

ECF – EVENT CORRELATION FOR FORENSICS

Kevin Chen[†]
Andrew Clark[†]
Olivier De Vel[‡]
George Mohay[†]

[†]Information Security Research Centre
Queensland University of Technology
Brisbane, Queensland
Email: {k.chen, a.clark, g.mohay}@qut.edu.au

[‡]Defence Science Technology Organization
Email: olivier.DeVel@dsto.defence.gov.au

Abstract

The focus of the research described in this paper is on the nature of the event information provided in commonly available computer and other logs and the extent to which it is possible to correlate such event information despite its heterogeneous nature and origins. The strategic purpose of the research has been to develop a means by which a consolidated repository of such information can be constituted and then queried in order to provide an investigator with post hoc event correlation for forensics purposes (ECF). The paper provides an account of the log processing techniques utilized, and the nature of the database and query engine that have been developed in our current prototype and two examples of scenarios investigated and identified by the current prototype.

Keywords

Event correlation, computer forensics, logs, events, heterogeneous event logs

INTRODUCTION:

The research described in this paper focuses on the nature of the event information provided in commonly available computer and other logs (such as system event logs, audit logs, door logs etc) and the extent to which it is possible to correlate such event information despite its heterogeneous nature and origins. The strategic purpose of the research has been to develop a means by which a consolidated repository of such event information can be constituted and then queried in order to provide an investigator with post hoc event correlation for forensics purposes (ECF). The research is part of an on-going collaboration with the Defence Science and Technology Organization of Australia in the area of computer forensics.

Event monitoring and event correlation have become fashionable terms and there are some quite comprehensive products on the market that intend to provide either or both of these. However they are in some cases operating system specific and in any case typically focus on network event correlation and/or centralized event monitoring and log management rather than post hoc correlation of events in general for forensic purposes. See for instance: NetForensics (2003), GuardedNet (2003), e-Security Inc (2003), GFI Software (2003), Flowerfire (2003), TNT Software (2003). In any case, a recent report from the Institute for Information Infrastructure Protection (I3P 2002) identifies a market gap with regard to developments in this area - that products/tools/services appear to be either unavailable or inadequate in the area of event correlation software capable of automatically collecting data from security appliances, routers, and servers and performing data mining.

There is a considerable body of recent research in this area of security event correlation, ranging from work in IDS (Intrusion Detection Systems) alert correlation (Morin et al. 2002) through to the standardization and formatting of audit or log records (Bishop 1995, CERIAS 2003a) and audit reduction (CERIAS 2003b). The singular purpose of both the research and the commercial offerings is to aid computer security i.e., to assist in intrusion detection and in countering or identifying computer attacks, their purpose is not the identification of activity scenarios that are of general forensic interest. Reinforcing this point is that less than 1% of all log data (Bird 2002) relates to security events, presenting a filtering problem for security analysis tools, and indicating the vast volumes of additional data which is grist for the forensic mill.

The research described in this paper and the ECF software that we have developed provides *inter alia* the following kinds of forensic trace information:

Chen, Clark, De Vel, Mohay (Paper #11)

1st Australian Computer, Network & Information Forensics Conference 2003
25 November 2003, Perth, Western Australia

Page 1

- host based activity tracing for objects and principals,
- activity profiles for principals, objects and applications across hosts,
- identification of unusual or anomalous activity relative to such profiles, and
- the correlation of activity traces e.g., "User A accessed files X, Y and Z - and so too did user B on the same day".

Future work on extensions to the current *ECF* system will in addition make use of data mining techniques applied to large volumes of log information in order to provide rule and pattern extraction.

An overriding objective in the research and development has been that of extensibility: it must be relatively straightforward to incorporate new types of event and transactional logs and to accommodate log changes and configurable log formats that may vary from system to system and organization to organization. The *ECF* system must provide an extensible framework for the analysis and correlation of heterogeneous event and transaction records.

The remainder of this paper is organized as follows. Section 2 addresses *ECF* research issues, *ECF* system requirements and design, and the development of a prototype implementation of that design. Section 3 presents the user interface to *ECF* and a summary of its functionality while Section 4 outlines a number of 'scenarios' – event sequences that may typically be of interest to an investigator – and how the *ECF* software identifies these scenarios and provides investigators with the means to query and investigate the nature of the events and the circumstances surrounding them. Section 5 summarizes the results of the research and identifies some challenges to be met and further work required to extend the *ECF* software.

ECF SYSTEM REQUIREMENTS AND DESIGN:

In this section we give an overview of the research issues, system requirements and design aspects that have arisen to date in the research and the design and implementation of *ECF*.

The design requirements were essentially to specify a software architecture that would support development of an extensible framework for event correlation across heterogeneous logs and systems for forensic purposes. The objective of the prototype implementation was to provide proof of concept of the design and to provide a software package that would achieve heterogeneous event correlation in order to identify activity scenarios of interest.

We first describe the structure of the database utilized in the prototype.

