RAID 2007, LNCS 4637, 146–166
http://www.springer.com/978-3-540-74319-4

ELICIT: A System for Detecting Insiders Who Violate Need-to-know

Marcus A. Maloof[1] and Gregory D. Stephens[2]

[1] Department of Computer Science,
Georgetown University, Washington, DC 20057, USA,
maloof@cs.georgetown.edu
[2] Center for Integrated Intelligence Systems,
The MITRE Corporation, McLean, VA 22102, USA,
gstephens@mitre.org

Abstract. Malicious insiders do great harm and avoid detection by using their legitimate privileges to steal information that is often outside the scope of their duties. Based on information from public cases, consultation with domain experts, and analysis of a massive collection of information-use events and contextual information, we developed an approach for detecting insiders who operate outside the scope of their duties and thus violate *need-to-know*. Based on the approach, we built and evaluated ELICIT, a system designed to help analysts investigate insider threats. Empirical results suggest that, for a specified decision threshold of .5, ELICIT achieves a detection rate of .84 and a false-positive rate of .015, flagging per day only 23 users of 1,548 for further scrutiny. It achieved an area under an ROC curve of .92.

Keywords: misuse, insider threat, anomaly detection.

1 Introduction

Recently, the FBI arrested analyst Leandro Aragoncillo after he allegedly "conducted extensive keyword searches relating to the Philippines" and "printed or downloaded 101 classified documents", also relating to the Philippines [1]. Although Aragoncillo was an intelligence analyst, information about the Philippines was "outside the scope of his assignments" [1].

We are interested in detecting this type of malicious insider, but the problem of detecting *insiders* is much more complex and multi-faceted. For instance, malicious insiders are often disgruntled [2], so better working environments could lead to a reduced threat. Better processes for screening employees could also reduce the threat. On corporate intranets, one may be able to deploy methods traditionally used against external intruders to counter an insider who is, say, attempting to gain unauthorized access to a server. In contrast, we are interested in detecting malicious insiders who operate within their privileges, but who engage in activity that is outside the scope of their legitimate assignments and thus violate *need-to-know*.

In this paper, we describe our efforts to develop and evaluate methods of detecting insiders who violate need-to-know. Based on analysis, research, and

consultation with domain experts, we designed an approach that consists of four main steps. First, *decoders* process network traffic from protocols associated with the use of information into higher-level *information-use* events. Second, a suite of detectors, supplanted with contextual information about users, groups, and organizations, examines these events and issues alerts. Third, a Bayesian network uses these alerts as evidence and computes threat scores. Fourth, an interface presents events, alerts, and threat scores of users to security analysts. Based on this approach, we implemented a system named ELICIT, which stands for "Exploit Latent Information to Counter Insider Threats."

To support ELICIT's development and evaluation, we derived a data set from 284 days of network traffic collected from an operational corporate intranet. Over a period of 13 months, we processed 16 terabytes of raw packets into more than 91 million information-use events for more than 3,900 users. We then examined these events to characterize the searching, browsing, downloading, and printing activity of individuals, groups of individuals, and the organization as a whole. We built 76 detectors and a Bayesian network that, together, produce an overall threat score for each user in the organization.

To evaluate our approach and ELICIT, a red team developed scenarios based on information from real, publicly-available cases. They translated the scenarios to the target environment and executed them during normal network operation. A trusted agent[3] used scripts to insert events of the scenarios into our collection of events. We then ran ELICIT, as would an analyst, in an effort to identify the users corresponding to the scenarios.

Over a period of two months, using a specified decision threshold of .5, ELICIT detected the insiders on 16 of the 19 days they were acting maliciously, corresponding to a detection rate of .84. During this same period, ELICIT scored an average of 1,548 users per day, with an average of only 23 users scoring high enough to warrant further scrutiny, meaning that ELICIT's average false-positive rate is .015. By varying the decision threshold, we produced an ROC curve, the area under which was .92.

2 Problem Statement

There are many detection tasks important for securing systems, their software, and their information, such as detecting intruders [3, 4], and anomalous command [5] and system-call [6] sequences. We focus on the task of detecting misuse, defined as legitimate users abusing their privileges.

Detecting misuse is a complex, multi-faceted problem, and malicious insiders, or simply *insiders*, may engage in a variety of activities. Insiders could use knowledge of their organization's intranet and behave in a manner similar to an intruder. Such activities could include scanning ports, executing buffer overflows, and cracking password files, and one can detect these activities with methods of

[3] Herein, all uses of the term *trusted agent* refer to the person serving as the intermediary between the red team and the research team.

intrusion detection. Insiders could also masquerade as another user by compromising his or her account. However, in our work, we are interested in detecting malicious insiders who do not engage in these activities.

In a computing system, access-control mechanisms yield a set of illegal and legal actions for each user. Such actions include viewing certain documents, and so, there will be documents that a user can and cannot view. Unfortunately, for large, dynamic organizations, it is difficult to design and maintain effective access control. Consequently, given the set of legal actions for a user, there is a set of such actions that is suspect, especially given contextual information about the user. In our work, we are interested in detecting insiders who browse, search, download, and print documents and files to which they have access, but that are inappropriate or uncharacteristic for them based on contextual information, such as their identity, past activity, and organizational context.

Our conception of detecting insiders is quite different than detecting external intruders. One rarely, if ever, has the contextual information for such intruders that one has for insiders. Our aim is to leverage this context for detection. It is also different than detecting internal intruders, since insiders who violate need-to-know do not need to break rules to achieve their goals. All detection systems must analyze events at correct levels of abstraction, and the insiders of interest to us usually gather and exfiltrate documents. Consequently, rather than detecting malicious activity based on connections, packets, or system-call sequences, we chose to detect insiders based on *information-use events*, which we describe further in the next section.

