A Survey of Insider Attack Detection Research

Malek Ben Salem Shlomo Hershkop
Salvatore J. Stolfo

Department of Computer Science
Columbia University
500 West 120th Street, New York, NY, 10027
{malek,shlomo,sal}@cs.columbia.edu

Abstract

This paper surveys proposed solutions for the problem of insider attack detection appearing in the computer security research literature. We distinguish between masqueraders and traitors as two distinct cases of insider attack. After describing the challenges of this problem and highlighting current approaches and techniques pursued by the research community for insider attack detection, we suggest directions for future research.

1 Introduction

Recent news articles have reported that the cell phones of prominent Greek leg-islators were found to be bugged [26]. Rogue software was injected into the operational systems of the Greek cell phone provider, Vodafone Greece, which controlled a tap for incoming and outgoing calls on selected phones. The phone used by the prime minister and other high ranking officials were apparently targeted. This act was eventually traced to a malicious insider who had hacked the Vodafone system sometime in 2004 and installed the equivalent of a rootkit on an internal Ericsson phone switch. The hack was accidentally discovered through a misconfiguration of a software update a considerable time after the tapping began. The rootkit update accidentally conflicted with other system processes and resulted in alarms being set off in the system. The complexity of the attack could only be attributed to someone with intimate knowledge of the Ericsson switch operating software, which was developed for the last 15 years in Greece.

External threats to the cyber-infrastructure of an organization are constantly evolving. The greatest threat, however, is the problem of insiders who misuse their privileges for malicious purposes. Insider attack has overtaken viruses and worm attacks as the most reported security incident according to a report from the US Computer Security Institute (CSI) [10]. The annual Computer Crime and Security Survey for 2007 surveyed 494 security personnel members from US corporations and government agencies, finding that insider incidents were cited by 59 percent of respondents, while only 52 percent said they had encountered a conventional virus in the previous year.

Much research in computer security has focused on the means of preventing unauthorized and illegitimate access to systems and information. Unfortunately, the most damaging malicious activity is the result of internal misuse within an organization, perhaps since far less attention has been focused inward. Despite classic internal operating system security mechanisms and the literature on formal specification of security and access control policies, including Bell-LaPadula and the Clark-Wilson models [1, 3], we still have an extensive insider attack problem. Indeed in many cases, formal security policies are incomplete and implicit or they are purposely ignored in order to get business goals accomplished. There seems to be little technology available to address the insider threat problem. The state-of-the-art seems to be still driven by forensics analysis after an attack, rather than technologies that prevent, detect, and deter insider attack.

The inside attacker has been defined in many different contexts with no standard definition agreed upon by the research community. How might one then think it is possible to make scientific progress if the problem itself is ill-defined? Nevertheless, there are many well known examples of insider attacks familiar to most people.

For our purposes in this paper, we define a malicious insider to be two classes of malfeasant users; *traitors* and *masqueraders*. A traitor is a legitimate user within an organization who has been granted access to systems and information resources, but whose actions are counter to policy, and whose goal is to negatively affect confidentially, integrity, or availability of some information asset [21]. The traitor uses his/her legitimate credentials when perpetrating their malicious actions, such as in the Greek Vodafone case mentioned above.

The most familiar example of an insider is a masquerader; an attacker who succeeds in stealing a legitimate user's identity and impersonates another user for malicious purposes. Credit card fraudsters are perhaps the best example of masqueraders. Once a bank customer's commercial identity is stolen (e.g. their credit card or account information), a masquerader presents those credentials for the malicious purpose of using the victim's credit line to steal money.

We may distinguish traitors and masqueraders based upon the amount of knowledge each has. A traitor of course has full knowledge of the systems they routinely use and likely the security policies in force. The masquerader may have far less knowledge than the traitor. Furthermore, an insider attack may be due to an innocent mistake by a legitimate user. Hence, insider attack may also be distinguished by intent of the user's actions. Traitors and masqueraders are two sides of what we consider to be the insider threat. The distinction is not entirely satisfactory. After all, a disgruntled insider employee may act as a traitor and a masquerader after stealing the identity of a coworker. But for our present purposes, the distinction is clear enough to consider the general themes of past research in insider attack detection.

An extensive literature exists reporting on approaches that profile user behavior as a means of detecting insider attack, and identity theft in particular. A traitor is presumed to have full knowledge of the internal systems of an organization to which they belong. They use their own credentials and the access granted by those credentials to perform their malicious deeds. A traitor may exhibit normal behavior and still perpetrate malicious acts. Profiling user behavior in this case may seem less relevant except for identifying subtle but significant changes in a user's normal behavior. A masquerader, on the other

hand, has stolen someone's credentials, and is unlikely to know the behavior of their victim. Thus, even though they control the victim's credentials that grant access to whatever the victim is authorized to use, the masquerader is likely to perform actions inconsistent with the victim's typical behavior.

Behavior is not something that can be easily stolen. Stealing someone's credit card information does not reveal the amount and frequency of what the victim typically buys and from whom. Hence, if one profiles the typical buying patterns of a customer (and keeps this historical information secret) an identity thief, a masquerader, has a relatively low probability of misusing the stolen quarry in a manner consistent with the victim's behavior that will go unnoticed. Fraudulent transactions are thus fairly easy to detect even given proper credentials and credit availability. It is this observation that the credit card companies recognized a couple of decades ago when designing early fraud warning systems, and this idea has largely been the driving theme for much subsequent research on masquerade detection.

On the other hand, a traitor is presumably behaving normally and hence profiling a user to detect significant change as a means of detecting malicious actions may not be the best strategy for detecting this class of insider attack. The intelligence and military communities are challenged with detecting traitors and have devised a host of means of using decoys and trap-based defenses to entice and trick users into revealing their nefarious actions. Far less work has been reported in the computer security literature on developing decoy network defenses beyond early work on honeypots and general ideas on the use of honeytokens of various forms. The detection of traitors is an area ripe with challenges begging for new research.