The Canonical Form and Database Tables:

An early and fundamental design issue to be resolved in our research was whether to use a canonical form for events emanating from different logs and systems or whether instead to present a canonical view via processors which interface directly to the raw logs themselves. The decision was made to opt for the former viz., for a canonical representation of what is essentially the minimal or base set of fields which appear in all event records viz., *Time*, *Subject*, *Action*, and *Object*. These fields map intuitively to almost all event records no matter how varied the events or systems on which they occur, and so a single database table (the *Canonical* table) which represents that information is consequently the central information store used for event correlation in *ECF*.

Notwithstanding the above, event records can include a considerable variety of event information in addition to the above quadruple (*Time*, *Subject*, *Action*, and *Object*) – additional information which is of considerable potential value e.g., in the case of an event which involves access to a file *Object* – is the file text or image, is it local or remote, is it big or small?

As a result, the decision was made to also allow a second level of interrogation via additional *log-specific* database tables. These tables represent log-specific information which is additional to the canonical information and is in each case idiosyncratic to a particular (type of) log. (This additional capability does not figure in the evaluation and scenarios presented in Section 4.)

In fact, the canonical representation finally adopted in our prototype comprises eleven fields, not four, and this relates to the need for additional qualifying information to identify different subclasses of *Subject* and *Object*, and to the need to represent – where relevant – the outcome or *Result* of an event. In addition, the second-last and third-last listed fields below relate to *hypothesis testing*, something we return to later in this section. The *Canonical* table representation currently comprises the following fields:

<i>Eventid:</i>	<i>Primary key for the event</i>
<i>Time:</i>	<i>The date and time of the event</i>
<i>Subject:</i>	<i>The instigator of the action</i>
<i>SubjectType:</i>	<i>The instigator type (e.g., account_name, url, ip_address)</i>

Object: The object being manipulated
ObjectType: The object type (e.g., url, ip_address)
Action: The action being performed
Result: The outcome of the action (Success, Failure, Unknown)
enteredOnline: Event information entered online, not by offline parser (Boolean)
excluded: Event to be excluded from current query set (Boolean; set/reset by user)
Logid The key into the Logrecord table which identifies the particular log from which this record derives

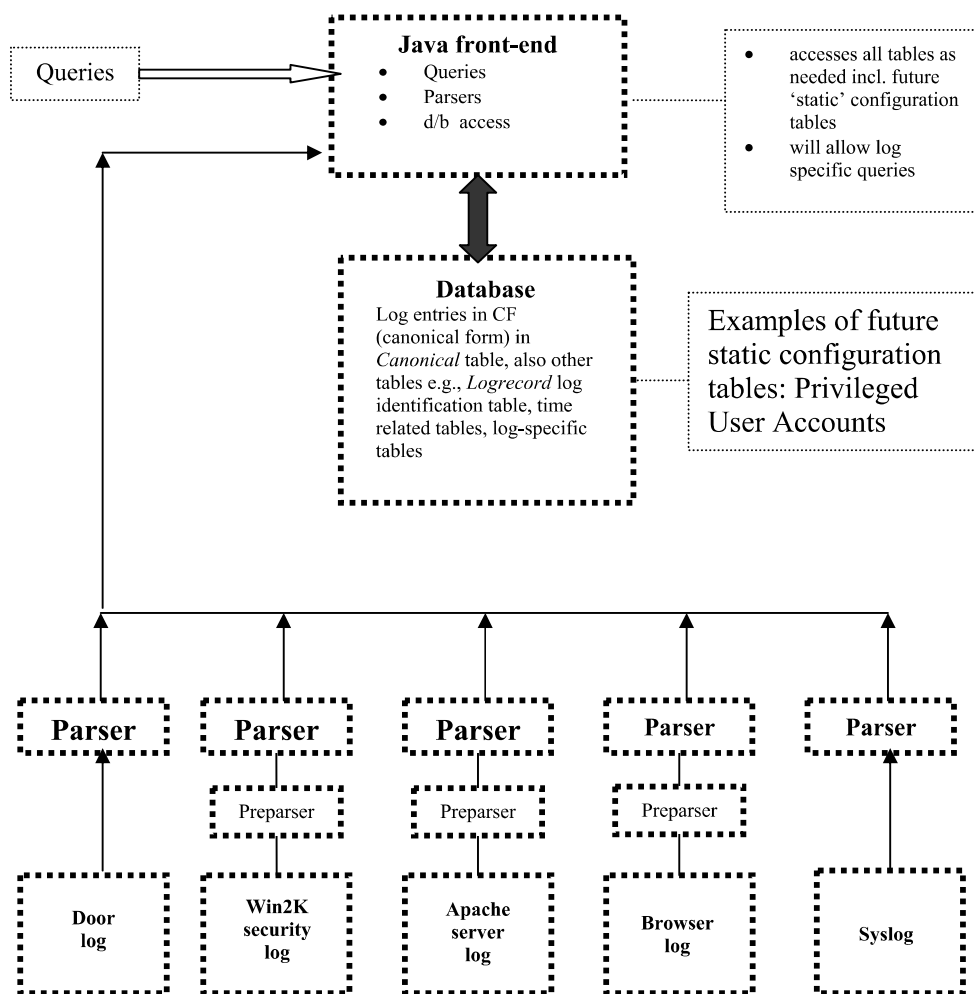


Figure 1: The ECF Architecture

The five fields *Subject*, *SubjectType*, *Object*, *ObjectType*, and *Action* are stored as character strings thus facilitating the querying of those fields by allowing not only exact match searches but also 'contains' or substring match searches.