3 Data Collection

We derived the data set for our study from an operational corporate intranet. In the following subsections, we describe how we processed network traffic into *information-use events*, collected contextual information about users and the information they accessed, and developed scenarios for the purpose of evaluation.

3.1 Transforming Network Traffic into Information-Use Events

To collect network events, we placed passive sensors between clients and servers within a large corporate intranet for 284 days.[4] The sensors collected packets from network protocols correlated with the legitimate use of information, a critical aspect of our work. In total, we captured approximately 16 terabytes of data,

[4] We experienced three outages. Two months into the period, an administrative error resulted in an outage for two days. Three months later, an unanticipated network reconfiguration caused a near-complete loss of data for five days. Four months into collection, we discovered and corrected an error in the software that captured packets. Subsequent analysis indicating that the flawed version failed to capture about 9% of the packets, with the majority of the loss occurring during traffic bursts. Nonetheless, data from this period was helpful for analysis and development. Crucially, the red team did not execute the scenarios until after we resolved these problems.

Table 1. Information Stored for Events

	Actions							
Field	List	Delete	Read	Write	Move	Print	Query	Send
Protocol	X	X	X	X	X	X	X	X
File Name/Path	X	X	X	X	X	X	X	X
Start/Stop Time	X	X	X	X	X	X	X	X
Client/Server IP	X	X	X	X	X	X	X	X
User Name	X	X	X	X	X	X	X	X
Bytes			X	X				
Original File Name					X			
Printer						X		
Pages						X		
Search Phrase							X	
E-mail Headers								X

corresponding to 27 billion packets. In the collection, 61% of packets were from the SMB protocol, 35% were from HTTP, 3%, from SMTP, and 1%, from FTP.

We developed a series of *protocol decoders* to transform the packets into information-use events. These decoders also tracked authenticated users across sessions and captured clues about their identity, which aided in subsequent attribution. Off-line, the trusted agent used Ethereal [7] to filter and dissect packets, and then applied our decoders to produce information-use events. Over a period of 13 months, the decoders processed more than 3.7 billion packets, producing more than 91 million events, which we stored in a relational database.

Referring to Table 1, each event consisted of an action and variable number of fields. Decoders extracted eight actions. In our collection, 35.8% of the actions were lists of files or directories, 42.1% were reads, 12.9% were writes, 4.6% were deletes, 2.9% were sends of e-mail, 1.1% were search-engine queries, 0.4% were prints of documents, and 0.3% were moves of files or directories. The decoders also extracted fields such as the start and end time of the action, the protocol involved, and other pertinent information. Table 2 contains an example of a print event in which user `p0314508p` printed a document named `Liz's form fax.doc` to the printer `\\spool2\335-HP`. Values for all other fields are null.

With the exception of send, we selected these actions and fields based on analysis of past insider cases and hypotheses about which would be useful for detecting violations of need-to-know. Then, during decoder development, we implemented routines to capture information from e-mail because we realized that it would be useful for constructing social networks.

We did not collect data directly on clients, so our approach is network-based, rather than host-based. In the environment we monitored, it would have been impractical—though desirable—to instrument every machine with the software necessary to collect events. We also did not collect packets inbound from or outbound to the Internet due to concerns about privacy. If our approach were used in an organization where such technical and privacy issues could be resolved, ELICIT's design is flexible enough to accommodate these new sources of information.

Table 2. Example of the Relevant Fields of a Print Event

Field	Value
Action	`print`
Protocol	`smb`
File Name	`Liz's form fax.doc`
Start Time	`2005-02-03 10:32:16.993`
Stop Time	`2005-02-03 10:32:17.003`
Client IP	`ddd.ddd.ddd.13`
Server IP	`ddd.ddd.ddd.239`
User Name	`p0314508p`
Bytes	`2672`
Printer	`\\spool2\335-HP`
Pages	`1`

3.2 Collection of Contextual Information

In addition to events, we developed sensors that periodically collected contextual information about the users and the information they accessed and manipulated. This included information from an employee directory, such as name, office location, job description, seniority, and projects. We also copied the contents of files in users' public directories on a shared file system, and we extracted information from the directory structure itself, the branches of which often corresponded to users, projects, and the organization's business units. We stored this information in a database, and Table 3 shows an example.

With this contextual information, we were able to build simple social networks with e-mail traffic, use a person's job description in the analysis of his or her search-engine queries, and determine if someone printed to a printer close to his or her office. It also let us compare a user's behavior to that of peers, such as those with the same job description and those working on the same floor, in the same department, and on joint projects.

These comparisons illustrate a critical aspect of our work. We are not simply examining network events between client and server IP addresses. We are monitoring how users access and manipulate information, and we are coupling this activity with contextual information about the users and the information itself.

3.3 Data Anonymization

To protect the privacy of the users, the trusted agent removed, anonymized, or abstracted any identifying information before releasing it to us, the research team. The trusted agent removed hire dates and phone numbers, replaced names and user IDs with pseudonyms, and abstracted office numbers to their floor.

An important concern is whether the process of anonymization introduced artifacts that may have affected detection. For this study, phone numbers and hire dates were not important for detection, so their removal was not problematic. Name and user ID are not relevant for detection, but are critical as labels connecting events and detector outputs. However, pseudonyms serve this purpose equally well.

Table 3. Example of Contextual Information for User `p0314508p`

Field	Value
User Name	`p0314508p`
E-mail Address	`p0314508p`
User ID	`p0314508p`
Home Directory	`s:\p0314508p\public`
Department	`Accounts Payable`
Division	`Purchasing`
Office Location	`7th Floor`
Job Title	`General Accounting Specialist`
Job Category	`General Accounting`
Job Level	`3`
Project 1	`Accounts Payable`
Project 2	`Travel Accounting`

Abstracting office location to its floor did make it difficult or impossible to conduct certain analyses, which may have negatively affected detection. For example, we could not identify relationships between people who shared offices or who worked in adjacent offices. Since ours is a research effort, we had to accept that there was certain information we simply could not use or collect. Nonetheless, even without this information, our results are promising.