In the following sections, we provide a general overview of the literature on the insider problem driven primarily by various methods of profiling user actions and the systems they use. Much of the work reports on studies describing various audit sources and algorithms to profile users that are tested on simulated masquerade attack data. Researchers have also distinguished between network-level and host-level detection systems. Most of this work is specific to masquerade attack detection, although some work is reported on trap-based defenses aimed to the traitor detection problem using honeypots and honeytokens. We conclude with a view of what we see as the state-of-the-art of the insider attack detection problem, and we provide recommendations on future research directions.

2 Insider Attacks

In order to understand how to detect malicious insider actions, we have to understand the many forms of attack that have been reported [25]. For example:

- Unauthorized extraction, duplication, or ex-filtration of data
- Tampering with data (unauthorized changes of data or records)
- Destruction and deletion of critical assets
- Downloading from unauthorized sources or use of pirated software which might contain backdoors or malicious code

- Eavesdropping and packet sniffing
- Spoofing and impersonating other users
- Social engineering attacks
- Misuse of resources for non-business related or unauthorized activities
- Purposefully installing malicious software

Each of these actions can be considered malicious, but not every one of them may leave an audit trail which can be easily accessed. Several of these actions do leave some trail in some log file which can be linked to the actions of a user after the fact. Hence, when a malfeasance is detected, there is some hope forensics could lead to the perpetrator. Log analysis remains the state-of-the-art in insider attack detection, after a breach has been discovered. Naturally, sophisticated attackers may expend much effort trying to cover their tracks and attacking the logging or auditing sources to remain stealthy. If an organization is not actively monitoring their systems (and users) with sufficient controls preventing tampering with monitor logs, an inside attacker will undoubtedly rarely be detected.

In an insider threat study in the banking and finance sector, Randazzo et al. [27] list the characteristics of insider attacks. Their analysis of validated cases of insider attack indicated that:

- Most incidents required little technical sophistication
- Actions were planned
- Motivation was financial gain
- Acts were committed while on the job
- Incidents were usually detected by non-security personnel
- Incidents were usually detected through manual procedures

These observations should motivate any organization to field monitoring systems to have any hope of automatically and reliably detecting, and deterring, insider attack. We note from this study that most insider attacks on hosts seem to occur at the application level and not at the network-level and hence host-based monitoring is not a desiderata, it is a requirement.

When monitoring systems to mitigate the insider threat, one can collect audit data at either host level activity, network level activity, and or a combination of the two. The main consideration is scalability versus coverage. Hosts sensors are hard to deploy, network sensors are relatively easy to install. Many of the insider problems do not even touch the network level. Schultz pointed out that not one approach will work but solutions need to be based on multiple sensors to be able to find any combination of features to detect insiders [29]. Models to detect insider threats will only be as good as the data collected.

3 Detecting Insider Attacks

3.1 Host-based User Profiling

Understanding the intent of some user action is important to mitigate the insider attack problem. Once an attack has taken place, an investigator needs to reconstruct the intent of the attacker from the audit source. This is a slow and manual process which cannot be easily generalized to pre-attack analysis. Rules might be able to be crafted to cover known attacks, but sophisticated attackers will find new ways and new attack methods to fly under the radar. In addition, the task of keeping rules or profiles updated to the latest threat is a significant challenge to using a host-based protection scheme.

One approach reported in the literature is to profile users by the commands they issue (among the first is [6]). In the general case of computer user profiling, the entire audit source can include information from a variety of sources:

- Command line calls issued by users
- System call monitoring for unusual application use/events
- Database/file access monitoring
- Organization policy management rules and compliance logs

The type of analysis used is primarily the modeling of statistical features, such as the frequency of events, the duration of events, the co-occurrence of multiple events combined through logical operators, and the sequence or transition of events. However, most of this work failed to reveal or clarify the user's intent when issuing commands. The focus is primarily on accurately detecting change or unusual command sequences. We begin with a survey of a collection of papers whose primary focus is command sequence analysis.

3.1.1 Modeling Unix Shell Commands

A hybrid high-order Markov chain model was introduced by Ju and Vardi [12]. A Markov chain is a discrete-time stochastic process. The goal of the work is to identify a 'signature behavior' for a particular user based on the command sequences that the user executed. In order to overcome the high-dimensionality, inherent in high-order Markov chains, a 'mixture transition distribution' (MTD) approach is used to model the transition probabilities. When the test data contains many commands unobserved in the training data, a Markov model is not usable. Here, a simple independence model with probabilities estimated from a contingency table of users versus commands may be more appropriate. The authors used a method that automatically toggled between a Markov model and an independence model generated from a multinomial random distribution as needed, depending on whether the test data were 'usual' (i.e. the commands have been previously seen), or 'unusual' (i.e. Never-Before-Seen Commands or NBSCs).

Schonlau et al. applied six masquerade detection methods to a data set of "truncated" UNIX shell commands for 70 users [28] collected using the UNIX acct auditing mechanism. Each user had 15,000 commands collected over a period of time ranging between a few days and several months. 50 users

were randomly chosen to serve as intrusion targets. The other 20 users were used as simulated masqueraders. The first 5000 commands for each of the 50 users were left intact or 'clean', the next 10,000 commands were randomly injected with 100-command blocks issued by the 20 masquerade users. When commands are grouped into blocks of 100 commands each, the block is either "'clean"', or 'dirty', that is all 100 commands were originated by a masquerader. The complete data set and more information about it can be found at http://www.schonlau.net. The objective was to accurately detect the 'dirty' blocks and classify them as masquerader blocks. This data set was widely used by several authors that investigated different detection methods and has served as the standard benchmark dataset for this line of research.

