RICH EVENT REPRESENTATION FOR COMPUTER FORENSICS

Bradley Schatz, George Mohay and Andrew Clark

Information Security Research Centre
Queensland University of Technology

{b.schatz , g.mohay, a.clark }@qut.edu.au

ABSTRACT

Recent advances in computer internetworking and continued increases in Internet usage have been accompanied by a continued increase in the incidence of computer related crime. At the same time, the number of sources of potential evidence in any particular computer forensic investigation has grown considerably, as evidence of the occurrence of relevant events can potentially be drawn not only from multiple computers, networks, and electronic systems but also from disparate personal, organizational, and governmental contexts. Potentially, this leads to significant improvements in forensic outcomes but is accompanied by an increase in both the complexity and scale of event information. In order for forensic investigators to effectively investigate this mass of data, semantically strong representational models and automated methods of correlating such event data is becoming a necessity. The contribution of the work described in this paper is the automated detection of a computer forensic scenario, based upon facts automatically derived from digital event logs. We present an expert systems based approach that has the ability to manage the scalability and semantic issues arising in such inter-domain forensics, using an extensible, semantic domain model specified using the Web Ontology Language (OWL). We have developed a prototype system, Forensics of Rich Events (FORE), which supports investigation of heterogeneous event data using a novel form of manipulation of hypothetical knowledge, while supporting the application of standard rule and signature based event correlation techniques. We demonstrate proof of concept of our approach by applying the prototype we have developed to a test case scenario that demonstrates the flexibility of the approach in a single domain context.

Key Words: A.I., Search Algorithms, Information Security

1. INTRODUCTION

Event correlation is a term that is used to describe an array of techniques applied to comprehending the dynamic behavior of systems, based on events and patterns of events in their history. Considerable effort has been expended on event correlation in a number of computer security application domains, in particular in the areas of network management and intrusion detection.

At the same time, developments in computer forensics have seen an increasing reliance upon event logs generated by computer systems as a source of evidence. Recent trends have resulted in considerable growth in both the number of sources of event-oriented evidence, and the volume of events. This is due to the potential for drawing evidence not only from multiple computers, networks and electronic systems, but also from disparate personal, organizational, and governmental domains. This potentially leads to very significant improvements in forensic outcomes but is accompanied by an increase in both the complexity and scale of processing involved.

When comparing the number of security related events to the total number of events logged by modern computer systems, we find that in practice, security related events comprise only something like 1% of logged information (Bird, 2002). This means that there is a huge amount of event log information (the other 99%) which is not related to security but which is available to the computer forensics investigator for use in identifying activities and events of potential forensic interest. In addition, forensic event correlation may consider event logs from other, disparate sources, which are not computer event logs per se, such as electronic door logs, telephone call records, and bank transactions records.

In order for forensic investigators to effectively investigate this mass of data, some means of addressing both the conceptual complexity, and the volume of events is becoming a necessity. Methods of representation search and reasoning with the quite heterogeneous semantics implicit in event-oriented evidence from disparate types of source is a central concern. Automated methods of analyzing and correlating these events are needed for forensic investigation to become a widely used and cost effective process.

Our motivation for this work is to explore methods of forensic investigation of heterogeneous event log based records. We provide for investigation methods such as human guided search, automated correlation, and hypothetical reasoning.

Existing approaches to event correlation have focused on single domains of interest only, and have employed models of correlation very specific in nature. Repurposing existing approaches to the task of more general task forensics is made difficult for a number of reasons. Existing specific models underlying event pattern languages don't necessarily generalise to application in wider domains. For example, while state machine based event pattern languages may work well for events related to protocols, they don't work well for patterns where time and duration are uncertain (Doyle, et al., 2001). Most approaches focus exclusively on events, but ignore context related information such as environmental data and configuration information. Furthermore, few approaches have available implementations in a form that is readily modifiable.

Where we have modifiable implementations, we find that extension is complicated by the software paradigm underlying its implementation. For example, extending the STAT (Eckmann, et al., 2000) language involves considerable burden. Adding new concepts within the semantics of the event language is slowed due to compilation and linkage overheads. Addition of concepts outside of the event pattern language requires reengineering of the STAT language compiler and supporting framework.

We need means to rapidly integrate knowledge from new types of event logs, in a manner that makes explicit the environmental or implicit concepts associate with those logs. This is in order to facilitate human understanding, and also automated inference. A general solution is needed.

Our approach enables this while supporting most rule and signature based correlation techniques, and provides extensibility for new models of event patterns and correlation. We have defined a extensible and semantically grounded domain model, a forensic ontology expressed using the Web Ontology Language (OWL) (McGuinness, et al., 2004). This model represents event or transaction based knowledge as well as environment-based knowledge (i.e. real world information such as people, places, etc, as opposed to event based knowledge).

Due to the richness or abundance of detail in the events we consider, we call our prototype system Forensics of Rich Events (FORE). The system is demonstrated using a scenario consisting of event correlation of events sourced from security related logs in the context of of an intrusion forensics investigation.

2. RELATED WORK

We aim to create a forensics investigation environment that enables separate formulation and incorporation of domain specific concepts as ontologies, and rules that refer to those ontologies, developed by experts in their respective domains. This section reviews work in areas related to this. We further review work in event and event pattern representation, which are a necessary part of event correlation.

2.1 Semantic Web

Recent efforts in the Knowledge Representation and Reasoning (KR&R, or KR) field have concentrated on building the Semantic Web, an extension of the current World Wide Web intended to enable publishing of information in a form that is easily inferable both to humans and machines alike. Current developments have led to the standardisation of the Web Ontology Language (OWL) by the World Wide Web Consortium (W3C). This language provides a means for specifying and defining ontologies – collections of descriptions of concepts within a domain of discourse (classes), properties of classes, and restrictions on properties

OWL's lineage can bee seen to be drawn from the frame based approach to knowledge representation introduced by Minsky (Minsky, 1975), the semi-structured data model, RDF (Miller, et al., 1997) and more recently, a branch of logic called Description Logics. A frame represents an object or concept, and attached to each frame are attributes (or slots) that represent component parts of the concept or object. The Object Oriented paradigm underlying Java and C++ may be seen as the application of frame based theory to the structuring of software (Lassila, et al., 2001). Description Logics (DL), are a subset of First Order Logic (FOL) and are well suited to expressing terminology and instance information, with efficient and decidable inference characteristics. The computational characteristics of FOL on the other hand are intractable (Grosof, et al., 2003).

