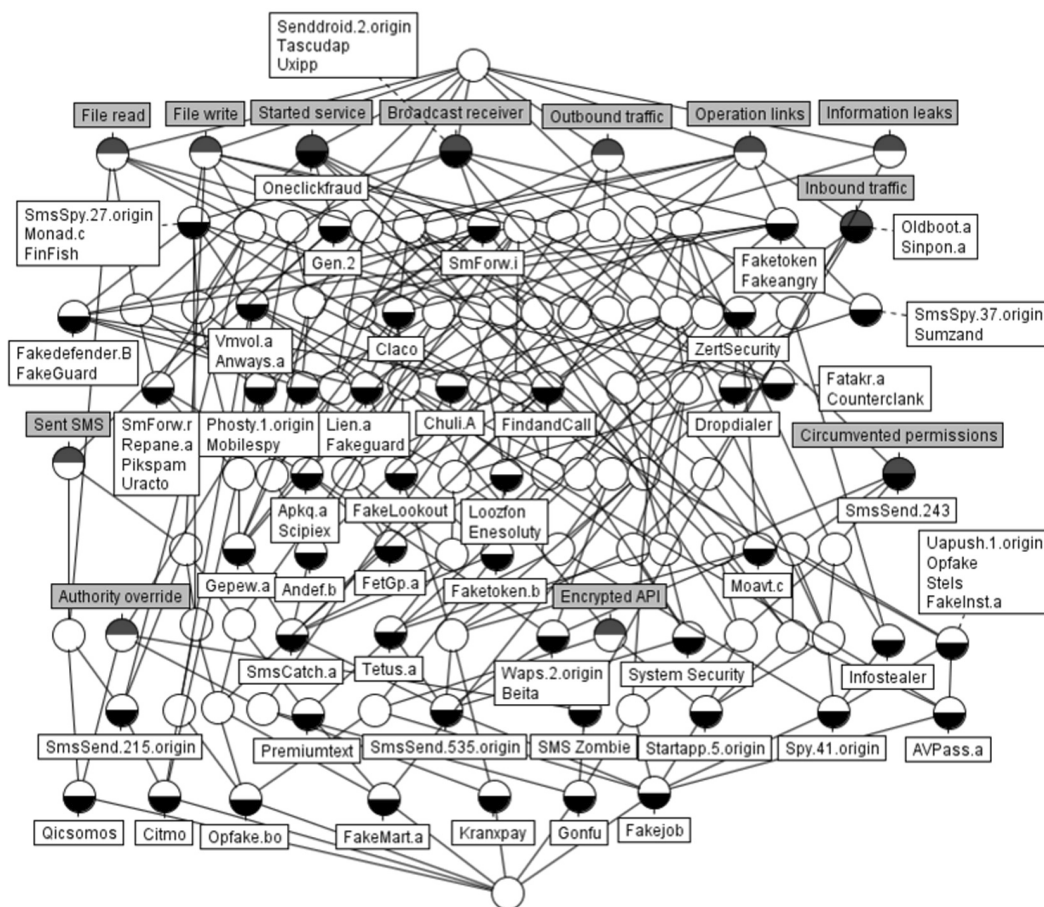


Fig. 10. An ontological model for the classification of mobile malware (STS = 70).

the SVC was trained to detect malware from the behavioural patterns of collected samples, including 20 legitimate applications from the Android Market associated with 60 families of real malicious applications. Subsequently, behavioural patterns for malware were classified accurately from those of normal applications running on the handset. Second, the accuracy of SVM in detecting existing or identified malware was evaluated. Finally, the capability of detecting unidentified malware in the wild was improved.

Step 3: Evaluating the predicted accuracy of identified malware. A cross-validation scheme was adopted to evaluate the predicted accuracy of identified malware for overcoming over-training problem by using various n -folds of the cross-validation scheme; for example, $k = 10$ means that 90% of the dataset collected were used in the training experiment, and the remaining 10% of the dataset were used for alternative testing repeated 10 times. The detailed algorithm for identifying the class of malware by using an SVM described by PDL is shown in Fig. 5.