One detection method explored by Schonlau, called 'uniqueness' relies on the fact that half of the commands in the training data are unique (i.e. used by one user only), and many more are unpopular amongst the users (i.e. used only by a few users). The second method investigated was the Bayes one-step Markov approach. It was based on one step transitions from one command to the next. The approach, due to DuMouchel [7], uses a Bayes factor statistic to test the null hypothesis that the observed one-step command transition probabilities were consistent with the historical transition matrix. The two hypotheses modeled were the null hypothesis, which assumed that the observed transitions probabilities stem from the historical transition matrix, and the alternative hypothesis which assumed that they were generated from a Dirichlet distribution.

A hybrid multi-step Markov method similar to the one introduced by Ju and Vardi [12] is also used. The fourth method used, called the compression method, was based on the premise that test data appended to historical training data compressed more readily when the test data stemmed from the very same user rather than from a masquerader, and was implemented through the UNIX tool 'compress' which implements a modified Lempel-Ziv algorithm.

IPAM (Incremental Probabilistic Action Modeling), another method applied on the same dataset, and introduced by Davidson and Hirsch [6] was also based on one-step command transition probabilities estimated from the training data. The probabilities were continuously updated following an exponential decay scheme with the arrival of a new command.

The sequence-match approach was presented by Lane and Brodley [15]. For each new command, a similarity measure is computed between the 10 most recent commands and a user's historical profile. A user's profile consisted of command sequences of length 10 that the user had previously used. The similarity measure was a count of the number of matches in a command-by-command comparison of 2 command sequences with a greater weight assigned to adjacent matches. This similarity measure was computed for the test data sequence paired with each command sequence in the profile.

Maxion and Townsend applied a naïve Bayes classifier, which had been widely used in text classification tasks, to the same data set [19]. Maxion provided a thorough and detailed investigation of classification errors of the classifier in a separate paper [20], highlighting why some masquerade victims were more vulnerable than others, and why some masqueraders were more successful than others. Killourhy and Maxion also investigated a shortcoming of the naïve Bayes classifier when dealing with NBSCs [13].

The semi-global alignment method presented by Coull et al. [4] is a mod-

Table 1: Accuracy performance Summary of Anomaly Detectors Using the Schonlau Data Set

Method	**False Alarms (%)**	**Missing Alarms(%)**
Uniqueness	1.4	60.6
Bayes one-step Markov	6.7	30.7
Hybrid multi-step Markov	3.2	50.7
Compression	5.0	65.8
Sequence Match	3.7	63.2
IPAM	2.7	58.9
Naïve Bayes (Updating)	1.3	38.5
Naïve Bayes (No Updating)	4.6	33.8
Semi-Global Alignment	7.7	24.2
Eigen Co-occurrence Matrix	2.5	28.0
Naïve Bayes + EM	1.3	25.0

ification of the Smith-Waterman local alignment algorithm. It uses a scoring system that rewards the alignment of commands in a test segment, but does not necessarily penalize the misalignment of large portions of the signature of the user.

Another approach called a self-consistent naïve Bayes classifier is proposed by Yung [38] and applied on the same data set. This method was a combination of the naïve Bayes classifier and the EM-algorithm. The self-consistent naïve Bayes classifier is not forced to make a binary decision for each new block of commands, i.e. a decision whether the block is a masquerader block or not. Rather, it assigns a score that indicates the probability that the block is a masquerader block. Moreover, this classifier can change scores of earlier blocks as well as later blocks of commands.

Oka et al. had the intuition that the dynamic behavior of a user appearing in a sequence could be captured by correlating not only connected events, but also events that were not adjacent to each other, while appearing within a certain distance (non-connected events). With that intuition they developed the layered networks approach based on the Eigen Co-occurrence Matrix (ECM) [24, 23]. The ECM method extracts the causal relationships embedded in sequences of commands, where a co-occurrence means the relationship between every two commands within an interval of sequences of data. This type of relationship cannot be represented by frequency histograms nor through n-grams.

Table 1 presents the estimated accuracy of the classification methods which are all based on a two-class supervised training methodology whereby data is labeled as self or non-self. The Schonlau data used is a mixture of command sequences from different users. The classifiers produced in these studies essentially identify a specific user from a set of known users who provided training data. Furthermore, mixing data from multiple users to train classifiers to detect masqueraders is complicated and fraught with problems. Besides potential privacy threats, requiring the mixture of data from multiple users requires substantial retraining of classifiers as users join and leave an organization.

In a real-word setting it is probably more appropriate to use a one-class, anomaly detection-based training approach. Wang and Stolfo experimented with one-class training methods in [36] using a naïve Bayes classifier and a Sup-

port Vector Machine (SVM) model of user commands to detect masqueraders. The authors have also investigated SVMs using binary features and frequency-based features. The one-class SVM algorithm using binary features was the best performing classifier among four one-class training algorithms that were analyzed. It also performed better than most of the two-class algorithms listed above, except the two-class multinomial naïve Bayes algorithm with updating. In summary, Wang and Stolfo's experiment confirmed that, for masquerade detection, one-class training is as effective as two-class training.

Szymanski and Zhang [34] proposed recursively mining the sequence of commands by finding frequent patterns, encoding them with unique symbols, and rewriting the sequence using this new coding. A signature was then generated for each user using the first 5000 commands. The process stopped when no new dominant patterns in the transformed input could be discovered. They used a one-class SVM classifier for masquerade detection. Although they presented a weighting prediction scheme for author identification, we will limit our focus here to the masquerade detection application of their approach. The authors used an individual intrusion detection approach with 4 features (the number of dominant patterns in levels 1 and 2, and the number of distinct dominant patterns in levels 1 and 2), as well as a 'communal' intrusion detection approach, where they added new features, such as the number of users sharing each dominant pattern in a block. Again, such an approach demands mixing user data and may not be ideal or easily implemented in a real-world setting.

Dash et al. [5] created user profiles from groups of commands called sequences. 13 temporal features are used to check the consistency of patterns of commands within a given temporal sequence. Probabilities are calculated for movements of commands within a sequence in a predefined reordering between commands. They achieve high accuracy, but also high false positive rates on their experiments.