Documents containing the OWL language is intended to be easily publishable on the Web, where it can be used by applications that process the content of information. Ontologies defined in OWL may import subsets of other ontologies. The language provides support for merging of ontologies, encouraging separate ontology development, refinement and re-use. It is therefore well suited for use in the research upon which we have embarked.

2.2 Event Representation

The representation used to model events has a significant impact on the usability of correlation approaches, including conceptual expressiveness, extensibility, and maintainability. The MODEL language, a component of the DECS network management system, used a frame-based model of events, including concept inheritance (also referred to as semantic generalization) (Yemini, et al., 1996). The event correlator translates from the MODEL language directly to C++, and presumably, is encumbered by the maintainability characteristics of C++ software development and deployment. Expert systems based approaches such as the EMERALD IDS (Lindqvist, et al., 1999) similarly use a frame based knowledge model. However, the model is dynamically constructed at run time, resulting in simpler extensibility and more rapid evolution compared to the DECS approach.

An alternate approach to knowledge representation has been the relational model popularised by the database language SQL. Chen et.al. have in their Event Correlation for Forensics (ECF) research reduced log events into a generalised data model (canonical form) implemented using a relational database (Chen, Clark, et al., 2003) which then allows either interactive or automated scenario identification. This work did not incorporate notions such as causal attributes and semantic generalization in the correlation methods.

A number of challenges were identified with the ECF approach. Identification of a methodology for mapping the detail rich, domain specific information contained in log files to the canonical representation appeared to be elusive. The simplicity of the semantics of the canonical event requires extensive human inference and interpretation in usage. While the mapping process retains the considerable amounts of potentially useful log-specific data as 'shadow data', it does so separate from the canonical form representation and access to that extra data is made more cumbersome. Further, it is unclear how the information model could be extended beyond the limits of the relational model adopted.

Doyle et al (Doyle, et al., 2001), in reviewing the expressiveness of the state of the art in intrusion detection correlation languages, suggests that the CYC (Lenat, 1995) KR&R system provides powerful constructs for reasoning across abstract and concrete concepts across multiple domains. The CYC system provides an extensive model of the world expressed as ontologies expressed in the language CYCL. OWL can be seen to address similar KR goals as CYC. While parts of CYC have recently been made available as open source software, large amounts remain closed. Furthermore, the research literature surrounding the use of CYC is very limited. As the research community is currently very active around the application and implementation of OWL, we have selected it as our knowledge representation system.

Our work is closest to the research performed by Goldman et.al. in their IDS alert fusion prototype, SCYLLARUS (Goldman, et al., 2001). This work used the description logic environment CLASSIC (Borgida, et al., 1989) to model a site's security policy, static network, software configuration, and intrusion events. Similar to SCYLLARUS, we use a DL based event representation, however we differ in focus. We concentrate on a wider domain of interest than event fusion in intrusion detection, and therefore require a different model of elementary events, and event patterns which we discuss in detail below.

2.3 Event Patterns and Event Pattern Languages

It is in the area of Misuse Intrusion Detection Systems (IDS) that there has been most work done on the matching of event patterns. These IDS use either signature or rule based approaches or a mixture of the two. Rule based approaches are characterized by statements that have the form of "IF condition THEN conclusion". Signature based approaches typically operate at a higher level of abstraction than rule based approaches by using declarative languages that model different aspects of situation. Both signature and rule based techniques typically entail specifying event signatures or rules using some kind of event language.

A number of signature based alert correlation languages aim to correlate events based on abstract models of intrusion goals. The LAMBDA correlation language is a signature based approach matches signatures of event consequences with event prerequisites, generating Prolog based correlation rules (Cuppens, et al., 2002). This language uses an ad-hoc combination of XML and Prolog syntax to model both Attacks and Alerts. JIGSAW uses a similar technique for correlating pre and post conditions, focusing more on language syntax (Templeton, et al., 2000). They model pre and post conditions as "requires" and "provides" relationships of events. Ning et. al. criticize JIGSAW as overly restrictive, and weaken the requires/provides relation in Hyper-Alerts to allow correlation in absence of certain prerequisites (Ning, et al., 2002). Similarly to LAMBDA, both JIGSAW and Hyper-Alerts are translatable to rules.

CEP (Perrochon, et al., 2000), employs a rule language called RAPIDE for event pattern recognition and correlation. This language contained features to match over parameters such as causal ancestry, repetition, as well as simple property based comparison. STATL (Eckmann, et al., 2000) uses finite state machine (FSM) models to specify signatures. Doyle et. al. critique using FSM's: "Representing events as transitions through a single chain of states precludes recognizing the achievement of a set of attack preconditions that have no innate required time order.” (p. 21) (Doyle, et al., 2001) Techniques for translating FSM’s into rules are well established.

The line where rule and signature based approaches become expert systems is unclear. Two differentiators would be the use of dynamic, frame based knowledge models, and a translation stage between signature specifications and underlying rule representations. A number of approaches have applied expert systems and logic based reasoning to event correlation. The EMERALD IDS (Lindqvist, et al., 1999) uses a expert system, P-BEST to specify intrusion signatures and react to signature matches. Doyle et al. criticise P-BEST for lacking any concepts specific to event recognition (Doyle, et al., 2001). Stallard and Levitt use the JESS expert system in the forensic context to detect inconsistencies in log entries as compared to invariant properties of a system model (Stallard, et al., 2003).

Of the correlation languages reviewed, only STATL was available with source code. RAPIDE is available only as executables, and has not been updated since 1998. JIGSAW has no implementation. Hyper-Alerts, while implemented, have no available implementation, nor has LAMBDA. P-BEST is only available embedded in the EMERALD product (Lindqvist, et al., 1999), and is not easily modified, nor is it straightforward how to access the P-BEST functionality. It has very few features differentiating it from the class of languages based on the CLIPS expert system (Riley, 2004) and is very similar to JENA’s rule language.