The database consists of four primary tables, *Canonical* and *Logrecord* plus two other event-time related tables for compatibility with DSTO's CFIT software and CFIT treatment of time. The *Logrecord* table has the following entries each of which describes a specific instance of a log which has been parsed and entered into the *Canonical* table:

LogId: primary key
Lognode: the ID of the host from which the log was collected
LogDatetime: time log was inserted into the database
LogFilename: name of the file containing the logs read by the parser

LogType: the type of log, e.g., Browser, Apache, Windows 2000 Security
LogTimezoneOffset: difference in time (milliseconds) between UCT and time where logs were collected from.

As foreshadowed earlier, *ECF* also provides for other tables relating to log-specific event information to be included, in order to enable additional rich but idiosyncratic log-specific event information also to be included in the database and to be queried. This is information additional to the quadruple of *Time*, *Subject*, *Object* and *Action*. *ECF* currently includes such additional log-specific information for web browser caches and *NIX *syslog sendmail* and *smap* logs. Currently being investigated is whether future versions of *ECF* should utilise an XML based approach to such additional log-specific information; this would not affect the handling of the existing base information (*Time*, *Subject*, *Object* and *Action*). The query mechanisms by which log-specific information is to be queried is the subject of current work.

The *ECF* system architecture is illustrated in Figure 1.

Time

Parsing of the time field of a log and insertion of the corresponding time value into the database takes place in three phases. Phase 1 involves extracting the date and time value from the log and converting it into a formatted string viz., YYYY-MM-DD HH:MM:SS. Phase 2 involves converting the date time string into a Java type, in particular into a Java SQL *TimeStamp*. The third and final phase involves extracting the UTC time from the actual Java *TimeStamp*. This time is expressed as the actual milliseconds from start of epoch (i.e., Jan 1 1970) but is relative to the local time zone. Identification of the local timezone is then used to determine the *timezoneoffset*, which is a millisecond value. This is subtracted or added to the UTC time millisecond value above to yield the corrected UTC time. The stringified representation of this is inserted into the *Time* field of the Canonical table while the binary representation is inserted into one of the two event-time related tables as *starttime* and *stoptime*. That table is keyed by *Eventid* and lists the event times of all events in the *Canonical* table. All querying is based on the *starttime* value.

Hypothesis Testing

It was decided for a number of reasons to incorporate an *online event entry* feature which allows the user to enter events one at a time into the database. These can be *bona fide* events or they can represent hypotheses or ‘what if’ propositions to be tested viz., ‘what if’ Person P was logged in twice, the second time on another machine, would that affect the conclusion returned by the query engine? The feature also provides a powerful system test feature by allowing test case event scenarios to be easily deployed into the database. The current *ECF* software provides the necessary functionality required for incorporation of all of the relevant logs (identified in parentheses above) into the database. However, it is a non-trivial task to set up and deploy the actual scenario above in real-time and as such the online event entry is useful for preliminary testing. Setting up and deploying the scenario in real time is of course necessary for comprehensive system testing and that was indeed carried out in the testing discussed in Section 4 *Scenarios and Evaluation*.

Hypothesis testing also requires event negation, that is negation of an event that appears in the database e.g., “what if event X had not occurred?”. This is supported in the *ECF* software by providing support for *EventExclude* and also *EventUnexclude* functions.

ECF USER INTERFACE:

The *ECF* design elaborated in the previous section has been implemented in Java using a Postgres database accessed via JDBC. The GUI front-end provides the following five modes of operation which we discuss in detail below:

- (i) Log Parse
- (ii) Dynamic Query
- (iii) Custom Query (for testing purposes only)
- (iv) Hypothesis Testing
- (v) (Access to) Log Specific Information

Log Parse

This provides the interface to all log parser software which in addition to parsing the logs (in some cases pre-parsed), inserts the event canonical form representation into the database. This involves insertion of the base or canonical event information into the *Canonical* table, and insertion of related information into the Logrecord and time-related tables. In some cases (this currently applies only to browser cache and *syslog sendmail* and *smap* logs), database entry involves creation of a corresponding log-specific table.

Currently, the front-end provides parsers for:

- Apache Server Log

Chen, Clark, De Vel, Mohay (Paper #11)

1st Australian Computer, Network & Information Forensics Conference 2003
25 November 2003, Perth, Western Australia

Page 4

Trend Micro, Inc.
EX1006-4

- Windows 2000 Security Log
- Door Log
- Browser Cache Logs
- *NIX SYSLOG - POP3, SSH, sendmail, smap

The syslog parsers make use of the Java implementation of regular expressions. It is expected that future parsers will likewise be regular expression based and we are reworking some of the other parsers to use the same approach. This will facilitate extensibility with regard to accommodating new logs or re-formatted versions of existing logs.

Pre-parsing is needed for the following logs and simply involves identifying and delimiting the various fields within each event record:

- Windows 2000 security logs
- Apache server logs
- Browser logs

Dynamic Query

Dynamic queries are satisfied largely by access to the main database table, the *Canonical* table with processing of time values relying upon the field *starttime* in one of the two event-time related tables. The primary key *Eventid* uniquely identifies each event across all tables.