Seo and Cha [30] experimented with combinations of SVM kernels with some success. They managed to increase the accuracy at the expense of somewhat higher false positives. Tan and Maxion investigated which detector window size would enable the best detection results [35]. They uncovered that the best detector window size was dependent on the size of the minimal foreign sequence in test data, which is not determinable a priori. A foreign sequence is one that is not contained in the alphabet set of the training data, but each of its individual symbols is, whereas a minimal foreign sequence is a foreign sequence that contains within it no smaller foreign sequences.

It has been shown that the Schonlau data set was not appropriate for the masquerade detection task. Maxion lists several reasons [20]. First, the data was gathered over varied periods for different users (from several days to several months), and the number of login sessions varied by user. Second, the source of data is not clear. We do not know whether the users perform the same jobs or are widely spread across different job functions. Moreover, in *acct*, the audit mechanism used to collect the data, commands are not logged in the order in which they are typed, but rather when the application ends. Hence the methods applied that focus on strict sequence analysis may be faulty.

In order to alleviate some of the problems encountered with the Schonlau data set, Maxion applied the naïve Bayes classifier to the Greenberg data set, a user command data set enriched with flags and arguments in [18]. He compared the performance of the classifier on the Greenberg data set by using enriched

commands and truncated commands. The hit rate achieved using the enriched command data was more than 15% higher than with the truncated data. However, the false positives rate was approximately 21% higher as well. Nevertheless, when plotting the ROC curves for both data sets, the one for enriched data runs above the ROC curve for truncated data, showing that a better detection performance can be achieved using the user commands enriched with flags and arguments.

As noted, several types of attributes and statistical features can be used for modeling a user's actions. Ye et al. studied the attributes of data for intrusion detection [37]. The attributes studied included the occurrence of individual events (audit events, system calls, user commands), the frequency of individual events (e.g. number of consecutive password failures), the duration of individual events (CPU time of a command, duration of a connection), and combinations of events, as well as the frequency histograms or distributions of multiple events, and the sequence or transition of events. The goal was to find out whether the frequency property was sufficient for masquerader detection, and if so whether there was a single event at a given time sufficient for detecting a masquerader. Five probabilistic techniques were investigated on system call data: a decision tree, Hotelling's T^2 test, the chi-square test, the multivariate test, and the Markov chain. The data set used was made up of 250 auditable security-relevant events collected by the Solaris Basic Security Module (BSM) and 15 simulated intrusions on the background of normal activities. The investigation confirmed the importance of both the frequency property, and the ordering property of events.

3.1.2 User Profiling in Windows Environments

Less research work has been applied to Windows environments compared to work directed for the Unix environment. Much of the difference lies in the auditing methods available on each platform. Linux apparently has cleaner auditing mechanisms (acct, BSM, etc.) whereas Windows has a plethora of system actions that can be captured by various monitoring subsystems.

Shavlik et al. presented a prototype anomaly detection system that creates statistical profiles of users running Windows 2000 [31]. Their algorithm measures more than two-hundred Windows 2000 properties every second, and creates about 1500 features from the measurements. The system assigns weights to the 1500 features in order to accurately characterize the particular behavior of each user - each user thus is assigned his or her own set of feature weights as their unique signature. Following training, each second all of the features 'vote' as to whether or not it seems likely that an intrusion has occurred. The weighted votes 'for' and 'against' an intrusion are compared, and if there is enough evidence, an alarm is raised.

Nguyen, Reiher & Kuenning propose detecting insider threats by monitoring system call activity [22]. Instead of building profiles on system call traces, they analyze relationships between users and files, users and processes, and processes and files. They build user-oriented models as well as process-oriented models using file system and process-related system calls exploiting the regularity in the patterns of file accesses and process-calling by programs and users. They focus on building a Buffer-overflow Detection System (BDS), which is able to detect buffer overflows in many cases, but only if they occur in a set of programs that

have a fixed list of children, i.e. only 92% of programs. The authors' approach, as they point out, was not suitable for detecting malicious insider activity on laptops, because the traces collected on laptops are very dynamic and users do not have a fixed pattern of working time which could be used to define an adequate time window for analysis.

Jha et al. present a statistical anomaly detection algorithm that has the potential of handling mixtures of traces from several users (this will occur when several users are colluding) by using mixtures of Markov chains [11]. The technique which has an unobserved or hidden component can be compared to Hidden Markov Models (HMMs). The training algorithm for HMMs runs in time $O(n \times m^2)$, where n is the number of states in the HMM and m is the size of the trace, whereas, the training time for Markov chains is $O(m)$. So the authors' approach was less computationally expensive than HMMs.

Li and Manikopoulos [16] explored modeling user profiles with SVMs using audit data from a Windows environment gathered over a year. They model the sequence of windows and processes over time in a manner similar to what a process sensor would see. They simulate attack data by mixing data between legitimate user sessions. They reported some success at modeling the user profiles, but suffer with high false positive rates.

In most of the approaches surveyed above, either user command data or system calls data were used. User command data fail to capture window behavior and do not include commands executed inside a script, whereas system call data are not particularly human-readable, nor easily attributed to direct user action. On the other hand, process table data includes window behavior and anything running in a script, and can be easily interpreted when read by a human. Moreover, window tracing provides information at a level of granularity somewhere between the levels of a command line and a system call, while most of the system noise can be filtered out (a formidable challenge when tracing Windows), which makes it a good candidate for user profiling.

Goldring collected user data consisting of successive window titles with process information (from the process table) for a group of users over 2 years [9]. The combination of data sources allowed use of the process tree structure to filter out system noise. However it complicated the feature selection task. The system reduces the stream of data to a single feature vector that consists of a mixture of different feature types per session. A record is generated each time a new window is opened including information about the window title, and all contents in a window title's bar (a wealth of new information, e.g. subject lines of emails, names of web pages, files and directories). Besides that, the window's process and parent process ID's are saved. The window titles' data allows one to distinguish between the operating system's programs such as Control Panel and find Files, which would not be distinguishable from inspecting the process table alone. Goldring reported no performance results, but rather presented a proof-of-concept system. Even if detailed accuracy results were reported, the datasets used bear little resemblance to other data used by researchers. This highlights another important methodological weakness of this line of research where a paucity of data makes it difficult to know whether advances have been made.