As the focus of our research is currently on the representation and reasoning required for effective forensic investigation, and due to the observation that most of the correlation languages reviewed are translatable to rules, we chose to use JENA’s built-in rule language.

3. APPROACH

3.1 KR&R Framework

For the reasons previously outlined, and given that the research community is currently very active around the application and implementation of OWL, we have selected OWL as our knowledge representation system.

Current implementations of DL reasoners, such as FaCT (Horrocks, 1999) have implemented translators from OWL to their native syntax. Instance reasoning is limited by their support for datatypes such as those defined by the OWL related language, XML Schema (XSD) (Sperberg-McQueen, et al., 2001). For our research, support for time datatypes supporting the expression of instants (or timestamps) as date and time values is a necessity that is not satisfied by any of the current breed of DL reasoners. The CLASSIC system appears to have been at a standstill for some time, with little evidence of an active user community. As this is the case, we investigated alternate logics for reasoning with OWL.

JTP (Fikes, et al., 2003) initially appeared useful, as it has a built in time ontology and a temporal reasoner, OWL support, and is open source. However, its FOL implementation proved to perform slowly, the surrounding online community dormant, and did not support simple Knowledge Base operations such as removal of previously asserted facts. OWLJessKB (Kopena, 2003), an implementation of OWL using the JESS (Friedman-Hill, 1995) production system provides a reasoning method over OWL and has a small, but active community, however OWLJessKB does not support reasoning over time in a clean way. Further, JESS is closed source software. The JENA semantic web toolkit (McBride, 2002) has recently added both forward chaining reasoner, similar to JESS, and backward chaining reasoner similar to the tabled Prolog, XSB. The ontology API is clear and well documented, the source is distributed under a liberal license, and is supported by an active community. For these reasons we chose JENA as the KR and Reasoning (KR&R) framework to use as the foundations of our architecture.

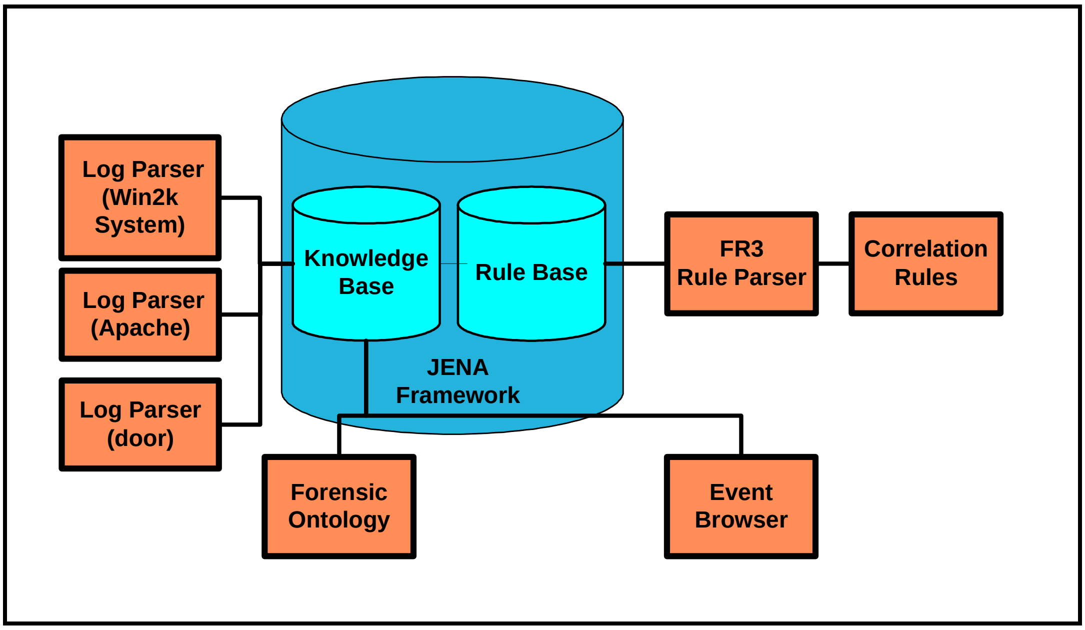

Figure 1. The FORE Architecture

3.2 Representation

We use the JENA framework as a knowledge base, by introducing instances of event related knowledge sourced from disparate event log sources, including windows security logs, apache web server logs, and UNIX syslog. These event instances are created as instances of concepts defined in a forensic ontology that we have defined. As the instances are inserted into the knowledge base, the JENA rule forwards rule engine applies correlation rules defined in the language FR3 to the knowledge.

3.2.1 The Forensic Ontology

While a number of ontologies related to security and intrusion detection were identified, (Chen, Finin, et al., 2003, Pan, et al., 2004, Undercoffer, et al., 2004) in developing our prototype system we have for simplicity and tractability developed our own ontology. Currently we are limiting our use of the features of OWL to class taxonomy definition, property definition and specification of event instance declaration. We are not using any constraints over the classes or object properties, or property hierarchies.

Our ontology is rooted in two base classes, an Entity class that represents tangible "objects" in our world, and an Event class that represents changes in state over time.

Due to a lack of standardised time primitives in the OWL language, and an absence of temporal reasoning support in JENA, we have adopted a simple model of time that assumes basic events to happen in an instant of time. Our basic temporal ordering property is thus supported by reasoning over the *startTime* owl:DatatypeProperty of the Event class. In the absence of a more comprehensive temporal model, we assume that the clocks of all time sources in our system are synchronized using a protocol such as NTP, or that times are subsequently adjusted or normalized.