As described above, there are four main database tables plus log-specific tables for representation of event information which is not accommodated in the main *Canonical* table. In future, there will also be other tables added to represent configuration information of the systems from which the raw logs have been derived.

In presenting a query, the operator has the option of selecting a subset of the *Canonical* fields to be displayed in the results by setting the appropriate checkboxes.

The query is built by selecting various constraints and relating them with AND and/or OR relationships. For example, if both *Time* and *Subject* are constraints, using the AND relationship means both *Time* and *Subject* constraints need to be satisfied, using the OR relationship means that the result set should include events that satisfy either constraint.

When a constraint is selected, the application populates the related drop down menu with all corresponding values currently in the database and this may then be used to simply choose the desired value as the constraint. The constraints can also be set to match a sub-string. For example, if *SubjectType* is set to *ip_address* and *Subject* is set to "contains 131.181" then all IP addresses in the 131.181.0.0 subnet will be matched and only records satisfying that constraint will be displayed (subject to other constraints also).

Custom Query

The Custom Query tab accepts an SQL query from the operator and displays the results in a table. It provides a means to execute more flexible queries than can be built using the Dynamic Query tab option.

Hypothesis Testing

To support 'hypothesis testing' as described in the previous section, and also to assist our testing procedures, the Hypothesis Testing tab provides an online event entry feature to allow the user to insert additional events online. (Note that we have purposely not been concerned with authentication or authorization.) In addition, to further support these same objectives, the software also implements *eventExclude* and *eventUnexclude* features:

- *eventExclude* allows the user to flag specified events to be logically excluded from the database query set;
- *eventUnexclude* allows the user to reverse the effects of an *eventExclude* operation.

Access to log specific information

As previously discussed, *ECF* also provides for event information other than that inserted in the *Canonical* table to be included in the database i.e., event information that is additional to the quadruple of *Time*, *Subject*, *Object* and *Action*. *ECF* currently includes such additional log-specific information for web browser caches and *NIX syslog *sendmail* and *smap* logs. The query mechanisms by which log-specific information of this sort is queried is the subject of current work.

SCENARIOS AND EVALUATION:

The two scenarios described in this section have been identified to demonstrate the capabilities of the ECF software. These scenarios show how the software can be used to test for a suspected sequence of events. The two identified scenarios are:

- (i) Download and distribution of porn/warez via email; and
- (ii) Download and execution of an exploit allowing elevated privileges on a Windows 2000 workstation.

For each of these scenarios a hypothesised sequence of events was devised, and queries to the database were performed (using the Dynamic Query interface) to extract canonical data that supports the hypothesis. The starting point used by an investigator will depend on the clues or suspicions they have at the outset. In each case we begin with a logical clue and develop database queries to piece together the scenario described. Each scenario is now presented in detail.

Scenario 1: Porn/Warez Distribution:

The following sequence of events could support the hypothesis that a particular person downloaded and distributed via email a particular file. The appropriate logs for identifying the listed actions are given in parentheses.

- (i) Person P enters room R (*door log*)
- (ii) P logs on to Windows 2000 workstation W (*Windows 2000 Security log*)
- (iii) P logs on to remote server S using an SSH client application (*syslog – ssh log*)
- (iv) P downloads a pornographic image file F from an Apache web server A using a web browser B running on server S (*Apache web server log; and browser log*)
- (v) P emails F to Person Q (*syslog – sendmail log*)

As shown above, the ECF tool currently supports all of these log types. This scenario was carried out on our test network. During experimentation with this scenario it was found that the Netscape browser cache did not store certain files, perhaps based upon their type or size. This makes it more difficult to perform a 'warez' type scenario where software is distributed illegally (since software is usually distributed either compressed or as executable files which are large in size). In this case we used the porn scenario and set up a web site containing HTML and image (GIF) files with suggestive names. The image file used for testing this scenario was relatively large (~500kb), and was not stored in the browser cache by Netscape (version 4). However, the Apache logs obtained from the web server clearly show the image file being retrieved from the suspect workstation.

After performing each of the steps listed in the scenario description, each of the required logs was obtained, and parsed by the ECF tool, for storage in its (PostgreSQL) database.

An investigator working on a porn distribution case (such as distribution of paedophilic material, or a sexual harassment case) might feasibly consider, as a starting point, searching logs for keywords (such as 'porn', 'sex', 'xxx', and so on) that may indicate activity associated with pornographic content. The 'porn site' used in our scenario (a simulated one with no pornographic content, just files with suggestive names) contained an HTML file 'porn.html' with an associated image file 'porn.gif'. The following SQL query, searching for records with an *Object* field containing the text 'porn' was first performed. (This query was constructed by the ECF tool through its Dynamic Query interface.)

```
SELECT * FROM CANONICAL WHERE OBJECT ilike ( '%porn%' );
```

The query above returned one record which originated from the browser cache, and four records which originated from the Apache web server log. In order to save space we do not show the results of this query here. An investigator may not have had the web server logs at the outset but could have requested them after finding the entry in the browser cache. The record returned from the browser cache also includes the UNIX username of the account from which the browser was run (chenk), and the time at which the event occurred (2003-08-08 03:54:36 UCT).