3.1.3 User Profiling in Web Environments

There is a vast literature on data mining methods applied to web user 'click' data for marketing analytics that goes well beyond the scope of this paper. However, some work has been done focusing on web profiling for security problems. Kim, Cho, Seo, Lee, and Cha studied the problem of masquerade detection in a web environment. They focused on 'anomalous web requests generated by insiders who attempted to violate existing security policies given by the specific organization' [14]. They applied SVMs to web server logs and used two different kernels: TinySVM (an implementation of SVM for pattern recognition) and the Radial Basis Function (RBF) kernel. Only simple features were used, i.e. neither session features, nor temporal features were included. Simple features are those related to a single web sever request such as the IP address, the hour of the day, the HTTP method (get, post, put, delete, options, head, and trace), the requested page ID, the request status code, the number of transferred bytes, etc. The results showed that SVMs achieved near-perfect classification rates using simple features only. However, the method used did not handle concept drift well, and failed to generalize the model for two users due to changes in user behavior.

3.1.4 Program Profiling Approaches

Besides user-issued commands, inside attackers may inject programs or infect host systems causing changes in underlying system configurations and program behaviors. Hence, approaches to profiling environments and program executions may have relevance to the insider attack detection problem. Much work in this area is devoted to detection of code injection attacks, too broad a topic to describe here. A few characteristic works are described in the following.

Forrest et al. proposed a real-time online anomaly detection system [8] that mimicked the mechanisms used by the natural immune systems. This is done by monitoring system calls of running privileged processes (profiles were built using normal runs of such programs). The modeling is limited to privileged root processes since they have more access to computer resources than user processes, and they have a limited range of behavior that is quite stable and predictable. A separate database of normal behavior is built for each privileged process. The database was specific to a particular architecture, software version and configuration, local administrative policies, and usage patterns, providing a unique definition of 'self'.

The underlying assumptions are that the sequences of system calls executed by a program are locally consistent during normal operation, and that if a security hole in a program is exploited, then abnormal sequences of system calls will occur. A number of experiments were performed using the normal traces of the *sendmail* and *lpr* processes as examples. The results obtained showed that the behavior of different processes was easily distinguishable using the sequence information alone for these two system programs. Several attacks on the *sendmail* process were tested, such as the sunsendmailcp script, the syslog attack, the lprcp attack script, the decode attack, and the *lpr* attack. Other sources of anomalous behavior tested included unsuccessful intrusion attempts, such as remote attack scripts, called sm565a and sm5x, and error conditions. The results have shown that short sequences of system calls could indeed define

a unique and stable signature, which allows for the detection of common sources of anomalous behavior.

The method proposed is computationally efficient and has very low storage requirements. Many aspects of process behavior are ignored (e.g. parameter values passed to system calls, timing information, and instruction sequences between system calls). Although the approach could enable the detection of several scenarios, such as when a program moves to an unusual error state during an attempted break-in, when an intruder replaces code inside a running program, and when new processes are forked, it would not detect race conditions or masqueraders using another user's account. This work led to a number of derivative ideas explored by the computer security community.

Stolfo et al. [33] present the modeling of accesses to the Windows registry by exploiting regularity in process accesses to Windows registry. They introduced a general purpose algorithm for anomaly detection, the Probabilistic Anomaly Detection (PAD) algorithm, that assumes anomalies or noise are a minority of the training data. PAD was applied to model Registry queries and contrasts with the One-Class Support Vector Machine (OCSVM) algorithm using several different kernels. PAD showed better performance, both in accuracy, and in computational complexity, achieving a 100% detection rate of anomalies with a 5% false positives rate for the particular test sets available for the study.

3.2 Network-Based Sensors

3.2.1 Network Observable User Actions

ARDA sponsored a Cyber Indications and Warning workshop dealing with the insider threat. One of the lessons learned was that in many cases insider threats have authorization to access information but may access information they do not have a 'need to know'. When an insider accesses information that they do not need to know, one may have good evidence of an insider attack. A system for detecting insiders who violate need-to-know, called ELICIT, was developed by Maloof and Stephens [17]. The focus of their work was on detecting activities, such as searching, browsing, downloading, and printing, by monitoring the use of sensitive search terms, printing to a non-local printer, anomalous browsing activity, and retrieving documents outside of one's social network. Five malicious insider scenarios were tested, that represented need-to-know violations. Contextual information about the user identity, past activity, and the activity of peers in the organization or in a social network were incorporated when building the models. HTTP, SMB, SMTP, and FTP traffic was collected from within a corporate intranet network for over 13 months, but no inbound or outbound traffic was gathered. In order to identify the information deemed outside the scope of an insider's duties, a social network was computed for each insider based on the people in their department, whom they e-mailed, and with whom they worked on projects. A Bayesian network for ranking the insider threats was developed using 76 detectors. Subject-matter experts defined the thresholds for these detectors, at which an alarm is set. A single threat score is computed for each user based on the alerts from these detectors.

Identifying specific users from observable network events consumed considerable effort. Event attribution proved to be a major challenge: 83% of events initially had no attribution, and 28.6% of them remained unattributed, even

after the use of two offline methods to determine the originator of a particular event. The evaluation of the system used scenarios that were executed over a short period of time, less than one day. However, attacks by insiders who violate need-to-know usually occur over days, months, and even decades, such as in the case of Robert Hanssen. Therefore, it is important to evaluate the ELICIT system using other scenarios that occur over longer periods of time. In any event, although interesting, the focus of this system is limited to environments and organizations that have a formal policy restricting access to information on a need-to-knowbasis. It is rare that such controls are easily discernible in most organizations.