Causal linkage is modelled as an owl:ObjectProperty, whose domain and range are both Event. In effect, this is a unidirectional relation where a parent event has a collection of causal ancestors. We borrow Luckham's definition of causality "If the activity signified by event A *had to happen* in order for the activity signified by event B to happen, then A caused B" (p.95) (Luckham, 2002). The causal linkage property is represented in OWL as the following:

```
  <owl:ObjectProperty rdf:ID="causality">
  <rdfs:range rdf:resource="#Event"/>
  <rdfs:domain rdf:resource="#Event"/>
</owl:ObjectProperty>
```

Semantic generalization and event composition are implemented by the creation of new events that represent the generalized concept. For example, we define a *DownloadExecutionEvent*. This class represents a composite event, representing that a user has executed content that has previously been downloaded, for example by downloading with a web browser, or as an email attachment.. This event composes a *FileReceiveEvent*, and an *ExecutionEvent*. In this case, these events are instances of the specific subclasses *WebServerDownloadEvent* (sourced from web server logs) and *Win32ExecutionEvent*, (sourced from a Windows 2000 Security log) respectively.

3.2.2 Log Parsers

In order to introduce the unstructured and heterogeneous event log data into the JENA knowledge base, we have developed a set of regular expression based parsers. These parsers use a similar syntax as ECF’s parsers to match event specifications, and create sets of OWL instances in the knowledge base. An example of an instance of a Windows 2000 login event sourced from the windows security log and converted to an event instance follows:

```
<Win32ConsoleLoginEvent rdf:ID="loginInstance1">
     <startTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
     >2002-03-04T20:30:00Z</startTime>
     <user rdf:resource="#user1" />
     <host rdf:resource="#host1" />
</Win32ConsoleLoginEvent>

<Win32DomainAccount rdf:ID="user1">
     <userName>jbloggs</userName>
     <domain>DSTO</domain>
</ Win32DomainAccount >
```

Notable points here are the usage of XML Schema (XSD) for encoding the time of the event in the representation, and the use of RDF resource references to link the event with instances of entities representing the user and the host.

The simplified model of time adopted currently avoids addressing the implications of timing irregularities, such as clock drift, cross time zone event sources, deliberate modification of time records, and lack of time synchronization. We currently provide features at the event parser layer to skew time data to address lack of precision to the year measure in syslog based event logs. This feature may be used to address timing irregularities where the irregularity is quantifiable and regular.

We currently assume that the integrity of the event sources has not been compromised.

3.3 The Correlation Rule Language

As the focus of our research is currently on the representation and reasoning required for effective forensic investigation, and due to the observation that most of the correlation languages reviewed are translatable to rules, we chose to use JENA’s built-in rule language. However, investigations indicated that the rule language was overly verbose. In order to create readable and manageable rules, we have created a rule language, FR3 to express correlation rules.

Our language is based on the language F-Logic (Kifer, et al., 1995) and the XML specific features of another F-Logic inspired language, TRIPLE (Sintek, et al., 2002). We have however adopted much simpler semantics, eschewing path expressions and reified statements (statements about statements).

3.3.1 *Namespace Support*

FR3 has specific support for XML namespaces and resource identifiers. Resource identifiers are the OWL standard of URI's (Berners-Lee, et al., 1998). Namespaces are declared as a clause of *namespace abbreviation := namespace.* as follows:

```
fore :="http://www.isrc.qut.edu.au/fore#".
```

The usage of namespaces, (along with a number of OWL features that we will not discuss here) enables integration of concepts from separate ontologies. The fore namespace declaration resolves the concepts used in FR3 rules to concepts specified in the FORE forensic ontology. Similarly, a RDF namespace declaration enables FR3 rules to reason over type information.

3.3.2 *Object Shorthand*

A shorthand form of object attribute access is expressed using a linguistic grouping called *molecules*, and is supported via the following F-Logic inspired syntax:

```
object[property -> value; property2 -> value2]
```

This is a convenient form of syntax that in the head of the rule, enables the assignment of values to the properties of an object, without repeated use of the object. The equivalent form of these clauses in JENA's rule language would be:

```
 (object property value), (object property2 value3)
```

In an object oriented paradigm the previous could be expressed as

```
object.property = value;
object.property2 = value2;
```

In the tail of the rule this form is interpreted as equality tests and variable assignment.

3.3.3 *Rules*

Rules are specified as follows:

```
antecedents -> consequences;
```

This can be read as *IF antecedents THEN consequences*, where antecedents and consequents may contain any number of molecules or procedures. Molecules appearing in the head of the rule (an alternative term for the consequences) are interpreted as new facts of knowledge to be inserted into the knowledge base. Molecules appearing in the antecedents (also known as the tail of the rule) must occur in the knowledge base for the *IF* part of the rule to be satisfied.

Rules operate at an abstract level with respect to the conceptual hierarchy, enabling rules to be written that operate on all subclasses of a higher-level class. Variables are introduced by including a question mark at the beginning of an identifier.

3.3.4 *Reasoning*

We currently use the JENA toolkit as the knowledge base, OWL parser, and reasoner. We use the RETE (Forgy, 1982) based forward chaining reasoning engine to implement our rule language. The RETE algorithm is a speed efficient and space expensive pattern-matching algorithm with a long history of use in expert systems and rule languages.

JENA's RETE engine currently does not support rules which match on all subtypes of an abstract type. For example, a rule matching events of class *LoginEvent* would not fire when a *Win32TerminalLoginEvent* was added to the knowledge base. Currently the machinery of JENA that implements the semantics of OWL type hierarchy inference relies on a hybrid implementation involving both forward (RETE) and backward chaining (similar to Prolog) reasoners. The inferred types of the *Win32TerminalLoginEvent* are currently not available as facts to the forward chaining rule engine, as they are only computed backwards as a query. In order to make this work, we modified the JENA's OWL implementation to pre-compute the type hierarchy information using the RETE engine.