3.4 Event Attribution

To use an individual's context, such as their job description or social network, we had to attribute each event to a user. Unfortunately, not all sessions, and thus not all events, had information about the user who produced them. For example, unprotected Windows file shares and web sites requiring no authentication generated events without identifying information. In the database, such events have null values for their user IDs.

Our collection contained three types of events: unattributed events, indirectly-attributed events, and events directly attributed to a user because of an observed successful authentication. For example, most SMB sessions begin with an authentication, and we can then attribute subsequent events of the session to the authenticated user. Indirectly attributed events are those with some type of user context, such as the sender's address in an e-mail. Of the more than 91 million events, 14.7% were directly attributed, 2.3% were indirectly attributed, and 83% were initially unattributed.

With network engineers familiar with the network environment, we devised two off-line methods to label unattributed events. Both used events occurring before and after an unattributed event. The first was a nearest-neighbor method that attributes an unattributed event to the user of the closest attributed event, as measured by time. The second method uses a kernel function to give more weight to the attributed events closer to the unattributed event. To reflect the uncertainty of attribution sources (e.g., due to configuration or human errors), network engineers determined measures of confidence for each, assigning print

events a weight of .999, send events a weight of .99, and FTP events a weight of .9. Directly attributed events had a weight of 1 and unattributed events had a weight of 0.

An *attribution event* e_i is then a 3-tuple $\langle u_i, w_i, t_i \rangle$, where u_i is the ID of the attributed user, w_i is the weight, and t_i is the time of occurrence. For a given client IP address, there is a sequence of attribution events with and without attribution. Let S be a sequence of n events ordered by t_i, and let $S^{(u)}$ be the sequence of events from S attributed to user u:

$$S^{(u)} = \{\langle u_i, w_i, t_i \rangle : \langle u_i, w_i, t_i \rangle \in S \wedge u_i = u\} \ .$$

If e_i is an unattributed event (i.e., $u_i = \emptyset$) occurring in the middle of sequence S (i.e., $i = n/2$), then we attribute e_i to the user in the sequence whose actions have the maximum weight. That is, given the kernel function

$$K(e_i, e_j) = w_j e^{-\gamma(t_i - t_j)^2} \ ,$$

where γ determines the width of the kernel,

$$u_i = \underset{u \in S}{\operatorname{argmax}} \left(\textstyle\sum_{e_j \in S^{(u)}} K(e_i, e_j) \right) \ .$$

For each unattributed event in a sequence, we applied both methods. With the kernel method, we set $\gamma = 5 \times 10^{-5}$ and used overlapping sequences of events that were 16 hours in length. If the weight calculated for an unattributed event was less than 1×10^{-5}, then the event remained unattributed. If both methods returned the same user ID, then we set the ID of the unattributed event to the inferred ID. If the methods did not agree, then the event remained unattributed.

To evaluate this procedure, we compared performance to our two network engineers. We randomly selected 100 unattributed events and applied our attribution procedure. We provided the same events to both experts, who independently attributed the events. They then resolved any differences to produce a single set of attributed events. Our procedure agreed with the experts on 99 of the 100 events. The disagreement involved an ambiguous event from a multi-user machine that belonged either to the end of one user's session or the beginning of another's.

Applying this procedure to all of the events, we were able to infer attribution for 65% of the previously unattributed events. About 28.6% of the events in the collection remained unattributable, but 82% of these originated from 25 hosts running automated processes. Those remaining were ambiguous events from clients with multiple, concurrent users.

3.5 The Need for Meta-Events

Early on, we noticed that users initiating certain simple actions produced an inordinate number of information-use events. Executing a find command on a large, shared file system is one example. We also found that software automatically updating files on clients or servers often accounted for most of a user's

Table 4. Scenarios and Their Descriptions

Scenario	Description
S1	Employee who gathers technical information about aviation and aeronautics, topics that are outside the scope of the employee's duties.
S2	System administrator who obtains financial information, such as internal reports, disclosure statements, labor rates, and the like.
S3	Disgruntled employee who is to leave the company and gathers a large amount of widely varying documents containing sensitive, proprietary information.
S4	Employee who collects a large volume of information about *knowledge management*, which is unrelated to the employee's duties.
S5	Employee who gathers software relating to aviation from a proprietary, internal repository.

"browsing" activity (e.g., software updating a public calendar from a personal calendar). While the information-use events of these sequences are interesting themselves, we did not want the number of events in a sequence to skew certain types of analyses.

As a result, we grouped certain sequences of directory and file events into *meta-events*. We segmented a user's events when there was a change in protocol, a change in the server's IP address, or when the separation between two events was greater than ten seconds. Over each segment, we computed the number of events in the segment, the rate at which the events occurred, and the percentage of list events in the segment. If a segment was longer than 20 events and the frequency of events was greater than two per second, then we labeled the segment as a meta-event. If the percentage of list events within the segment was greater than 90%, then we further labeled the segment as being the result of a find command. In the database, we used a unique identifier to label events of a meta-event. An additional field indicated whether the meta-event was the result of a list or find command. Although we determined these thresholds empirically, we found that this heuristic method worked well for our events.

3.6 Scenario Development and Execution

The data set described so far consists of activity for 3,983 users. It has proven invaluable for analysis and the development of detectors. However, it contains no known malicious activity, which limits our ability to evaluate our approach.