The next query we perform is aimed at determining activities by the user 'chenk' in the time window which commences ten minutes prior to the browser event, and finishes ten minutes following that event. By searching for a *Subject* which contains the string 'chen', within that time window, we obtain the results shown in Figure 2.

Query Results (62 rows total, 13 rows shown, 49 rows hidden, 49 rows excluded)

entryid	rawtime	subject	subjecttype	object	objecttype	action	result	lognode	enter...	exclu...
9735	2003-08-08 03:48:00	KEVIN CHEN...	username	GP. S BLOCK RM S826A	door	open	success	gut security	FALSE	FALSE
9771	2003-08-08 03:51:32	ISRC chen	username	528 ISRC/BND # Logon ID: (0x0,0x23F41F)	GENERIC...	Logon	unknown	bnd	FALSE	FALSE
9878	2003-08-08 03:51:53	chen	username	needham/needham	computer	ssh - login	success	bnd	FALSE	FALSE
10830	2003-08-08 03:52:30	chen	username	http://cdn.netscape.com/_media/wpsite/pop...	url	get	success	needham	FALSE	FALSE
10835	2003-08-08 03:52:32	chen	username	http://cdn.netscape.com/wpton_03/aolforbro...	url	get	success	needham	FALSE	FALSE
10846	2003-08-08 03:52:33	chen	username	http://cdn.netscape.com/wpt_tonr_03/20030...	url	get	success	needham	FALSE	FALSE
10847	2003-08-08 03:52:37	chen	username	http://cdn.netscape.com/wpton_03/dot_d2c9...	url	get	success	needham	FALSE	FALSE
10836	2003-08-08 03:52:37	chen	username	http://cdn.netscape.com/wpton_03/gradient	url	get	success	needham	FALSE	FALSE
10831	2003-08-08 03:52:40	chen	username	http://home.netscape.com/_ads/adsPopup2...	url	get	success	needham	FALSE	FALSE
10832	2003-08-08 03:53:45	chen	username	http://www.microsoft.com/library/include/ctre...	url	get	success	needham	FALSE	FALSE
10837	2003-08-08 03:53:45	chen	username	http://www.microsoft.com/library/toolbar/ima...	url	get	success	needham	FALSE	FALSE
10848	2003-08-08 03:54:18	chen	username	http://www-int.isrc.qut.edu.au/images/bg2.jpg	url	get	success	needham	FALSE	FALSE
10849	2003-08-08 03:54:36	chen	username	http://www-int.isrc.qut.edu.au/dev/ac/porn.html	url	get	success	needham	FALSE	FALSE

Exclude Entries Un-exclude Entries Show Excluded Entries

Figure 2: Results of query searching for records whose Subject contains the substring 'chen'.

The query used to obtain the results shown in Figure 2 is as follows:

```
SELECT * FROM CANONICAL WHERE TIME >= '2003-08-08 03:44:36' AND TIME <= '2003-08-08 04:04:36' AND SUBJECT ilike ( '%chen%' );
```

Note that this query also extracted a number of Windows 2000 logs, of which 49 are not relevant and have been 'excluded' from the results shown in Figure 2. A future enhancement to the *ECF* tool will allow a query to limit the Windows 2000 records shown to only 'Logon' type records. The remaining records show that someone with the magnetic swipe card issued to 'KEVIN CHEN' entered room S826A approximately 6½ minutes prior to the suspicious browser log record containing the pornographic content. The Windows 2000 workstation was then logged on to using the domain account 'ISRC\chenk'. Following that the user 'chenk' logged on to the server 'needham' using an SSH client. The remaining records returned from this query relate to the browser activity of the user 'chenk', culminating in the retrieval of the 'porn.html' URL.

Note that the combination of the results could have been obtained in a single query if the *ECF* tool allowed more complex queries to be formulated. We are currently planning to enhance the interface to allow such queries to be made. The Custom Query interface could also have been used in this instance with an SQL 'WHERE' clause such as: 'WHERE (TIME >= t1 AND TIME <= t2) AND (SUBJECT ilike ('%chen%') OR OBJECT ilike ('%porn%'))';.

If the investigator suspects that the pornographic content is being distributed via email, then the next logical step is to search the email logs for evidence of such activity. The *ECF* tool was used to generate a query which extracted all of the mail logs recorded in the 20 minute window described above. There were 19 such logs obtained in our test. The query used was as follows:

```
SELECT * FROM CANONICAL, LOGRECORD WHERE CANONICAL.LOGID = LOGRECORD.LOGID
CANONICAL.TIME >= '2003-08-08 03:44:36' AND CANONICAL.TIME <= '2003-08-08 04:04:36'
AND LOGRECORD.LOGTYPE = 'SENDMAIL Log (SYSLOG)';
```

In order to save space we do not show the results of this query here. Of the 19 records obtained from this query, 3 were related to the delivery of an email containing the image file 'porn.gif'. The *sendmail (SYSLOG)* records parsed by the *ECF* tool currently do not allow the sender of an email to be identified. Thus, it is difficult to make the link between the downloading of the pornographic material and its subsequent distribution by email. Future enhancements to the *ECF* tool will allow it to query log-specific information, such as file size, which would assist in this case since both browser logs and mail logs contain size information which could be linked to strengthen the relationship between these two steps in the scenario. The three email log records associated with the delivery of the image file in this scenario show the recipient of the email who, if possible, could be the subject of further enquiries by the investigator.