3.2.2 Honeypots

Honeypots are information system resources designed to attract malicious users. Honeypots have been widely deployed in De-Militarized Zones (DMZ) to trap attempts by external attackers to penetrate an organization's network. Their typical use is for early warning and slowing down or stopping automated attacks from external sources, and for capturing new exploits and gathering information on new threats emerging from outside the organization. These trap-based defenses are also useful for the insider threat.

Spitzner presented several ways to adapt the use of honeypots to the insider attack detection problem [32]. Since insiders probably know what information they are after, and in many cases, where that information is to be found, and possibly how to access it, he recommends implanting honeytokens with perceived value in the network or in the intranet search engine. A honeytoken is 'information that the user is not authorized to have or information that is inappropriate' [32]. This information can then direct the insider to the more advanced honeypot that can be used to discern whether the insider intention was malicious or not, a decision that may be determined by inspecting the insiders interaction with the honeypot. In order to reach such interaction that will be used to gather information, it is important to ensure that the honeypot looks realistic to the insider. Humans have a keen sense of suspicion, and hence the grand challenge for honeypots or any trap-based defense is believability, while preventing poisoning of operational systems.

Honeypots suffer from some shortcomings. First, the inside attacker may not ever use or interact with the honeypot or honeytoken, especially if their identity is known or discovered by the insider. Moreover, if an attacker discovers a honey-pot, he/she can possibly inject bogus or false information to complicate detection.

3.3 Integrated Approaches

Among the first integrated systems devised for the malicious insider detection problem was the one presented by Maybury et al. in [21]. The integrated system used honeypots, network-level sensors for traffic profiling to monitor scanning, downloads, and inside connections, and Structured Analysis, a real-time and top-down structural analysis that uses the models of insiders and pre-attack indicators to infer the malicious intent of an insider. Moreover, several data sources were used in addition to auditing of cyber assets. Physical security logs, such as employee badge readers, were also integrated to keep track of the loca-

tion of a user. The program funding this effort apparently ended prematurely. Insufficient test and evaluations were performed on an approach that seemed quite promising.

3.4 Summary

By way of summary, the papers surveyed report the use of heterogeneous audit sources. Most user profiling techniques designed for use in the Unix or Linux environment used the Schonlau data set, a data set made up of truncated sequences of user commands. We have surveyed all two-class based methods and the few one-class based methods applied to this data set. Other approaches using other data sets, such as the Greenberg data set that includes command flags and arguments, were presented. User commands in Unix and Linux environments are easily captured in and are directly observable user actions. The Schonlau datasets serve as a general benchmark dataset and hence most of the literature has been focused on masquerade detection using Unix commands.

In the Windows operating system environment, a variety of audit sources can be exploited. The range of data available inclues system calls, registry accesses [33] which occur when users execute applications, and a combination of process and windows data (window title, how long a window has been open, etc.).

On the network level, the observables are more distant from a distinct user. Attributing of a network level event to a distinct user is a hard. Detecting masqueraders from network level data alone remains a challenge. However network level events are valuable in detecting malicious or unusual activities such as massive downloading of information that the insider does not have a need to know, or the dissemination of information outside the organization's network.

In the reports appearing in the research literature it appears that the data used for training is real data acquired from real sources. However, for testing of proposed detection methods, most authors had to resort to simulated attacks. For instance, Maloof and Stephens asked a red team to perform some attacks based on pre-defined scenarios, and Schonlau used normal user data injected into a different user's data set to serve as a simulated masquerade data. That is hardly a real masquerade attack.

The approaches used also depend on the type of insider problem tackled. For masquerade detection the approach of choice was host-based user profiling, whereas for traitor detection other approaches, such as host-based program profiling using systems calls or registry access data, were used to detect the malicious activity on a system. Network-level sensors were used for traitor detection by Maybury et al. and by Maloof and Stephens, whose approach seems promising for the detection of need-to-know violations. There have been a limited number of reports on trap-based, or honeypot-based, detection methods for the insider problem.

Of particular note is the difficulty of comparatively evaluating competing methods and approaches. This is primarily due to the lack of a uniform test data with known ground truth. Although, the Schonlau data set has been widely used by many authors, it has been shown that it is far from being suitable for an objective evaluation of the insider attack detection algorithms.

Table 2 represents a broad summary of the assumptions on how a specific audit source may contribute to detecting masqueraders or traitors gleaned from

Table 2: Summary of Insider Approaches and Suitability of Audit Mechanism

	Masquerader)	**Internal Traitor**
Two Class: Unix Command Sequences	High - Unfamiliar with local setup	Low - Can mimic another normal user
One Class: Unix Command Sequences	High - Unfamiliar with local settings	Medium - Malicious actions may deviate from norm
Unix Audit Events	Medium - Has credentials and might not trigger alerts	Low - Application level malicious acts may not manifest as unusual
Unix System Calls	Medium - Might not violate system call profile	Low - Application level malicious acts may not manifest as unusual
Window Usage Events	High - May use Windows in unusual ways	Medium - May use Windows Applications in unusual ways
Windows Registry access	Medium - unless malicious programs access Registry	Medium - unless malicious programs access Registry
Network Activity Audit	Medium - If attack occurs over network	Medium - If attack uses network

the surveyed research papers. Each cell of the table represents our opinion about how well a specific approach may be suitable as an audit source to detect masqueraders or traitors. For example, a masquerader is more likely to trigger unusual behavior models by executing commands that are unusual for the victim whose credentials they have stolen. That assumption has driven a considerable amount of research activity. Network-level audit sources are assumed helpful in detecting anomalous application-level violations, such as ex-filtration of data. No formal study of each audit source has been reported validating or refuting these assumptions. However, this table may serve as a guide for such future research.