In response, a red team constructed fifteen scenarios inspired by public cases that involved the gathering of illicit information by individuals such as Aldrich Ames, Ryan Anderson, and Brian Regan. Domain experts reviewed the scenarios and adapted them to the network we monitored. Once approved, the red team executed the five scenarios listed in Table 4 during normal network operation. Three of the five scenarios were executed by two different members of the red team (i.e., S2, S3, and S4) in an effort to assess the role that individual personality might play in scenario execution and detection. This resulted in a total of eight scenario executions. (We did not execute the remaining scenarios because

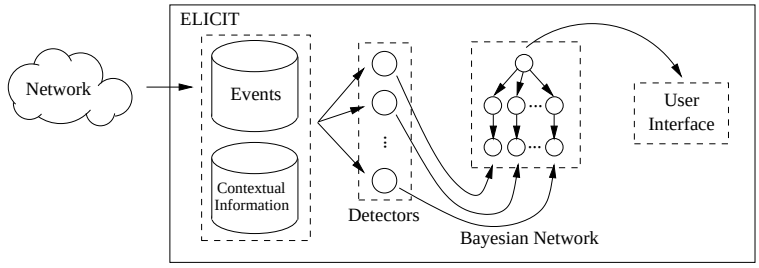

Fig. 1. ELICIT's architecture.

members of the red team were assigned to other projects; we plan to use the scenarios in future work.)

The red team used their legitimate accounts to execute scenarios during normal network operation, which let the trusted agent process the "benign" and scenario data together. Using the red team's detailed logs of their activity and demarcation events sent via e-mail, the trusted agent isolated and then removed the scenario events from the benign collection of information-use events. This let the trusted agent insert and remove individual scenarios at will.

The members of the red team were knowledgeable about insider activity and investigations. They were given the scenario and its translation, and instructed to achieve an objective (i.e., steal information) in a manner consistent with the scenario and its insider. These instructions identified specific topics, documents, and systems (e.g., financial), but their actions were not tightly scripted. Members were not told how to achieve their objective or the time over which an attack must occur. To make inserting and removing the scenario events possible, they were told not to intermix benign and malicious activity between demarcation events. While not all insider attacks follow this profile, many do because insiders often take advantage of windows of opportunity.

The research team and the red team worked in isolation with the trusted agent mediating interactions. The teams did not share domain experts, and the research team had no insight into the development, execution, collection, and insertion of scenarios and their events until *after the completion* of ELICIT's development and evaluation. Although the teams worked in isolation, in retrospect, they independently profiled some of the same insiders, such as Regan, Ames, and Hanssen. However, the red team also profiled insiders that the research team did not, such as Ryan Anderson and Ana Montes. The research team did not know how the red team would translate the scenarios (e.g., that aviation would be a topic of interest).

4 The ELICIT System

ELICIT is a research prototype designed to help analysts investigate malicious insiders. As shown in Fig. 1, it consists of four main components: a database of events and contextual information, a set of detectors, a Bayesian network, and a user interface.

As described previously, we processed packet data from the network and stored the resulting events in a relational database system. Based on our analysis of these events, consultation with experts, and public information about past cases, we designed and built detectors that, over specified periods of time, examine events in the database and return a set of alerts. A Bayesian inference network uses the alerts as evidence and computes, for each user, an overall threat score. Finally, ELICIT presents the users and their scores to an analyst through the user interface.

4.1 Detectors for Anomalous Activity

To date, we have developed 76 detectors that examine events for volumetric anomalies, suspicious behavior, and evasive behavior. We define each detector along three dimensions: the activity's type, its characteristics, and its context. The type of activity can be browsing, searching, downloading, and printing. Each detector examines characteristics of the activity, such as when the activity occurred, where it occurred, and how (or to what extent) it occurred. Finally, each detector evaluates activity in context with past activity, with the activity of organizational or professional peers, or with the activity of the peers in some social network.

Each detector works by taking as arguments a time period and a set of parameters, by examining each person's activity during the time period and relevant contextual information, and by issuing an alert, provided that the user's activity meets the detector's criteria for reporting. Some detectors use only the user's events that occurred during the specified period of time, while others analyze events of other types, of other users, or from other periods of time. Some detectors alert when users engage in specific activities, such as conducting searches using inappropriate terms. Others alert when some aspect of user activity is excessive or anomalous, which means that some measure of that activity falls into a rejection region.

We based each detector on a hypothesis about the activities in which malicious insiders might engage. We formed and supported each hypothesis with analysis of the events in our data collection, with advice from domain experts, with information from public cases, or with some combination thereof. As one might expect, we could not always support a hypothesis because one or more of the other sources refuted it. For example, if we found evidence of suspicious activity, but an expert advised that it was not indicative of malicious insiders, then a detector for that activity would be of little use, at least for the environment we monitored. It is important to note that detectors suitable for one environment may not be suitable for another.

Since we had no traces of real insider attacks and no models of insider behavior for the network we monitored, we consulted with three domain experts. For several years, they have performed technical analysis of active insider cases involving the theft or misuse of information. They were familiar with the network we monitored and its users. They advised us on the activities in which insiders might engage and helped determine the parameters of detectors.

To implement detectors, we used a variety of methods, including hand-coded rules, and parametric and nonparametric density estimation. We also exploited social networks. To set their parameters, we described and presented to our experts the observed activity in both textual and graphical form using events from the database and histograms of the activity of individual users and groups of users (e.g., with the same job title). The experts came to a consensus about what they considered excessive or anomalous, and we used this information to set the parameters and thresholds of the detectors. We make no claims that these are optimal settings, but based on observation and expert feedback, they seem to be reasonable estimates. Unfortunately, due to space restrictions, we cannot describe all 76 detectors, so we describe a representative few.

Sensitive Search Terms. Insiders use search engines to find, evaluate, and collect information. Most organizations can define a set of sensitive terms that appear in documents, but that would be inappropriate for use as query terms. The term *proprietary* is an example. With the help of domain experts, we constructed a list of thirteen such search terms, and if someone's query contains one of the terms on this list, then the detector issues an alert.