3.3.5 An Example Correlation Rule

The following correlation rule is used to causally correlate apache web log entries with a user logon sessions. Standard web server access logs only provide enough detail to determine the host that downloaded content, rather than the user on the host. As this is the case, we currently correlate the download with all login sessions that exist on the host at the time of the download.

```
fore :="http://www.isrc.qut.edu.au/fore#".

?e1[rdf:type -> fore:WebFileDownloadEvent; fore:clientHost -> ?sh ; fore:startTime
-> ?t2],
?e3[rdf:type -> fore:LoginSessionEvent ; fore:host -> ?sh ; fore:startTime -> ?t1;
fore:finishTime -> ?t3],
lessThan(?t1, ?t2), lessThan(?t2, ?t3)
->
?e1[fore:causality -> ?e3];
```

To translate the rule we have:

"Take a web file download event ?e1 that came from the Host ?sh and occurred at time ?t2 and take a login session on the same Host. If the web file download event occurred during the login session, then add the web file download event to the causal ancestry (causality property) of login session event."

All of the event Classes mentioned refer to concepts defined in the document identified in the fore: namespace declaration.

3.4 Event Browser

The event browser currently provides a number of methods of interacting with the events in the knowledge base. Two views form the basis of the user interface, the event causality view, and the entity view.

The event causality view provides a display for all events matching a certain context, displaying the properties of each event in a drill down manner. It further provides means to drill down, following the causal ancestry of a sequence of events. We implement a simple query interface for finding sets of event instances based on type and property values.

The entity view presents all entities identified in the event base, along with their properties. Entities selected in this view may be used as the basis of a query of all related events. The Entity View provides an operation enabling the investigator to hypothesise that entities of equivalent conceptual value may be correlated as representing the same instance. This is discussed below in section 4.2.2.

4. FORENSIC INVESTIGATION – A CASE STUDY

This section describes the results of the application of the FORE system to a forensic scenario identified by the ECF (Chen, Clark, et al., 2003) research. In the following sub-sections we compare and contrast the two approaches from the perspective of the forensic investigative process.

4.1 An Example Scenario - Operating System Exploit

We describe here a forensic scenario identified in the ECF research. The scenario consists of the following trace of events, and provides support for the following hypothesis: 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. The various heterogeneous event logs from which each event is sourced is identified in parentheses.

1. Person P enters room R (door log)
2. P logs on to Windows 2000 workstation W (Windows 2000 Security log)
3. P downloads exploit file F from Apache web server A (Apache web server log)
4. P executes the downloaded file F on a host W (Windows 2000 Security log)
5. Workstation W is rebooted (Windows 2000 Security log)
6. Administrator logs on to the workstation W a short time later (Windows 2000 Security log)

In this scenario, a user either noticing the server rebooting, or a user being unable to log in as Administrator would most likely alert the investigator. While it is likely that the attacker would cover their tracks by deleting the event log, one still can envisage finding the log entries via forensic analysis of the disk where the event log is located.

We are currently not focusing on cross-domain correlation, so we will not address the door log in this case.

4.2 Investigation Using FORE

The FORE approach to forensic investigation supports three methods of interacting with event log based data: search, hypothetical entity correlation, and notification. We expect that for most investigations, a mixture of all three methods would be used. We discuss an example application of these methods of investigation below.

4.2.1 Automated Investigation – Notification

Most signature based approaches to Network Management and IDS enable specification of signatures which, when they match, indicate an occurrence of interest. Adopting this approach, we facilitate the specification of correlation rules that operate over events in the KB. In this case, our investigation merely involves looking for a certain set of events that are related to misuse oriented correlation rules.

We have developed a set of rules that causally correlate authentication events and login session, and common actions performed on a computer, during a session of activity. We further have defined a misuse rule that will detect the OS exploit scenario:

```
fore :="http://www.isrc.qut.edu.au/fore#".

?e1[rdf:type -> fore: DownloadExecutionEvent; fore:startTime -> ?t1 ; fore:host ->
?h ; fore:user -> ?u; fore:causality -> ?e2 ],
?e2[rdf:type -> fore:Win32RebootEvent ; fore:host -> ?h; fore:startTime -> ?t2],
?e3[rdf:type -> fore:LoginEvent; fore:startTime -> ?t3 ; fore:hasUser->
fore:AdministratorUser ; fore:host -> ?h],
during(?t2, ?t1, "http://www.w3.org/2001/XMLSchema#duration^^P10M"),
notEqual(?u, fore:AdministratorUser),
lessThan(?t1, ?t2), lessThan(?t2, ?t3),
makeTemp(?s)
->
?s[rdf:type -> fore:OSExploitEvent ; fore:causality ->  ?e1];
```

By searching for instances of the class *OSExploitEvent* the investigator may immediately and directly find an *OSExploitEvent* that has been automatically inferred. The *OSExploitEvent* is a semantic generalization of the correlation of a *DownloadExecutionEvent* followed by a *Win32RebootEvent* followed by a *LoginEvent* with account Administrator. We require that the reboot be within a 10-minute duration from the *DownloadExecutionEvent.* The sequence of events which are causally related to an *OSExploitEvent* are displayed as a graph in Figure 2. It should be noted that the *DownloadExecutionEvent* event pattern would be matched by an instance of an *ApacheWebFileDownloadEvent,* as the latter is a concrete subclass of the former (not shown in the Figure). A Win32LoginSessionEvent will similarly satisfy the LoginSessionEvent.