4 Future Research Directions

User profiling as a means of identifying abnormal user behavior is well established as a primary methodology for masquerader attack detection. As we have noted, a masquerader impersonates another persona and it is unlikely the victim's behavior will be easily mimicked. Hence, abnormal behavior is a good indicator of a potential masquerade attack as a consequence of identity theft. User profiling may also be useful in detecting a traitor, if subtle but significant changes in a user's behavior indicate a malicious activity. We believe that it will be important to derive user profile models that reveal user intent in order to hone in on insider actions that are suspicious and likely malicious. It may not be enough to know of a malicious act merely from knowing that a user has issued an abnormal command sequence unless that sequence could violate a security policy. For example, we conjecture that modeling a user's search behavior may be one way of capturing a user's intent to seek information for malicious

purposes, something that a masquerader, and possibly a traitor, is likely to do early in their attack behavior. Too much searching, or searching in abnormal directories or locations, seems more than odd, it may seem sinister in intent.

A major challenge of insider attack detection research is the lack of real data in order to study general solutions and models. It is hard, if not impossible, to collect data from normal users in many different environments. It is especially hard to acquire real data from a masquerader or traitor while performing their malicious actions. Even if such data were available, it is more likely to be out of reach and controlled under the rules of evidence, rather than being a source of valuable information for research purposes. Because of the scarcity of real data, Chinchani et al. created RACOON [2], a system for generating user command data for anomaly detection from customizable templates representing particular user profiles. However, the system suffered from the same shortcomings of most simulated data: Even though noise was introduced into the simulated data, that noise still followed a predictable distribution.

It is hard to obtain real intrusions for ground truth test and evaluation for a number of reasons:

- Researchers generally do not have direct access to real attacks
- Attacks may be undetected and thus unavailable for study
- Organizations do not admit that they were attacked and hence shy away from cooperating with researchers
- Attacks might be mistaken for incompetence

Given these challenges, devising capture the flag exercises to generate insider attack datasets that are realistic in nature may provide a means of advancing the state-of-the-art in understanding and solving the insider threat.

It is generally unknown what types of audit sources are most discriminatory to reliably detect insider malicious behavior. Moreover, it is not obvious what amount of data is needed for modeling, nor how long the training or data collection period should be.

We posit that malicious insider actions on computer systems are likely to occur at the application level. For instance, a customer service employee in a call center may access more customer records on one particular day than he/she typically accesses on other days, possibly to commit a crime to sell confidential information. Detecting such unusual events can only occur at the business application level, and application-level knowledge is needed to understand the user's intent and confirm whether the intent of user actions is possibly malicious. This can be detected using host-based sensors, and possibly through network-based sensors if the application is accessed remotely.

The most vexing problem for researchers is to devise detection methods that accurately distinguish between the cases where an insider attack is verified with high confidence versus cases where an insider attack is inferred from partial knowledge of possibly suspicious actions. Distinguishing false positives from true positives in the presence of uncertainty is particularly challenging when people's reputations are at stake. Hence, we also believe that any technologies developed to detect insider attack have to include strong privacy-preserving guarantees to avoid making false claims that could harm the reputation of individuals whenever errors occur.

Another important topic for research is the investigation of alternative mitigation strategies. For instance, how does a monitoring or detection system challenge a user when the system detects what it believes are malicious activities? How might a system alert a supervisor of a possible attack without disclosing an employee's true identity unless and until an attack has been validated?

Beyond the significant challenges in computing accurate user profiles, considerable effort is needed on developing techniques for trapping traitor behaviors. We believe a major challenge will be to develop and inject bogus data and information that is believable to sophisticated humans with full knowledge of an organization's internal systems without negatively impacting operations. How does one develop a trap for those, who are aware that such technology is in use, and do so, without poisoning the operations of the business functions conducted by insiders, who may get inadvertently confused by realistic, but bogus information?

5 Conclusion

Insider threat detection is a nascent research field begging for new approaches and new research methodologies. A plethora of modeling algorithms are available as well as a wealth of audit sources that can be acquired effectively. However, building effective systems for detecting insider attacks remains an open challenge. The lack of ground truth data limits the potential value of various proposed solutions since progress is so hard to measure. Even so, much work has been published using 'simulated' masquerade attack data. We surveyed the different machine learning and modeling algorithms applied to masquerader attack detection using host-based and network based audit sources. There has been a modest amount of work in the area. However, the best audit sources and most discriminating features one might use in automated systems to detect masquerader are still unknown. The experimental methodology has been generally weak. Although, there have been many methods proposed, their utility is uncertain, and none of them is clearly superior to all others. Although one dataset, the Schonlau dataset, has been useful for a community of researchers to use in comparative evaluations, that dataset itself is insufficient to conduct realistic evaluations. The data set is limited in scope of information it provides, and does not contain true insider attack command sequences. At best, the dataset may be useful to compare computational performance between competing algorithms, but accuracy is not measurable.

By way of summary, new methods of detecting insider attack, whether by trai-tor or masquerader, remains an open and active area of research, and we expect it to be so for some time to come.

References

[1] D. Elliott Bell and Leonard J. Lapadula. Secure computer systems: Mathematical foundations, 1973.

[2] Ramkumar Chinchani, Aarthie Muthukrishnan, Madhusudhanan Chandrasekaran, and Shambhu Upadhyaya. RACOON: Rapidly generating user

command data for anomaly detection from customizable templates. *Computer Security Applications Conference, Annual*, 0:189–204, 2004.

[3] D. D. Clark and D. R.Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pages 184–194. IEEE Computer Society Press, 1987.

[4] S. E. Coull, J. Branch, B. Szymanski, and E. Breimer. Intrusion detection: A bioinformatics approach. In *Proceedings of the 19th Annual Computer Security Applications Conference*, pages 24–33, 2001.

[5] Subrat Kumar Dash, Sanjay Rawat, G. Vijaya Kumari, and Arun K. Pujari. Masquerade detection using ia network. In *First International Workshop on Applications of Constraint Satisfaction and Programming to Computer Security Problems*, pages 18–30, 2005.

[6] B. D. Davison and H. Hirsh. Predicting sequences of user actions. In *Working Notes of the Joint Workshop on Predicting the Future: AI Approaches to Time Series Analysis, 15th National Conference on Artificial Intelligence/15th International Conference on Machine Learning*, pages 5–12. AAAI Press, 1998.