Scenario 2: Operating System Exploit

The following sequence of events could support the hypothesis that a particular person downloaded and executed an exploit against a computer, and later gained elevated privileges on that computer. In this example we assume the exploit allows the user to reset the administrator password to a known value. Once again, the appropriate logs for identifying the listed actions are given in parentheses.

- (i) Person P enters room R (*door log*)
- (ii) P logs on to Windows 2000 workstation W (*Windows 2000 Security log*)
- (iii) P downloads exploit file F from Apache web server A (*Apache web server log*)
- (iv) P executes the downloaded file F on host W (*Windows 2000 Security log*)
- (v) Workstation W is rebooted (*Windows 2000 Security log*)
- (vi) Administrator logs on to workstation W a short time later (*Windows 2000 Security log*)

This scenario relies on only three log types: the door log; the Windows 2000 security log; and the Apache web server log. As with the previous scenario, this scenario was carried out in our test environment. The browser logs located on workstation W could be used to gain additional information about the different sites that the user visited, however we have not included those logs in this particular scenario as the browser cache did not contain a copy of the exploit file (due to restrictions on the type/size of files stored in the cache by browsers, as described above).

In this case the investigator may have very little idea about the exact nature of the scenario above, but will most likely become suspicious upon noticing that the Administrator password has changed. This discovery may not occur until some time after the exploit was actually carried out. Of course an intelligent adversary would probably clear the Windows 2000 security log after having carried out their activities but we assume that, in this case, we are not dealing with someone so clever, or perhaps forensic analysis of the workstation's hard disk allowed a portion of the pre-wiped security log to be recovered.

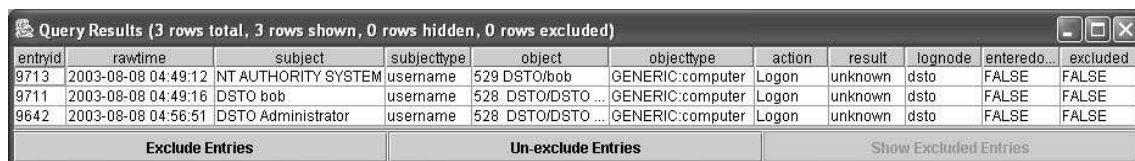
The investigator would presumably be interested in knowing times when the administrator account on the workstation was logged in to. The ECF tool Dynamic Query interface was used to generate the following query which will return a list of records relating to logins to the 'administrator' account on the workstation in question ('dsto'):

```
SELECT * FROM CANONICAL, LOGRECORD WHERE CANONICAL.LOGID = LOGRECORD.LOGID AND
CANONICAL.SUBJECT ilike ( '%administrator%') AND CANONICAL.ACTION = 'Logon' AND
LOGRECORD.LOGNODE = 'dsto';
```

This query returned a single record from our test logs which we do not show here due to space limitations. However, the record returned occurred at the time '2003-08-08 04:56:51' (UCT). The investigator would also be interested in other accounts that had been used to login to the workstation at around the same time. Therefore, in a similar fashion to the previous scenario we perform a query to determine all the users who logged onto the workstation 'dsto' in the time window commencing ten minutes prior to the identified records, and finishing ten minutes after it. The following SQL query, which was generated using the Dynamic Query interface, extracts the desired records:

```
SELECT * FROM CANONICAL, LOGRECORD WHERE CANONICAL.LOGID = LOGRECORD.LOGID AND
CANONICAL.TIME >= '2003-08-08 04:46:51' AND CANONICAL.TIME <= '2003-08-08 05:06:51'
AND CANONICAL.ACTION = 'Logon' AND LOGRECORD.LOGNODE = 'dsto';
```

This query returned the three entries shown below in Figure 3¹. Note that the Windows 2000 security audit system creates multiple events when a normal user logs on (observe the event ID's 528 and 529 in the *Object* field). The results show that the user DSTO\bob logged in approximate seven and a half minutes prior to the administrator account being used. The investigator would presumably then be interested in discovering more about the activities of that user in the same time-frame.



Query Results (3 rows total, 3 rows shown, 0 rows hidden, 0 rows excluded)

entryid	rawtime	subject	subjecttype	object	objecttype	action	result	lognode	entered...	excluded
9713	2003-08-08 04:49:12	NT AUTHORITY SYSTEM	username	529 DSTO/bob	GENERIC:computer	Logon	unknown	dsto	FALSE	FALSE
9711	2003-08-08 04:49:16	DSTO bob	username	528 DSTO/DSTO ...	GENERIC:computer	Logon	unknown	dsto	FALSE	FALSE
9642	2003-08-08 04:56:51	DSTO Administrator	username	528 DSTO/DSTO ...	GENERIC:computer	Logon	unknown	dsto	FALSE	FALSE

Exclude Entries Un-exclude Entries Show Excluded Entries

Figure 3: Results of query searching for logons around the time of the 'administrator' logon.