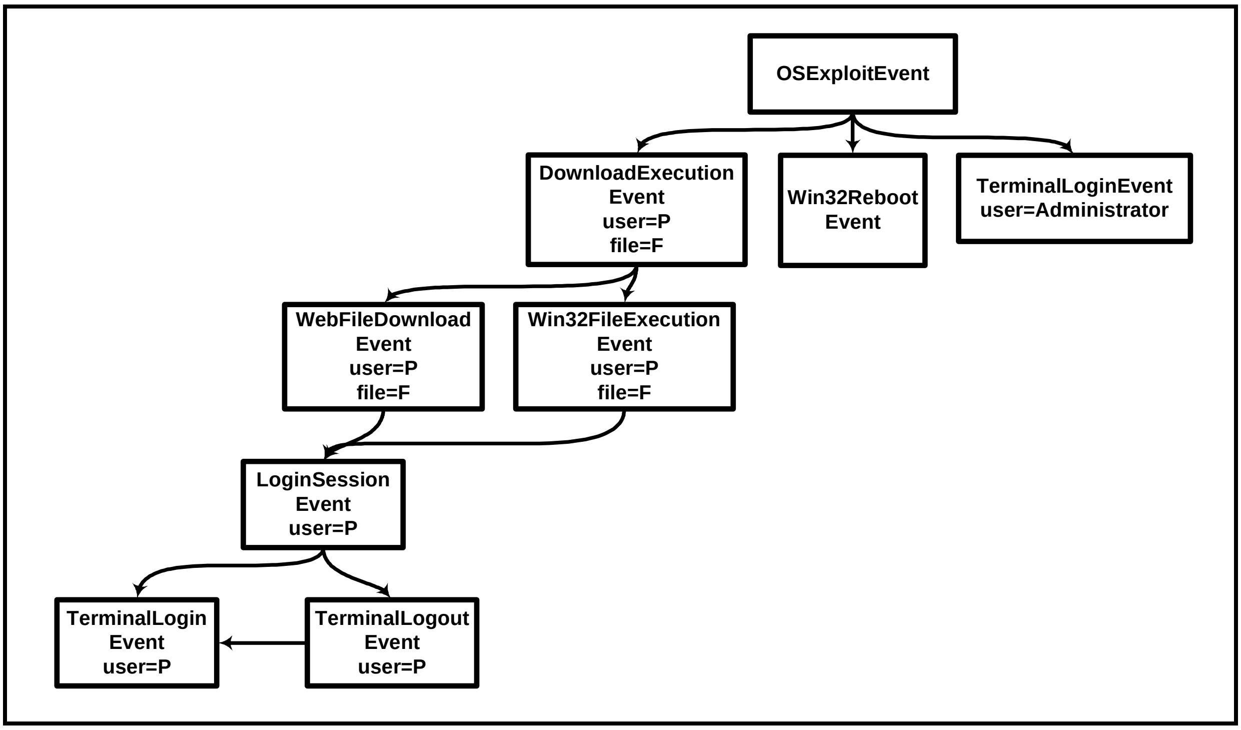
OSExploitEvent
DownloadExecution Event user=P file=F
Win32Reboot Event
TerminalLoginEvent user=Administrator
WebFileDownload Event user=P file=F
Win32FileExecution Event user=P file=F
LoginSession Event user=P
TerminalLogin Event user=P
TerminalLogout Event user=P

Figure 2. Causal Ancestry Graph of Exploit

This method of investigation will only result in success for cases where the client computer in the Apache web server logs is the same as the name of the computer in the windows security log related events. This is often not the case. We discuss below a method of addressing this shortcoming.

4.2.2 Hypothetical Entity Correlation

Heterogeneous authentication environments make the notion of identity in the security field difficult. Login names are often different to the real name of a user, and a user may have several different login names associated with different computer systems. Similarly, identifying computer hosts from log entries is complicated by the use of hostnames in some cases, and IP addresses in others.

FORE provides a novel means of investigating under the presence of this kind of uncertainty. The entity view provides an operation enabling the investigator to incorporate hypotheses, for example, to hypothesise that separate individual entities may represent the same individual. Consider the following unrelated sets of correlated events in Figure 3. The previous web server related rule in section 3.3.5 would not correlate the *ApacheWebFileDownloadEvent* with the *Win32LoginSessionEvent*, as the hosts (the client in the web log and the host in the *LoginSessionEvent*) are not the same. Similarly, in an unrelated scenario where we were interested in correlating remote shell sessions with the activities on a client computer, the *SSHPasswordAuthenticationEvent* would not correlate with the *Win32ProcessCreationEvent* that executed the SSH client, putty.exe.

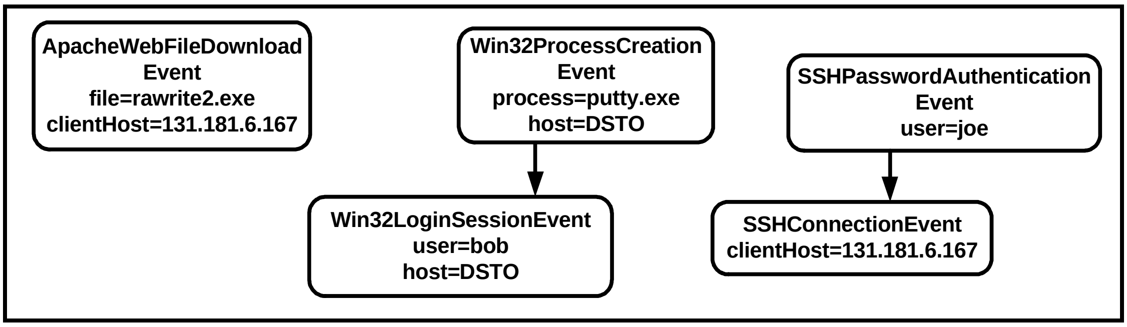
ApacheWebFileDownload Event file=rawrite2.exe clientHost=131.181.6.167
Win32ProcessCreation Event process=putty.exe host=DSTO
SSHPasswordAuthentication Event user=joe
Win32LoginSessionEvent user=bob host=DSTO
SSHConnectionEvent clientHost=131.181.6.167

Figure 3. Unrelated Event Graphs

If the investigator looks in the entities view of the event browser he will see all of the individual entities that have been identified from the event logs, including the *Host* with IP address 131.181.6.167 and another host with name “DSTO”. Through examining DNS logs, or by other means, the investigator may hypothesise that the Host “DSTO” and the Host with IP address 131.181.6.167 are in fact the same host. The investigator can select the two entities and invoke the *sameAs* operation on the two. The individual entities representing the two Hosts are now treated by the knowledge base as one single entity, a Host that has aspects of the two. The *sameAs* operation relies on the underlying semantics of the OWL individual equivalence mechanism, owl:sameAs. This language feature may be used to state that seemingly different objects or individuals are actually the same. This one individual will now suffice to fire the WebFileDownload-LoginSession rule discussed previously, and causally correlate the *ApacheWebFileDownloadEvent* to the *Win32LoginSessionEvent*, as shown in Figure 4. Similar rules may now correlate the connection initiated by the “putty” SSH client to another UNIX host.