Once the SVMs were trained, it was used to distinguish malware and benign signatures. SVM parameters were obtained through training and were required to obtain two types of training sample, namely the signature patterns of legitimate and malicious applications. In this study, the experiments were separated into two parts: 1) 12 behavioural patterns (Table 4) of the malware samples were identified using behavioural analysis to conduct the training experiment, and 2) seven code structural features (Table 5) were extracted to derive 19 features for use in the training experiment. The accuracy (%) associated with optimal parameter C - and γ -values by using the cross-validation method ($k = 2-10$) is shown in Table 6. Table 6 shows that the prediction accuracy using 19 features is higher than that of 12 behavioural patterns, i.e., the training signature derived using a heuristic approach can efficiently improve the detection precision for identified malware.

4.2. Numerical case II

Case I focused primarily on detecting identified mobile malware, which was performed to measure the correctness of training signatures, rather than the generalisation capability of detecting unidentified malware and their variants.

To evaluate the effectiveness of SVC in capturing unidentified malwares, different sizing of samples were tested including the captured malware samples from Contagio Blogger, where malware binaries could not be identified by Clam-AV tool kit.

Table 6Accuracy associated using different n -fold of cross-validation scheme.

n -fold	Behaviour analysis (12 behaviour patterns)			Behaviour analysis + code analysis (19 behaviour patterns)		
	Accuracy (%)	C	γ	Accuracy (%)	C	γ
$k = 2$	92.5	−5.47	0.2279	95.0	−6.37	0.2836
$k = 3$	95.0	−7.56	0.2211	96.2	−8.32	0.2710
$k = 4$	96.2	−8.70	0.2222	96.2	−9.60	0.2655
$k = 5$	96.2	−9.12	0.2235	96.2	−10.47	0.2748
$k = 6$	97.5	−9.12	0.2167	97.5	−10.47	0.2665
$k = 7$	96.2	−9.12	0.2104	97.5	−10.47	0.2586
$k = 8$	96.2	−9.12	0.2073	97.5	−10.48	0.2559
$k = 9$	97.5	−9.12	0.2044	97.5	−10.48	0.2522
$k = 10$	96.25	−9.74	0.2103	98.75	−11.36	0.2590

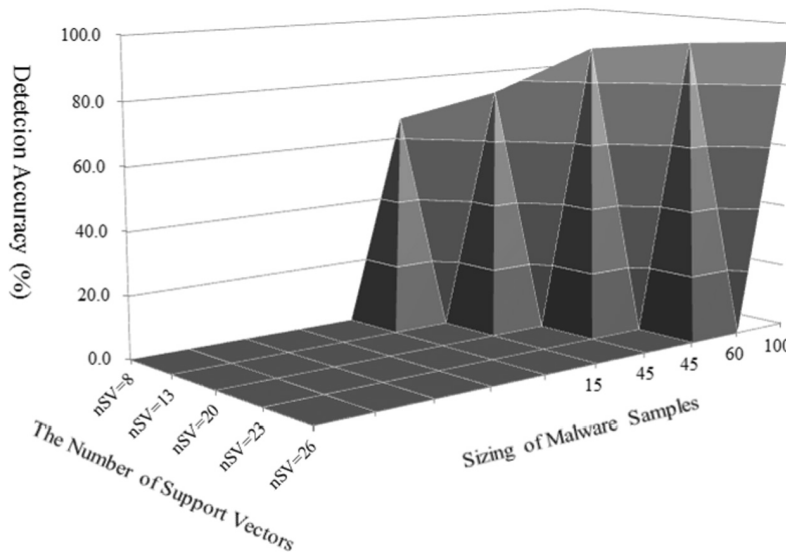
*Note: $\gamma = |2^4, 2^3, \dots, 2^{-10}|$ and $C = |2^{12}, 2^{11}, \dots, 2^{-2}|$.

Table 7

Classification accuracy of unidentified malware with different sizing of samples.

Sizing of samples	Total testing dataset (Malware samples/Legitimate apps)	Number of support vectors	TN + TP	Accuracy $\frac{TN+TP}{N}$
$N = 15$	15 (10/5)	8	10	66.6%
$N = 25$	25 (13/12)	13	19	76.0%
$N = 45$	45 (30/15)	20	41	91.1%
$N = 60$	60 (35/25)	23	56	93.3%
$N = 100$	100 (55/45)	26	94	94.0%

Where TN is the number of true negatives and TP refers to the number of true positives, $TP + TN$ represents the total number of correct classification in the testing.