Printing to a Non-Local Printer. It is important to note that in the organizations of interest to us, printing a document is often the only way of removing information. Computers are on networks isolated from the Internet and have their USB ports and external storage systems disabled. Evidence from publicly-available documents suggests that insiders collect and print documents on topics outside the scope of their duties. Presumably, the insider's co-workers would recognize if the topic of a printed document were inappropriate. Consequently, to avoid suspicion, an insider may print inappropriate documents to a distant or non-local printer.

In the environment we monitored, an overwhelming majority of users printed to their local printer, as shown in Fig. 2. The graphs show the number of documents printed plotted against the distance to the printer when the user prints from his or her office. The distance is the number of floors from an individual's office to the printer.

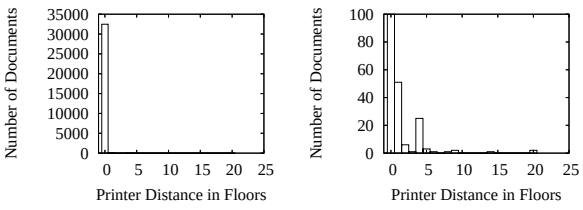

Fig. 2. Number of documents printed plotted against the distance of the printer from the user's office. (a) The fully scaled graph. (b) A version scaled to emphasize the bars right of the mode.

As shown in Fig. 2a, over a six-week period, users printed 34,279 documents, with 32,475 of these documents sent to printers on the same floor as their office. In Fig. 2b, which contains a scaled version to emphasize the bars to the right of the mode, notice the small spike at a distance of four. Five users, all from the same department, were responsible for 22 of these 25 print jobs. We are unsure why these users would print from their offices to a printer four floors above. They may have been printing to a printer in a lab.

Writing a detector for non-local prints was quite easy. In the database, we stored each user's office location, each printer's location, and for each document printed, the user who issued the print command, the location where the user issued the print command, and the printer to which the user sent the document. The detector alerts if the user prints from his or her office to a printer on a different floor.

Anomalous Browsing Activity. To take into account a user's past activity, we implemented a number of detectors that alert when anomalous events occur. These include the size of a document printed, the number of documents downloaded, the number of search queries issued, and the like. One such detector alerts when a user browses an anomalous number of documents in a 15-minute period.

In the environment we monitored, in 15-minute periods, people often browsed few documents and rarely browsed many documents. Using a χ^2 test of goodness of fit, we determined that the number of documents browsed in 15-minute periods follows a folded-normal distribution [8].

For a given time period and user, the detector calculates the maximum number of browses for the user during a 15-minute interval within the time period. The detector then retrieves the number of browses during each 15-minute period going back a certain number of days from the start of the time period. It then estimates the parameters of a folded-normal distribution [8], the mean, the standard deviation, and the number of nonzero 15-minute intervals. Then, using the density function, it computes the probability that the user would conduct the maximum number of browses observed in the time period. If the probability is below a threshold, which we determined with the help of domain experts, then the detector alerts. We also implemented a version that uses a kernel-density estimator [9].

Retrieving Documents Outside of One's Social Network. Insiders often steal information to which they have access, but that is outside the scope of their duties, and thus, is not closely associated with them—closely associated in terms of topic and the information's owners and originators at individual and organizational levels. If the organization discovers that its information has been compromised, then this disassociation makes it more difficult to determine the leak's source.

For each individual of the organization, we automatically built a social network based on the people in their department, whom they e-mailed, and with whom they worked on projects. With nodes corresponding to people, we used unweighted directed arcs to represent these associations. We then examined the

extent to which individuals retrieved documents from the public directories of people inside and outside their social network.

Over a period of five months, we tallied the number of documents that each user retrieved during each 15-minute interval. We then expressed this count as the percentage of documents retrieved from others who were outside the user's social network. Subject-matter experts selected as a threshold the percentage that they considered excessive. We built a detector that, when invoked, constructs a social network for each user and counts the number of documents retrieved from outside this network. If the count surpasses the threshold, then the detector alerts.

4.2 Bayesian Network for Ranking

For a given user, ELICIT's 76 detectors may alert in any combination. Presently, if a detector alerts, it simply reports true, so there are 2^{76} possible combinations of alerts. It is unlikely that any analyst would be able to understand such a set of alerts for all but the smallest of organizations or groups of users.

We wanted ELICIT to rank each user of the organization using a *threat score*. Naturally, each user's score would be based on the alerts that his or her activity produced. The simplest method would be to use as a score the total number of alerts, but alerts are not equally predictive of insider behavior, and benign users may engage in many of the same activities as does an insider. We considered asking experts to weight the alerts based on their correlation to malicious behavior, but this brought up the issue of how to combine weights, especially when detectors do not alert and there is an absence of evidence. There also may be other "external" events that cause benign users to change their behavior. For example, a task force created in response to a crisis may produce anomalous activity, such as searching, browsing, and printing during odd hours.

To cope with these challenges, with the help of domain experts, we designed and constructed a Bayesian inference network [10]. Our early designs, while accurate, were too complex, especially when we considered the task of eliciting probabilities from analysts. We settled on a three-level, tree-structured network (see Fig. 1) consisting of Boolean random variables.

The first level consists of one node for the random variable *MaliciousInsider*. The second and third levels correspond to the activities in which a malicious insider will or will not engage (e.g., using inappropriate search terms) and the detectors of those activities that will or will not alert, respectively. There are 76 nodes in both the second and third levels. The nodes of the second level represent the probability that a user will or will not engage in some activity given that he is and is not a malicious insider. The nodes of the third level represent the probability that a detector will or will not detect such activity given that it does and does not occur on the network.

For nodes of the top two levels, we elicited probabilities from three domain experts, mentioned previously. We conducted several sessions and elicited the conditional probabilities for all of the activities given that the insider was and was not malicious.