[7] William Dumouchel. Computer intrusion detection based on bayes factors for comparing command transition probabilities. Technical report, In National Institute of Statistical Sciences, 1999.

[8] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 120–128. IEEE Computer Society Press, 1996.

[9] Tom Goldring. Authenticating users by profiling behavior. http://www.galaxy.gmu.edu/interface/i03/i2003html/goldringtom/goldringtom.presentation.ppt. In *DMSEC'03: ICDM Workshop on Data Mining for Computer Security*, 2003.

[10] Lawrence A. Gordon, Martin P. Loeb, William Lucyshyn, and Robert Richardson. CSI/FBI computer crime and security survey, 2006.

[11] S. Jha, L. Kruger, T. Kurtzy, Y. Leez, and A. Smith. A filtering approach to anomaly and masquerade detection, 2004.

[12] Wen-Hua Ju and Yehuda Vardi. A hybrid high-order markov chain model for computer intrusion detection. *Journal of Computational and Graphical Statistics*, June, 2001.

[13] K. S. Killhourhy and Roy Maxion. Investigating a possible flaw in a masquerade detection system. Technical report, University of Newcastle upon Tyne, Technical Report CS-TR-869, 2004.

[14] H. Kim, S. Cho, J. Seo, Y. Lee, and S. Cha. Use of support vector machine (svm) in detecting anomalous web usage patterns. In *Symposium on Information and Communications Technology*, 2004.

[15] T. Lane and C. E. Brodley. Sequence matching and learning in anomaly detection for computer security. In *In AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, pages 43–49. AAAI Press, 1997.

[16] L. Li and C.N. Manikopoulos. Windows NT one-class masquerade detection. In *Proceedings of the Fifth Annual IEEE SMC Information Assurance Workshop*, pages 82–87, June 2004.

[17] M. A. Maloof and G. D. Stephens. elicit: A system for detecting insiders who violate need-to-know. In *RAID*, pages 146–166, 2007.

[18] R. A. Maxion. Masquerade detection using enriched command lines. *Dependable Systems and Networks, International Conference on*, 0:5, 2003.

[19] R. A. Maxion and T. N. Townsend. Masquerade detection using truncated command lines. In *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 219–228. IEEE Computer Society, 2002.

[20] R. A. Maxion and T. N. Townsend. Masquerade detection augmented with error analysis. *IEEE Transactions on Reliability*, 53(1):124–147, 2004.

[21] Mark Maybury, Penny Chase, Brant Cheikes, Dick Brackney, Sara Matzner, Tom Hetherington, Brad Wood, Conner Sibley, Jack Marin, and Tom Longstaff. Analysis and detection of malicious insiders. In *Proceedings of the International Conference on Intelligence Analysis*, 2005.

[22] Nam Nguyen, Peter Reiher, and Geoffrey H. Kuenning. Detecting insider threats by monitoring system call activity. In *Proceedings of the 2003 IEEE Workshop on Information Assurance*, pages 18–20. United States Military Academy West Point, 2003.

[23] M. Oka, Y. Oyama, H. Abe, and K. Kato. Anomaly detection using layered networks based on eigen co-occurrence matrix. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, 2004.

[24] M. Oka, Y. Oyama, and K. Kato. Eigen co-occurrence matrix method for masquerade detection. In *Publications of the Japan Society for Software Science and Technology*, 2004.

[25] A. H. Phyo and S. M. Furnell. A detection-oriented classification of insider it misuse. In *Proceedings of the Third Security Conference*, 2004.

[26] Vassilis Prevelakis and Diomidis Spinellis. The athens affair. *IEEE Spectrum*, 44:7:26–33, 2007.

[27] M. R. Randazzo, M. Keeney, E. Kowalski, D. Cappelli, and A. Moore. Insider threat study: Illicit cyber activity in the banking and finance sector, 2004.

[28] M. Schonlau, W. Dumouchel, W. Ju, A. F. Karr, M. Theus, and Y. Vardi. Computer intrusion: Detecting masquerades. *Statistical Science*, 16:58–74, 2001.

[29] E. Eugene Schultz. A framework for understanding and predicting insider attacks. *Computers & Security*, 21(6):526 – 531, 2002.

[30] Jeongseok Seo and Sungdeok Cha. Masquerade detection based on svm and sequence-based user commands profile. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 398–400, 2007.

[31] Jude Shavlik and Mark Shavlik. Selection, combination, and evaluation of effective software sensors for detecting abnormal computer usage. In *KDD '04: Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 276–285, New York, NY, USA, 2004. ACM.

[32] Lance Spitzner. Honeypots: Catching the insider threat. *Annual Computer Security Applications Conference*, 2003.

[33] Salvatore J. Stolfo, Frank Apap, Eleazar Eskin, Katherine Heller, Shlomo Hershkop, Andrew Honig, and Krysta Svore. A comparative evaluation of two algorithms for windows registry anomaly detection. *Journal of Computer Security*, 13(4):659–693, 2005.

[34] Boleslaw K. Szymanski and Yongqiang Zhang. Recursive data mining for masquerade detection and author identification. In *Proceedings of the 13rd Annual IEEE Information Assurance Workshop*. IEEE Computer Society Press, 2004.

[35] Kymie M. C. Tan and Roy A. Maxion. "why 6?" defining the operational limits of stide, an anomaly-based intrusion detector. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 188–202, 2001.

[36] K. Wang and S. J. Stolfo. One-class training for masquerade detection. In *Proceedings of the 3rd IEEE Workshop on Data Mining for Computer Security*, 2003.

[37] Nong Ye, Xiangyang Li, Qiang Chen, S.M. Emran, and Mingming Xu. Probabilistic techniques for intrusion detection based on computer audit data. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 31(4):266–274, Jul 2001.

[38] K. H. Yung. Using self-consistent naïve bayes to detect masqueraders. In *PAKDD'08: Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 329–340, 2004.