**Fig. 11.** Classification accuracy of unidentified malware with samples of different size.

Notably, the experiment varies the sizing of the testing set to examine the detection accuracy on the classification error in practice; the experimental results are shown in Table 7 and Fig. 11. Table 7 lists the classification accuracy and the number of true positives and true negatives for a test set of distinct signatures with the different data sizes. Experimental results show that the classification error decreases as the sizing of testing data is increased. In other words, for different sizing of malware samples (N), the prediction accuracy of malware detection increases with an increasing value of N . The prediction accuracy of malware detection goes up to 98.7% with $N = 100$. The overall accuracy rate is 85%.

In further, to evaluate the capability of newly discovered variants by using SVCs, the collected malware were divided into two groups. The signatures of the identified malware were placed in the malicious signature database and used to train the SVM model. The samples of newly discovered variants were regarded as the test dataset and then applied in the AMDS to capture the signatures and compare them with existing signatures [39]. For example, Android.Enesoluty is a Trojan horse of Android devices that steals information and sends it to a remote host. According to the Contagio Blogger [36], Android.Enesoluty has three variants (Lime Pop, Android.Uracto, and Android.Maistealer), most of which are minor variations

Table 8

Detection accuracy of variants based on identified malware.

	Training dataset (Identified malware)	Testing variant 1A	Testing variant 1B	Testing variant 1C	Overall accuracy for variants
Training dataset	Android.Enesoluty (7)	Variant on Lime Pop game (4)	Android.Uracto (5)	Android.Maistealer (7)	
Accuracy	100%	100%	100%	93%	97.67%
Training dataset	Plankton (7)	Startapp.5 (4)	Applovin.1 (2)	Airpush.7 (10)	
Accuracy	100%	100%	100%	100%	100%
Training dataset	SMSZombie (3)	MulDrop.5 (5)	SmsSend.186 (2)		
Accuracy	100%	100%	100%		100%

NOTE (): number of malware sample.

Table 9

Detection accuracy for different mobile malware signatures.

Training dataset	Testing dataset		
	Trojan (group 1)	Spyware (group 2)	Trojan (group 3)
Trojan (group 1)	100%	X (0%)	16.0%
Spy (group 2)	X (0%)	100%	X (0%)
Trojan (group 3)	X (0%)	X (0%)	100%

of the original Trojan. For instance, Android.Uracto shares a common code with Android.Enesoluty and Android.Maistealer, malware that was designed to target Japanese users. Android.Uracto differs from Android.Enesoluty in only the infected file name. Lime Pop may become the next generation of Android.Enesoluty and is currently being developed. Therefore, only three major types of variation were implemented to classify all of the variants in the wild.

The resulting detection accuracy for varied combinations of identified and unidentified malware is summarised in Table 8. Table 8 indicates that the training signature derived using a heuristic approach could be used to detect malware variants. Thus, the SVC was observed to be robust against variants by using simple obfuscation and successfully detecting them. For example, when the training set contained three malwares, the detection achieved a high accuracy (average accuracy was 97.67%) for the remaining malware variants. Therefore, the size of the malicious signature database remained as small as possible, which was consistent with the report in Bose et al. [10].

5. Discussion

In Table 8, the similar malware signatures were used to examine the detection effect on the variations. It's an interesting question as to be discussed regarding accuracy when detecting different kinds of malware signatures for AMDS. For this experiment, three groups of malware samples were obtained from the malware ontological database in which malware were allocated a name based on VirusTotal [40]. The training experiment used one of three groups of mobile malware sample to test the detection ability on the other two groups of malware samples in the testing experiment. Notably, mobile worms and viruses in associated with their variants are not considered due to small amount of malware samples were captured. The identified mobile malware mainly focused on Trojans and spyware, in which two kinds of malicious programs have distinct signatures from Table 4 and Table 5. For example, typical behavioural patterns of spyware include 'start a service' and 'file write' for advertisement display, but viral infections for spyware do not comprise the behavioural patterns such as authority override, inbound or outbound traffic. Conversely, infection processes of Trojan generally contain authority override and network connections (inbound or outbound). Meanwhile, a large number of Trojans is captured and identified compared with spyware. Thus two kinds of malicious programs are divided into three groups of malware samples for testing and listed in Table 9.