For the nodes of the bottom, detector level, we determined the conditional probabilities using either theoretical arguments or empirical methods. For these nodes, we set the probability of detection given that the activity occurs to 1. (Strictly speaking, these probabilities are not 1, and we discuss this issue further in Sect. 6.) To determine the probabilities of false alarm for the detectors, we first assumed the events in our collection are normal. For detectors based on, say, parametric estimators, we set the false-alarm rate based on the threshold that the detector uses to report anomalous events.

For example, a detector that alerts when a user prints an anomalously large number of documents uses an estimator based on a folded-normal distribution [8]. Our experts indicated that they would consider suspicious any number of jobs occurring with a probability of less than .015. Since the number of print jobs for a given user follows a folded-normal distribution and the events in our database are normal, the detector's false-alarm rate is also .015. For other detectors, we determined their false-alarm rate empirically, by calculation or by applying them and counting the number of alarms. For example, consider detectors that alert when activity occurs outside of normal working hours. Since we assumed that the events in our collection are normal, the false-alarm rate for such detectors is the proportion of events that occur outside of normal working hours.

When ELICIT invokes the detectors for a given user, for the detectors that alert, it sets to true the value of the nodes of the third level corresponding to those detectors. It then propagates this instantiated evidence throughout the network, thereby calculating a probability distribution for the node *MaliciousInsider*. We use $P(\mathit{MaliciousInsider})$ as the user's threat score, and if it is above a specified decision threshold (e.g., .5), then ELICIT issues an alert for that user. We store all of this information in the database.

5 Evaluation

When we were ready to evaluate ELICIT, the trusted agent selected a scenario at random, inserted it into the database of events, and told us the month into which it was inserted. We ran ELICIT over the entire month and notified the trusted agent, who scored ELICIT's performance and removed the scenario's events from the database. The trusted agent then reported to the research team ELICIT's rank and threat score for each day the scenario's insider was active.

We evaluated ELICIT on eight scenario executions, and in Table 5, we present results for six of these executions. The table contains the member of the red team who executed the scenario, the day of activity, the number of preceding days of inactivity, the number of sessions per day, and the insider's threat score and relative ranking. Using a decision threshold of .5, we counted as a detection any insider scoring above this threshold. We present results for only six executions because when we analyzed the scenarios after completing ELICIT's evaluation, we noticed that for two of the executions (S3 and S4), an unintentional error in the settings of a user's web proxy routed the scenario's traffic around our sensors. We removed these two executions from further consideration.

Table 5. ELICIT's Performance on Six Scenario Executions

Scenario	Executor	Active Day	Preceding Days of Inactivity	Sessions	Score	Rank	Detect
S1	RT4	1	–	1	.994	1	√
		2	1	1	.999	1	√
		3	2	1	.999	1	√
		4	3	1	.994	1	√
S2	RT1	1	–	3	.999	1	√
S2	RT4	1	–	1	.033	341	
		2	1	2	.999	1	√
		3	3	1	.999	1	√
S3	RT2	1	–	1	.999	1	√
		2	1	1	.999	1	√
		3	1	1	.999	1	√
		4	1	1	.984	2	√
S4	RT2	1	–	2	.999	2	√
		2	3	1	.999	1	√
		3	1	1	.992	2	√
		4	1	3	.999	1	√
		5	2	1	.587	20	√
S5	RT2	1	–	3	.071	149	
		2	1	5	.037	238	

As one can see, ELICIT detected insiders on 16 of the 19 days they were active. Notice that RT1 executed scenario S2 in one day, whereas RT4 executed it on three days spanning one week. With the exception of S5 and RT4's execution of S2, ELICIT detected the insiders on their first day of activity. ELICIT performed poorly on scenario S5, and we discuss the reason for this in the next section.

6 Analysis and Discussion

After completing our evaluation of ELICIT, the red team provided information about the scenarios and their events, which we analyzed along with the threat scores for individual users and for all users. In Fig. 3, we present the threat scores for three users for the month of March. The scores of most users were similar to those pictured in Figs. 3a and 3b, which indicate little or no malicious activity. However, less typical are the scores in Fig. 3c, which spike above .9

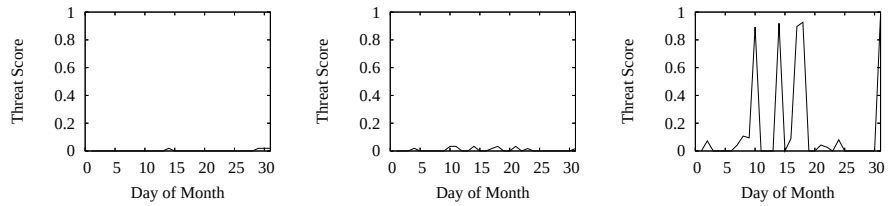

Fig. 3. Threat scores for three users for March

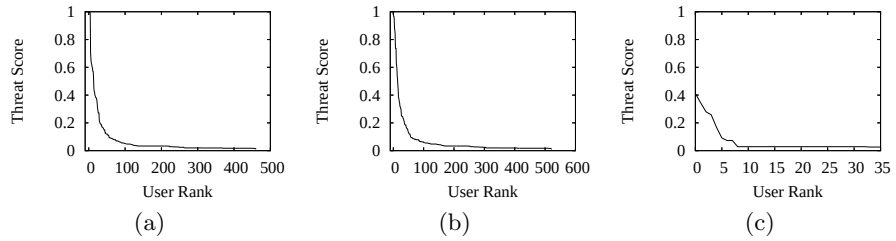

Fig. 4. Threat scores for all users for three days. (a) 462 users for 3/4/05, a Friday. (b) 523 users for 3/7/05, a Monday. (c) 36 users for 3/27/05, a Sunday.

on five different days. In an operational environment, ELICIT would flag and analysts would further investigate individuals with such scores.