In fact, we perform a query which shows all of the activity that occurred on the workstation in the chosen time window:

```
SELECT * FROM CANONICAL, LOGRECORD WHERE CANONICAL.LOGID = LOGRECORD.LOGID AND
CANONICAL.TIME >= '2003-08-08 04:46:51' AND CANONICAL.TIME <= '2003-08-08 05:06:51'
AND LOGRECORD.LOGNODE = 'dsto';
```

This query returns 122 records which we do not show here. Through a manual inspection of these logs we notice that the workstation restarted at '2003-08-08 04:56:28'. We still don't really have any clues as to why the workstation may have been rebooted. We now decide to investigate other logs available to us in order to try to gain more information above the activities of 'bob' in the chosen time period. A similar query is performed, this time without the LOGNODE = 'dsto' restriction. Figure 4 below shows the results of the query with all rows hidden except those associated with the files downloaded from the web server by the 'dsto' workstation. We see that the workstations IP address (131.181.6.167) appears in the Apache logs and observe that a number of suspicious looking files have been downloaded by 'bob'.

¹ Note that the *Result* value of 'unknown' in this table is due to a minor bug in the ECF Windows 2000 security log parser which will be fixed prior to publication, should this paper be accepted. The *Result* field should contain the value 'success' in this case.

Query Results (133 rows total, 11 rows shown, 122 rows hidden, 122 rows excluded)

entryid	rawtime	subject	subjecttype	object	objecttype	action	result	lognode	enteredonline	excluded
9591	2003-08-08 04:50:59	131.181.6.167	ipaddress	/poweredby.png	url	get	success	cruncher1	FALSE	FALSE
9592	2003-08-08 04:50:59	131.181.6.167	ipaddress	/icons/apache_pb.gif	url	get	success	cruncher1	FALSE	FALSE
9593	2003-08-08 04:50:59	131.181.6.167	ipaddress	/	url	get	success	cruncher1	FALSE	FALSE
9590	2003-08-08 04:51:20	131.181.6.167	ipaddress	/bd030426.bin	url	get	success	cruncher1	FALSE	FALSE
9587	2003-08-08 04:51:23	131.181.6.167	ipaddress	/poweredby.png	url	get	unknown	cruncher1	FALSE	FALSE
9588	2003-08-08 04:51:23	131.181.6.167	ipaddress	/icons/apache_pb.gif	url	get	unknown	cruncher1	FALSE	FALSE
9589	2003-08-08 04:51:23	131.181.6.167	ipaddress	/	url	get	unknown	cruncher1	FALSE	FALSE
9586	2003-08-08 04:51:28	131.181.6.167	ipaddress	/rawrite2.exe	url	get	success	cruncher1	FALSE	FALSE
9583	2003-08-08 04:51:34	131.181.6.167	ipaddress	/poweredby.png	url	get	unknown	cruncher1	FALSE	FALSE
9585	2003-08-08 04:51:34	131.181.6.167	ipaddress	/	url	get	unknown	cruncher1	FALSE	FALSE
9584	2003-08-08 04:51:34	131.181.6.167	ipaddress	/icons/apache_pb.gif	url	get	unknown	cruncher1	FALSE	FALSE

Exclude Entries Un-exclude Entries Show Excluded Entries

Figure 4: Results of query searching for other activity in the selected time region.

The files 'rawrite2.exe' and 'bd030426.bin' actually relate to the exploit in this case. The 'rawrite2.exe' utility is a tool for copying bootable floppy disk images to a floppy disk. The image file, in this case 'bd030426.bin',

contains a utility for wiping the Windows administrator password (on vulnerable Windows 2000 hosts). These logs give us a very strong indication of what actually happened in this case.

Note that it is our intention that the *ECF* tool will eventually be able to make many of the inferences that we currently perform manually, by itself. It is envisaged that data mining approaches, combined with the ability to relate the additional log-specific information between the different log types, will allow the tool to extract only the most relevant logs, leaving extraneous ones out. This problem is analogous to the current problem with intrusion detection systems which can generate very large amounts of alerts, many of which are not relevant.

CONCLUSION:

The results presented in Section 4 above indicate that cross-correlation of event information emanating from heterogeneous logs potentially provides a powerful basis for the forensic identification of scenarios of interest even though only a very small base of common denominator information is being taken into account. Clearly, the inclusion of additional information from various log sources can only expand the potential usefulness of the technique. The present system recognizes scenarios via single-event queries presented by an investigator, what is needed next is to develop a capability for recognizing combinations of events or event patterns.

We are now undertaking further work in four related areas:

- the fully automated recognition of scenarios of interest: this ultimately needs a scenario description or event pattern specification language and a pattern recognition capability within *ECF* as a means of matching scenario descriptions (event pattern specifications) against stored events;
- the incorporation into *ECF* of query mechanisms by which log-specific information in addition to the base information (*Time*, *Subject*, *Object* and *Action*) is queried and correlated with information from the main *Canonical* table and the possible use of an XML based approach to do this (this feature will expand the range of queries possible and provide a 'drill-down' functionality into the front end);
- the development of a hierarchic event model which incorporates event specification, and event abstraction and composition; and (eventually)
- the incorporation of rule or pattern extraction into *ECF* for scenario discovery.