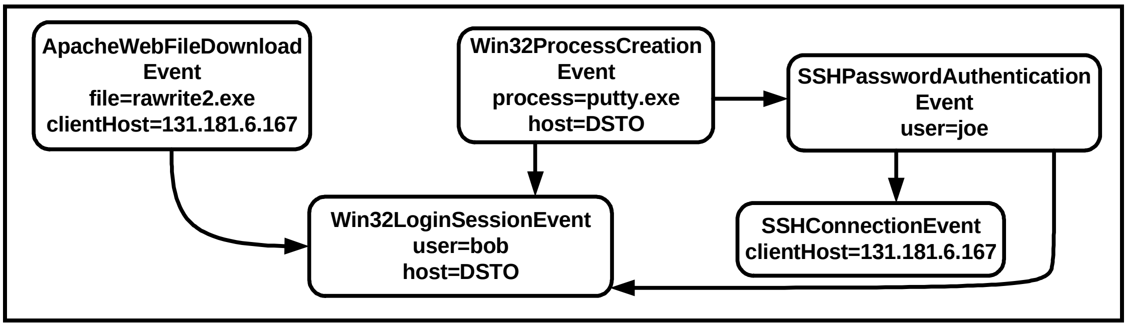
ApacheWebFileDownload Event file=rawrite2.exe clientHost=131.181.6.167
Win32ProcessCreation Event process=putty.exe host=DSTO
SSHPasswordAuthentication Event user=joe
Win32LoginSessionEvent user=bob host=DSTO
SSHConnectionEvent clientHost=131.181.6.167

Figure 4. Correlated Event Graphs from Entity Correlation

With these causal links correlated, the rules for the *OSExploitEvent* will now be satisfied, driving the creation of an *OSExploitEvent* and its subsequent display in the event view of the user interface.

4.2.3 Search Oriented Investigation

In order to explore arbitrary hypothesis, or in absence of notification outcomes, one may follow an interactive search methodology. The investigator uses the query interface of the FORE event browser to find all instances of a *LoginEvent* with the user property equal to the Administrator user. With our current knowledge base, this will return a *Win32LoginEvent* (which is a subclass of *LoginEvent*), corresponding to the login on machine “DSTO”. The investigator also now knows that machine “DSTO” runs a Windows Operating System, by virtue of the specific nature of the *Win32LoginEvent*.

The investigator at this point may be interested in what other users were doing on the computer in the time leading to this event. By querying for all instances of the *LoginSessionEvent* class prior to the administrator login event on the machine “DSTO”, the investigator will find that the user DSTO\bob was logged into the machine previously. The *LoginSessionEvent* is an abstract event we use to represent a user’s login session on a machine.

Examination of the event’s causal ancestry information will now accelerate the process. In this case, we would find many events of the user DSTO\bob, causally correlated together by rules relating web file downloads to login sessions, logins to logouts, and so on. Figure 2 includes the causal ancestry graph for bob’s activity.

By navigating through the causal ancestry graph the investigator may become suspicious of the *DownloadExecutionEvent*. This event generalises the events of the user downloading a file, (in this case sourced from an Apache log) and the execution of the file. Examining the *processName* property of the *DownloadExecutionEvent* reveals the file ‘rawrite2.exe’. The investigator knows that this file is a tool for copying bootable floppy images to a floppy disk. Exploring the causal graph further reveals that the user has downloaded a file, ‘bd030426.bin’ which is a bootable image file that in this case contains a utility to wipe the Windows administrator password, resetting it to a known value.

4.3 The ECF Investigative Process

The ECF tool provides a Dynamic Query interface that may be used to query the event database. Investigation with the ECF tool occurs largely in a work cycle revolving around the following tasks; formulate, query, and evaluate. The formulation task involves coming up with a potentially interesting event. For example, the initial investigation tactic in this scenario would likely be an investigation of what times the administrator account was logged in.

Executing this query against a database leads to the identification of a single login of the administrator account on the machine in question, “DSTO”. Evaluation of this then leads to the investigator being interested in events that occurred around the time of this login on the machine, “DSTO”. Querying for all events that occurred at the “DSTO” machine during a 20 minute window around the administrator login further reveals that user DSTO\bob was logged on approximately 7 ½ minutes prior to the Administrator account being used. The investigator would then presumably be interested in what activity occurred during the time period of DSTO\bob’s login session.

Querying for all activity on machine “DSTO” in that time period would lead to noticing a reboot event in this period. By consulting DNS logs or through other inference, the investigator may hypothesise that the machine “DSTO” had the IP address 131.181.6.167 during the time period. Widening the search to all activity involving both names of the machine, “DSTO” and 131.181.6.167, in that period, would lead to web server logs associated with the machine. Examining these further could lead the investigator to identify certain suspicious looking files being downloaded by DSTO\bob.

In all of these activities performed, a common point is that it takes the formulation of a search strategy, and a corresponding expression as a query to find the next goal in the investigation. This requires the investigator to drive the process, using the results of successful searches to infer correlation between events.

While further developments in the ECF project have led to new approaches to automate the SQL query approach, the extensibility of that approach is still subject to the limitations and inflexibility imposed by use of a database schema to represent the knowledge base, the facts and their attributes.

4.4 Comparison of FORE and ECF

Both the ECF and FORE approaches support querying event data in the exploration of a hypothesis. FORE differs from ECF in that the underlying event base, in the case of FORE contains many linkages that have been inferred by event correlation rules at the time events are loaded into the system. These causal linkages enable an investigator to explore simple relationships without manual inference. ECF however, contains none of these causal linkages. Following causal linkage in ECF involves the operator inferring the linkages, and expressing the conditions required in SQL queries. For example, for the scenario explored previously, the investigator would have to write a series of SQL queries to successively narrow the set of events in question. Investigation using ECF thus requires considerably more human inference than FORE.

FORE adds investigation features not found in ECF. In addition to providing a general purpose KR framework and a complementary event correlation language, our approach introduces the ability of hypothetically unifying entities of equivalent identity, further enhancing the effectiveness of existing rule based correlation approaches. We enable the representation of generalized events, enabling investigators to reason with generalized concepts, at higher levels of abstraction. FORE aims for high semantic consistency with little information loss, while ECF values information normalization.