The resulting detection accuracy for the different kinds of identified malware is summarised in Table 9, in which a list of the relevant mobile malware groups is shown in Table 10. Table 9 shows that behaviour-based approaches are not robust against different type of malware, because any two groups regularly shared dissimilar malicious signatures in the training database. In other words, a low rate of detection results was obtained even used the signatures derived from the same families of malware (e.g., group 1 and group 3). Overall, the behaviour-based approaches were effectively robust against variations of malware; only if the training dataset contained similar signatures of malware, i.e., SVC cannot guarantee a small classification error when the training process used dissimilar or invalid malicious signatures.

6. Conclusion

This paper presents a behaviour-based SVM scheme that incorporates an AMDS to state as a method for enhancing the extraction of malicious behaviours based on a heuristic approach, enabling defence systems to improve the classification accuracy of SVCs. The experimental results have confirmed that the automatic behavioural detection system detects unidentified malware or variants of existing malware, the behaviour of which was only partially matched with the signatures in the training phase. Conversely, using precise matching of signature-based detection approaches, defence systems typically

Table 10
List of mobile malware groups.

Training dataset		
Trojan (G1)	Spyware (G2)	Trojan (G3)
SMSZombie	Enesoluty	Loozfon
Chuli.A	FinFish	Kranxpay
SystemSecurity	Scipix	Dropdialer
FakeGuard	SmForw	Gonfu
FakeLookout	Uracto	FindandCall
Opfake	Tapsnake	FakeInst

require a simple mechanism and small training sample to classify additional malware with incomplete malicious behavioural signatures and are generally robust against variations of malware in practice. Overall, the proposed approach can enhance the detection of identified malware and improve the classification accuracy of new malware and variants to respond quickly to possible cyber attacks.

Although SVM techniques for malware classification have been proposed, a comprehensive and useful taxonomy used to classify novel malware is required. Behavioural analysis used for the ontological model with classification rules may be considered for determining the class of new malware in the future to produce rapid countermeasures against cyber attacks.

Acknowledgment

This work was supported by the National Science Council Taiwan under Grant No. NSC102-2218-E-168-044.