In Fig. 4, we show the threat scores for all users on three different days. The scores in Figs. 4a and 4b are from work days, and those in Fig. 4c are from a Sunday (when there are fewer users). These plots are typical and illustrate how ELICIT scores users on different days and how few users obtain relatively high scores. During work days, ELICIT scored an average of $1,548$ users, with 552 users scoring above 0. Of these, 23 users scored above .5.

Using .5 as the decision threshold, we estimated ELICIT's false-positive rate. Although our collection contains activity for $3,938$ distinct users, during March and June, when the red team executed the scenarios, there were $1,548$ active users on average. For these same months, not including the insiders, an average of 23 users scored above .5. Consequently, ELICIT's average false-positive rate is .015. Since ELICIT detected insiders on 16 of the 19 days they were active, its detection rate is .84. We constructed an ROC curve by varying the decision threshold and then approximated the area under the curve using the trapezoid rule, which yielded an area of .92.

We have conducted a thorough analysis of the scenarios and their events, but here, we can present only the key insights. We first examined why ELICIT failed to detect scenario S5, which required the member of the red team to retrieve proprietary software from an internal repository. Although our sensors captured the activity, we had not developed detectors to monitor that specific server. Put simply, our detectors were focused on documents rather than on software. We will address this issue in future work.

ELICIT did not detect the first day of activity of RT4's execution of S2. It consisted of the browsing of a specific financial system and relatively few events (135). As before, we had no detectors tailored expressly for activity involving the financial system. On the first day of the scenario, two detectors alerted on RT4's browsing activity, but these alerts were insufficient to produce a high threat score. However, on the second and third days, there were more events (202 and 306, respectively) and a broader range of activity. This activity produced substantially more alerts—22 on the second day and 20 on the third—and higher threat scores, resulting in detections on those days.

We were concerned that a large number of events might have produced high threat scores. Indeed, of the scenarios we detected, all were in the 75th percentile in terms of the total number of *daily* events (i.e., accounting for both benign and anomalous activity). However, when we examined the correlation between the number of events and the threat scores for all users scoring above the decision threshold in the months of March and June, the coefficients were .026 and −.023, respectively. Consequently, we concluded that ELICIT's detections were not due to the number of events in the scenario executions.

We were also concerned that a red-team member's decision to run all of his or her sessions in a single day (versus one session per day) might have affected the daily threat scores. To investigate, for the days involving multiple sessions, we inserted each session individually and ran ELICIT. ELICIT did not detect the first session of the second day of RT4's execution of S2, and it did not detect the first session of the first day of RT1's execution of S2. On the other hand, evaluating ELICIT on the individual sessions did not result in any new detections. Ultimately, we concluded that a person's decision to execute multiple sessions in a single day did not significantly affect performance.

As for ELICIT's successes, our analysis suggests that based on the scenarios and their executions, the right detectors were alerting on the right activities. Scenarios S1, S2, and S4 involved employees who gathered information unrelated to their duties. Detectors for anomalous volumes of searching and browsing alerted, as did detectors indicating that there were queries atypical for the user.

As for ELICIT's false positives, we found no single consistent pattern of alerts that resulted in false alarms. However, individuals in non-technical administrative roles, representing 10% of the user population, accounted for 39% of all false alarms. Our analysis suggests that administrative staff engaged in many of the activities that ELICIT should detect: large amounts of activity and a breadth of activity spanning organizational boundaries.

Regarding the red-team member's incorrect proxy settings, in practice, insiders could attempt to route traffic around sensors. However, this requires specific, technical knowledge of sensor placement and traffic routing. The insider may obtain little feedback about the success of these countermeasures, and attempting such changes could increase the chance of being detected. In the organizations of interest to us, such changes could be a serious violation regardless of intent. Nonetheless, this illustrates the importance of complementary host-based approaches, which could make such attacks more difficult to launch.

Our results are significantly better than the current state of practice, which involves matching ad hoc patterns, auditing randomly-selected individuals, and auditing based solely on the volume of user activity. When interpreting our results, it is important to keep in mind that this task is quite different than detecting intrusions, which focuses on rule breaking.

False positives are always a concern, but the number of entities (i.e., users) that ELICIT processes per day is orders of magnitude smaller than the number of entities (e.g., connections) that intrusion-detection systems process in much shorter periods of time. Once ELICIT reports a detection, a user's historical ac-

tivity and contextual information play a critical role in subsequent analysis. Such information is largely absent when investigating potential external intrusions. Indeed, ELICIT's interface provides enough information and context about individuals that analysts were able to quickly absolve false positives.

When interpreting the number of false positives, one must also take into account the cost of false negatives, which is substantially higher than that of other detection tasks. At stake is national security. We have not conducted a formal cost analysis. However, anecdotal evidence suggests that, because of the damage these insiders cause, organizations interested in detecting violations of need-to-know are willing to tolerate false positives at much higher rates than with other applications.

Two other important distinctions of this task are the rate of attack and the time over which attacks occur. Rather than occurring in milliseconds (in the case of worms), attacks by insiders who violate need-to-know occur over days, months, and even decades, in the case of Robert Hanssen. Publicly-available information suggests that insider activity occurs in bursts, like other types of attacks, but insider activity is spread over days and months. Consequently, analysts may have to investigate, say, ten false-positives per day, rather than thousands per hour.

As mentioned previously, the probabilities of detection are, strictly speaking, not 1. For example, we did have three days when our network sensors were down, there is a small percentage of events that we could not attribute to users, and there may have been packets that the sensors did not capture. These events certainly affect a detector's probability of detection in some way, but it is unclear whether there is a practical procedure for taking into account all of these factors and then estimating the probabilities. We suspect that most changes would be small and that many would uniformly change the probabilities of detection. This will affect the absolute probabilities, but not the relative probabilities, and we are most interested in a user's rank.