The FORE prototype holds the promise of collaborative development of correlation rules that correlate events across and within domains, reducing the amount of manual inference and query tasks, and assisting in interactive investigation. At a higher level, correlation rules can automatically correlate whole forensic scenarios without interactive investigation.

5. CONCLUSION

The contribution of the work described in this paper is the automated detection of a computer forensic scenario, based upon facts automatically derived from digital event logs. We have demonstrated proof of concept of our framework and of a prototype implementation of the framework by applying the prototype to a test case scenario described in Chen et al (2003). Our prototype implementation supports rapid integration of event and environment based knowledge, and automated inference over this knowledge, while supporting the application of standard rule based event correlation techniques.

We have demonstrated that OWL is an effective means for representing event log based knowledge and event correlation specific rules, involving temporal, spatial and virtual domains. The results presented in section 4 above indicate that automated inference applied to correlation of events in this representation provides a sound basis for minimising the amount of manual inference the forensic investigator must perform.

We have introduced a novel mechanism for allowing an investigator to assert hypotheses regarding identity, which are in turn temporarily incorporated into the knowledge base. This technique increases the effectiveness of rule based correlation approaches.

Our work has demonstrated the flexibility of the approach in a simple cross-domain context. We intend now to apply the approach to multiple domains and to evaluate it against scenarios that cross those domains. We are currently undertaking further work towards the integration of standardised component ontologies for time, place, space and security related concepts. We have identified a number of potential candidates for integration. The simplified model of time adopted currently avoids addressing the implications of timing irregularities, such as clock drift, cross time zone event sources, deliberate modification of time records, and lack of time synchronization. Further work will investigate the incorporation of models of time that address these issues.

REFERENCES

Borgida, A., Brachman, R. J., McGuinness, D. L. and Resnick, L. A. (1989), CLASSIC: A Structural Data Model for Objects, ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May-June, 1989

Chen, H., Finin, T. and Joshi, A. (2003), An Ontology for Context-Aware Pervasive Computing Environments, Adjunct Proceedings of the 6th International Conference on Ubiquitous Computing, Seattle, Washington, October 12-15, 2003

Chen, K., Clark, A., DeVel, O. and Mohay, G. (2003), ECF - Event Correlation for Forensics, 1st Australian Computer, Network & Information Forensics Conference, Perth, Western Australia, November 25, 2003

Cuppens, F. and Miege, A. (2002), Alert Correlation in a Cooperative Intrusion Detection Framework, IEEE Symposium on Security and Privacy, Berkeley, California, May 12-15, 2002

Doyle, J., Kohane, I., Long, W., Shrobe, H. and Szolovits, P. (2001), Event Recognition Beyond Signature and Anomaly, IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, New York, June 5-6, 2001

Fikes, R., Jenkins, J. and Frank, G. (2003), JTP: A System Architecture and Component Library for Hybrid Reasoning, Proceedings of the Seventh World Multiconference on Systemics, Cybernetics, and Informatics, Orlando, Florida, July 27 - 30, 2003

Forgy, C. (1982), Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem, Artificial Intelligence, 19 1, 17-37

Goldman, R., Heimerdinger, W., Harp, S., Geib, C., Thomas, V. and Carter, R. (2001), Information Modeling for Intrusion Report Aggregation, DARPA Information Survivability Conference and Exposition II, Anaheim, CA, June 12–14, 2001

Grosof, B. N., Horrocks, I., Volz, R. and Decker, S. (2003), Description Logic Programs: Combining Logic Programs with Description Logic, Twelfth International World Wide Web Conference, Budapest, Hungary, May 20-24, 2003

Kifer, M., Lausen, G. and Wu, J. (1995), Logical Foundations for Object-Oriented and Frame-Based Languages, Journal of the Association of Computing Machinery, 42 3, 741-843

Lassila, O. and McGuinness, D. (2001), The Role of Frame-Based Representation on the Semantic Web, Linköping Electronic Articles in Computer and Information Science, 6 5,

Lenat, D. B. (1995), CYC: a large-scale investment in knowledge infrastructure, Communications of the ACM, 38 11, 33-38

Lindqvist, U. and Porras, P. A. (1999), Detecting computer and network misuse through the production-based expert system toolset (P-BEST), IEEE Symposium on Security and Privacy, Berkeley, California, May 9-12, 1999

Luckham, D. (2002), The Power of Events, Pearson Education, Indianapolis, Indiana

McBride, B. (2002), Jena: a semantic web toolkit, IEEE Internet Computing, 6 6, 55-59

Ning, P., Cui, Y. and Reeves, D. (2002), Constructing attack scenarios through correlation of intrusion alerts, 9th ACM conference on Computer and Communications Security, Washington, DC, November 18-22, 2002

Pan, F. and Hobbs, J. R. (2004), Time in OWL-S, 2004 AAAI Spring Symposium Series - Semantic Web Services, Stanford University, March 2004

Perrochon, L., Jang, E., Kasriel, S. and Luckham, D. C. (2000), Enlisting Event Patterns for Cyber Battlefield Awareness, DARPA Information Survivability Conference & Exposition, Hilton Head, South Carolina., January 25-27, 2000

Sintek, M. and Decker, S. (2002), TRIPLE---A Query, Inference, and Transformation Language for the Semantic Web, International Semantic Web Conference (ISWC), Sardinia, June 2002

Stallard, T. and Levitt, K. (2003), Automated Analysis for Forensic Science: Semantic Integrity Checking, 19th Annual Computer Security Applications Conference, Las Vegas, Nevada, December 08 - 12, 2003

Templeton, S. J. and Levitt, K. (2000), A Requires/Provides Model for Computer Attacks, New Security Paradigms Workshop, Ballycotton, County Cork, Ireland, September 19-21, 2000

Undercoffer, J., Joshi, A., Finin, T. and Pinkston, J. (2004), A Target-Centric Ontology for Intrusion Detection, 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003