Event patterns of interest include both event sequences and event groups, where the events are characterized by event type and event attributes (the latter includes not only content but possibly other event attributes also). In most cases, it is the temporal relationship between events that is typically paramount. Some event correlation languages have been criticized for catering for too many temporal relationships (Zhu and Sethi 2001) and it will be important to identify the minimal set of such relationships required for the purposes of this project e.g., happened before, coincident with, overlapping, within a window of time, etc. There are a number of event models and languages that may be suitable including SEL (Zhu and Sethi 2001), Lambda (Cuppens and Mieke 2002), Complex Event Processing or CEP (Perrochon *et al.* 2002) and STATL (Eckman *et al.*)

The stored event database includes provision for the inclusion of state information (e.g., is/was User X a privileged user?) so that a rich set of preconditions is available – based not only on other events, but also on state or context information. This in turn relates to event abstraction and event composition, the third area for further work noted above. Event composition and abstraction is needed to assist in making meaningful correlations to be drawn between events which may originate from logs operating at distinctly different levels of granularity e.g., alerts emanating from a network packet filter or IDS on the one hand and events reported in a *Windows 2K* log or in a *Unix syslog* log on the other.

REFERENCES:

- Bird T. (2002) *Finding the Forest in Your Trees: What Audit Logs can Tell You about Your Network*, http://www.counterpane.com/log_short.pdf, Accessed August 2003
- Bishop M. (1995) *A Standard Audit Trail Format*, Proceedings of the 18th National Information Systems Security Conference, pages 136--145, October 1995
- CERIAS, Purdue University (2003a) *Audit Trails Format Group*, <http://www.cerias.purdue.edu/about/history/coast/projects/audit-trails-format.php>, Accessed August 2003
- CERIAS, Purdue University (2003b) *Audit Trail Reduction Group*, <http://www.cerias.purdue.edu/about/history/coast/projects/audit-trails-reduce.php>, Accessed August 2003
- Cuppens F. and Mieke A. (2002) *Alert Correlation in a Cooperative Intrusion Detection Framework*, Proc 2002 IEEE Symposium on Security and Privacy
- Eckmann, S. T., Vigna, G., and Kemmerer, R. A. (2000). *STATL: An Attack Language for State-based Intrusion Detection*. In Proceedings of the ACM Workshop on Intrusion Detection.
- e-Security Inc. (2003) *e-Security Management System*, <http://www.esecurityinc.com/>, Accessed August 2003
- Flowerfire (2003) *Sawmill*, www.sawmill.net, Accessed August 2003
- GFI Software USA, Inc. (2003) *LANguard Security Event Log Monitor*, <http://www.gfisoftware.de/>, Accessed August 2003
- GuardedNet (2003) *neuSECURE*, <http://www.guarded.net/>, Accessed August 2003
- I3P - Institute for Information Infrastructure Protection (2002) *National Information Infrastructure Protection Research and Development Agenda Initiative Report, Information Infrastructure Protection: Survey of Products, Tools and Services*, Sept 9, 2002, Accessed August 2003 http://www.thei3p.org/documents/analyses/I3P_Products_Analysis_V1.0s.pdf,
- Morin, B., Mé L., Debar H., and Ducassé M. (2002) *M2D2: A Formal Data Model for IDS Alert Correlation*, Lecture Notes in Computer Science, Volume 2516 / 2002, *Recent Advances in Intrusion Detection : 5th International Symposium*, RAID 2002, Zurich, Switzerland, October 16-18, 2002. Proceedings.
- NetForensics (2003) *netForensics*, <http://www.netforensics.com/>, Accessed August 2003
- Perrochon, L., et al. (2002) *Enlisting Event Patterns for Cyber Battlefield Awareness*. in DARPA Information Survivability Conference and Exposition, 2000.
- TNT Software (2003) *ELM Log Manager*, <https://www.tnttechnology.com/>, Accessed August 2003
- Zhu D. and Sethi A.S. (2001) *SEL - A New Event Pattern Specification Language for Event Correlation*, Proc. ICCCN-2001, Tenth International Conference on Computer Communications and Networks, Scottsdale, AZ (Oct. 2001), pp. 586-589

ACKNOWLEDGEMENTS

The above work was supported by a continuing collaborative research grant from the Defence Science and Technology Organization (DSTO) of Australia. In addition to the authors, the following people have been involved at various times with the project and their contributions are gratefully acknowledged: Dr Alison Anderson, Rohith Raju, and Tod Thomson.

COPYRIGHT

Chen, Clark, De Vel, Mohay © 2003. The author/s assign the We-B Centre & Edith Cowan University a non-exclusive license to use this document for personal use provided that the article is used in full and this copyright statement is reproduced. The authors also grant a non-exclusive license to the We-B Centre & ECU to publish this document in full in the Conference Proceedings. Such documents may be published on the World Wide Web, CD-ROM, in printed form, and on mirror sites on the World Wide Web. Any other usage is prohibited without the express permission of the authors.