Eliciting probabilities from domain experts proved challenging. They had little difficulty specifying numeric thresholds and conditional probabilities for rules. However, for the detectors based on statistical methods, it was difficult to communicate how the detectors worked in a non-technical manner. Graphical aids and phrasing questions using percentages rather than probabilities helped, but even though all of the experts agreed on the importance of modeling individual activity, we still had trouble eliciting probabilistic cutoffs and conditional probabilities based on these cutoffs. Ultimately, it was easier for us to present and for experts to specify a number rather than a probability or a percentage.

We have attempted to convey that detecting malicious insiders is challenging and different than detecting intruders. One key difference is the availability of contextual information for insiders, information that one rarely has for intruders. With the help of such information, organizations must understand how its users access and manipulate information. To accomplish this, we must attribute actions to users, rather than to IP addresses, which in turn, raises important issues of privacy, especially for researchers.

Complicating matters is the lack of public data sets and information regarding insider behavior and activity. One solution is to engineer data sets. There have been attempts to do so for intrusion detection, mostly notably the MIT Lincoln Labs data set [11], but no similar data set exists for insider threat. Engineered data sets are not without problems, such as guaranteeing that the malicious activity is in correct proportion to the benign activity and that the benign activity is truly representative of the target environment [12]. Our collection of scenarios and information-use events is an attempt to address these concerns for insiders who violate need-to-know.

7 Related Work

We provide only a brief review of related work, but see Chapter 25 of Bishop [13] for a more complete survey. Denning [14] referred to specific instances of such activity as *leakage* and *inference* by legitimate users: Leakage involves a legitimate user leaking or exfiltrating information. Inference is inferring information based on queries to a database or a search engine.

One early attempt to address the problem of misuse was IDES [15], which used statistical profiles of user behavior to detect masqueraders by observing departures from established patterns. (It also applied rules to identify specific intrusions.) Another is UNICORN [16], which examined audit records for misuse by forming profiles using counts over multiple time scales and by applying rules to transform profiles into anomalies, into likely misuse events, and then into alarms. In contrast, ELICIT is geared more toward the misuse of user-level privilege and has a broader notion of context, such as social networks and job descriptions.

Several studies have examined methods of detecting masqueraders from command sequences [5, 17, 18]. The main points of departure between this work and ours are, we are monitoring network traffic; we are interested in legitimate users acting as themselves, but in a manner that is uncharacteristic and inappropriate; finally, to improve detection, we bring to bear contextual information about users and the information they access.

The research most similar to ours is that of Maybury et al. [19]. Workshop participants built a database of 11 million events collected over a period of 3 months from 18 hosts of a 31-node intranet with 75 users. There is overlap with our work, but they examined different sources of information, approaches, and insider profiles.

8 Concluding Remarks

In this paper, we described the construction and evaluation of ELICIT, a system designed to help analysts investigate insider threats. We are interested in malicious insiders who operate within their privileges, but outside the scope of their duties. This is quite different from intrusion detection. We stressed the importance of contextual information and of tracking how individuals access and

manipulate information. One rarely has this information for detecting intruders, but it is critical for detecting insiders.

References

1. United States v. Leandro Aragoncillo and Michael Ray Aquino: Criminal complaint. (District of New Jersey, 9 September 2005)
2. Keeney, M., et al.: Insider threat study: Computer system sabotage in critical infrastructure sector. Technical report, US Secret Service and CERT Program, SEI, CMU, Pittsburgh, PA (May 2005)
3. Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection systems. ACM Transactions on Information and System Security **3**(4) (2000) 227–261
4. Porras, P.A., Neumann, P.G.: EMERALD: Event monitoring enabling responses to anomalous live disturbances. In: Proceedings of the 20th NIST-NCSC National Information Systems Security Conference. National Institute of Standards and Technology, Gaithersburg, MD (1997) 353–365
5. Lane, T., Brodley, C.E.: Temporal sequence learning and data reduction for anomaly detection. ACM Transactions on Information and System Security **2**(3) (1999) 295–331
6. Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. Journal of Computer Security **6**(3) (1988) 151–180
7. Ethereal, Inc.: Ethereal. Software, http://www.ethereal.com (2007)
8. Leone, F.C., Nelson, L.S., Nottingham, R.B.: The folded normal distribution. Technometrics **3**(4) (1961) 543–550
9. Silverman, B.W.: Density estimation for statistics and data analysis. Chapman & Hall/CRC, Boca Raton, FL (1998)
10. Jensen, F.V.: Bayesian networks and decision graphs. Statistics for Engineering and Information Science. Springer, New York, NY (2001)
11. Lippmann, R., et al.: The 1999 DARPA off-line intrusion detection evaluation. Computer Networks **34** (2000) 579–595
12. McHugh, J.: Testing intrusion detection systems. ACM Transactions on Information and System Security **3**(4) (2000) 262–294
13. Bishop, M.: Computer security. Addison-Wesley, Boston, MA (2003)
14. Denning, D.E.: An intrusion-detection model. IEEE Transactions on Software Engineering **SE-13**(2) (1987) 222–232
15. Lunt, T., et al.: IDES: A progress report. In: Proceedings of the Sixth Annual Computer Security Applications Conference. Applied Computer Security Associates, Silver Spring, MD (1990) 273–285
16. Christoph, G.G., et al.: UNICORN: Misuse detection for UNICOS^{TM}. In: Supercomputing '95. IEEE Press, Los Alamitos, CA (1995) 56–56
17. Schonlau, M., et al.: Computer intrusion: Detecting masquerades. Statistical Science **16**(1) (2001) 58–74
18. Maxion, R.A.: Masquerade detection using enriched command lines. In: Proceedings of the International Conference on Dependable Systems and Networks. IEEE Press, Los Alamitos, CA (2003) 5–14
19. Maybury, M., et al.: Analysis and detection of malicious insiders. In: Proceedings of the 2005 International Conference on Intelligence Analysis. The MITRE Corporation, McLean, VA (2005)