References

- [1] J.J. Julian, N. Surya, A survey of emerging threats in cybersecurity, *J. Comput. Syst. Sci.* 80 (5) (2014) 973–993.
- [2] H. Shafieia, A. Khonsaria, H. Derakhshia, P. Mousavia, Detection and mitigation of sinkhole attacks in wireless sensor networks, *J. Comput. Syst. Sci.* 80 (3) (2014) 644–653.
- [3] M. Zhao, T. Zhang, F. Ge, Z. Yuan, A lightweight malware detection using SVM, *J. Netw.* 7 (4) (2012) 715–722.
- [4] A. Gonzalez, W. Mata, A. Crespo, M. Masmano, J. Félix, A. Aburto, A hypervisor based platform to support real-time safety critical embedded java applications, *Comput. Syst. Sci. Eng.* 28 (3) (2013) 157–168.
- [5] S. Peng, G. Wang, S. Yuc, Modeling the dynamics of worm propagation using two-dimensional cellular automata in smartphones, *J. Comput. Syst. Sci.* 79 (5) (2013) 586–595.
- [6] C.W. Hsu, A comparison of methods for multiclass support vector machines, *IEEE Trans. Neural Netw.* 13 (2) (2002) 415–425.
- [7] http://en.wikipedia.org/wiki/Support_vector_machine, last accessed January 2013.
- [8] V.N. Vapnik, *The Nature of Statistical Learning Theory*, second ed., Springer, New York, 1995.
- [9] J.H. Min, Y.C. Lee, Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters, *Expert Syst. Appl.* 28 (4) (2005) 603–614.
- [10] A. Bose, X. Hu, K.G. Shin, T. Park, Behavioral detection of malware on mobile handsets, in: *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, 2008, pp. 225–238.
- [11] M. Egele, C. Kruegel, E. Kirda, H. Yin, D. Song, Dynamic spyware analysis, in: *Proceedings of USENIX Annual Technical Conference, ATC'07*, 2007, pp. 233–246.
- [12] K. Rieck, T. Holz, W. Carsten, P. Dussel, P. Laskov, Learning and classification of malware behavior, in: *Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2008, pp. 108–125.
- [13] http://en.wikipedia.org/wiki/Hyperparameter_optimization, last accessed January 2013.
- [14] A. Wichmann, B. Fraunhofer, E. Gerhards-Padilla, Using infection markers as a vaccine against malware attacks, in: *Proceedings of IEEE International Conference on Green Computing and Communications, GreenCom*, 2012, pp. 737–742.
- [15] U. Bayer, A. Moser, C. Kruegel, E. Kirda, Dynamic analysis of malicious code, *J. Comput. Virol.* 2 (2006) 67–77.
- [16] M. Christodorescu, S. Jha, Static analysis of executables to detect malicious patterns, in: *Proceedings of the 12th USENIX Security Symposium*, 2003, p. 12.
- [17] A. Desnos, Androguard, available at <http://code.google.com/p/androguard/wiki/Usage>, accessed 21 May 2013.
- [18] Y. Goldberg, M. Elhadad, splitSVM: fast, space-efficient, non-heuristic, polynomial kernel computation for NLP applications, in: *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics, ACL-08*, 2008, pp. 237–240.
- [19] S.B. Kotsiantis, Supervised machine learning: a review of classification techniques, *Informatica* 31 (2007) 249–268.
- [20] N.S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, *Am. Stat.* 46 (3) (1992) 175–185.
- [21] W. Hamalainen, M. Vinni, Comparison of machine learning methods for intelligent tutoring systems, in: *Proceedings of the 8th International Conference on Intelligent Tutoring Systems, ITS'06*, 2006, pp. 525–534.
- [22] M. Zickus, A.J. Greig, M. Niranjana, Comparison of four machine learning methods for predicting PM10 concentrations in Helsinki, Finland, Water, Air, & Soil Pollution. Focus 2 (5–6) (2002) 717–729.
- [23] X. Ugarte-Pedrero, I. Santos, C. Laorden, B. Sanz, P.G. Bringas, Collective classification for packed executable identification, *Comput. Syst. Sci. Eng.* 28 (1) (2013) 25–36.
- [24] C.L. Huang, M.C. Chen, C.J. Wang, Credit card scoring by support vector machine, *Int. J. Oper. Quant. Manag.* 1 (2) (2004) 155–172.
- [25] E.B. Bernhard, M.G. Isabelle, V. Vapnik, N. Vladimir, A training algorithm for optimal margin classifiers, in: D. Haussler (Ed.), *5th Annual ACM Workshop on COLT*, Pittsburgh, 1992, pp. 144–152.
- [26] S. Gunn, Support vector machines for classification and regression, *ISIS Technical Report*, 1998.
- [27] M.D. Buhmann, *Radial Basis Functions: Theory and Implementations*, Cambridge University Press, 2003.
- [28] S.S. Keerthi, C.J. Lin, Asymptotic behaviors of support vector machines with Gaussian kernel, *Neural Comput.* 15 (7) (2003) 1667–1689.
- [29] H.T. Lin, C.J. Lin, A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods, Technical report, Department of Computer Science & Information Engineering, National Taiwan University, 2003.

- [30] T. Gershoni, M. Mowbray, S. Pearson, Mechanisms for protecting sensitive information in cloud computing, *Comput. Syst. Sci. Eng.* 28 (6) (2013).
- [31] Z.J. Kolter, A.M. Maloof, Learning to detect and classify malicious executables in the wild, *J. Mach. Learn. Res.* 7 (2006) 2721–2744.
- [32] DroidBox, <http://www.honeynet.org/gsoc/slot11>, last accessed February 2013.
- [33] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, *ACM Trans. Intell. Syst. Technol.* 2 (3) (2011) 1–27.
- [34] Java Agent DEvelopment Framework, <http://jade.tilab.com/>, last accessed March 2012.
- [35] <http://www.ntop.org/get-started/download/>, last accessed April 2013.
- [36] <http://contagioexchange.blogspot.tw/>, last accessed January 2013.
- [37] Dr.Web Live, available at <http://live.drweb.com/>, last accessed 9 January 2013.
- [38] Protégé, <http://protege.stanford.edu>, last accessed October 2012.
- [39] T. Blasing, L. Batyuk, A.D. Schmidt, S.A. Camtepe, S. Albayrak, An android application sandbox system for suspicious software detection, in: *Proceedings of 5th International Conference of Malicious and Unwanted Software, MALWARE*, 2010, pp. 55–62.
- [40] VirusTotal, <https://www.virustotal.com/en/>, last accessed March 2014.