Behavioral Analysis of Insider Threat: A Survey and Bootstrapped Prediction in Imbalanced Data

Amos Azaria, Ariella Richardson, Sarit Kraus, and V.S. Subrahmanian

Abstract—The problem of insider threat is receiving increasing attention both within the computer science community as well as government and industry. This paper starts by presenting a broad, multidisciplinary survey of insider threat capturing contributions from computer scientists, psychologists, criminologists, and security practitioners. Subsequently, we present the BAIT (Behavioral Analysis of Insider Threat) framework, in which we conduct a detailed experiment involving 795 subjects on Amazon Mechanical Turk in order to gauge the behaviors that real human subjects follow when attempting to exfiltrate data from within an organization. In the real world, the number of actual insiders found is very small, so supervised machine learning methods encounter a challenge. Unlike past works, we develop bootstrapping algorithms that learn from highly imbalanced data, mostly unlabeled, and almost no history of user behavior from an insider threat perspective. We develop and evaluate 7 algorithms using BAIT and show that they can produce a realistic (and acceptable) balance of precision and recall.

Index Terms—computer security, behavioral models, insider threat

1 Introduction

Insider threat refers to the threat posed to organizations by individuals who have legitimate rights to access the internal system of an organization. In a detailed study [1] of 23 insider threat incidents in the banking and finance sector between 1996 and 2002 which was carried out jointly by the US Secret Service and CERT (at Carnegie-Mellon University), the authors found that insider threat events included fraud, theft of intellectual property and attempts to sabotage the organization's network. The same organizations conducted a similar study focusing on 36 incidents in the government sector during the same time frame that involved document fraud, financial fraud (embezzlement), fraud using a computer, theft of confidential information and/or sabotage, and more. CERT's insider threat page, http://www.cert.org/insider_threat/, presents an excellent summary and several reports detailing various kinds of insider threat. In a similar vein, [2], quoting a US Justice Department survey, states that 74% of all cyber-theft within organizations was carried out by insiders and that 40% of all cyber-incidents reported by 36,000 US businesses involved insiders. According to [3], "personal records harvested from databases can be sold on the open market for $4 - 8£$ per record." Thus, an insider who steals a million data records from a credit card, insurance or health care company can potentially make himself over $10M ! Fyffe [3] describes a US Secret Service operation that busted a criminal network dealing in information on 1.7M credit cards.

Detecting malicious insiders poses a huge challenge for many reasons. First, the number of malicious insiders who have been discovered within a given organization is usually very small, maybe just a handful over a decade. From the perspective of machine learning algorithms, this provides a highly imbalanced data set (over 99.9% of honest users, and usually well under 0.1% of malicious insiders) from which to automatically learn [4]. Learning from an imbalanced dataset poses a great challenge. Machine Learning Algorithms usually assume the data is balanced. Using imbalanced data often results in very high accuracy for the majority class, and very low accuracy for the minority class. In the case of insider threats this type of result is very problematic, as we are interested in detecting the minority class. Second, there is no publicly available comprehensive data set for testing purposes — companies are reluctant to share such data and security organizations cannot. Thus, even training data from real-world sources is hard to come by. Third, even the little training data that is publicly available is flawed — for instance, we know of training data in which the honest users' data

• *Amos Azaria is with the Machine Learning Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 E-mail: {azariaa}@cs.cmu.edu; Ariella Richardson is with the Department of Industrial Engineering at Lev Academic Center, Jerusalem 91160, Israel, E-mail: {richards}@jct.ac.il; Sarit Kraus is with both the Computer Science Department at Bar Ilan University, Ramat Gan 52900 Israel, and with UMIACS at the University of Maryland, College Park, MD 20742, E-mail: {sarit}@umiacs.umd.edu; and V.S. Subrahmanian is with both the Computer Science Department and UMIACS at the University of Maryland, College Park, MD 20742 E-mail: {vs}@cs.umd.edu.*

First Decision Date: July 21, 2014 Received Date: Feb 2, 2014 Revised Date(s): Nov 20, 2014 Accepted Date: Nov 20, 2014 Manuscript number: TCSS-2014-02-0001

is real, but the "malicious insiders" are synthetically injected [5] — moreover, in such cases, the injected insiders are often based on past attacks, making the implicit assumption that future attackers will attack in the same ways that past attackers used. Thus, even the data that was previously claimed as "real" suffers from serious flaws. *One of the fundamental flaws in past studies is that there have been almost no behavioral studies in which real world human users' behavior as potential malicious insiders has been carefully studied.* A deep human study that seeks to understand how people within an organization might try to exfiltrate data *in the future* is critical if we are to prevent such attacks in the future. The only exception we are familiar with is a small scale experiment reported in [6] that we will discuss in section 2.3.2. The fourth challenge that complicates insider threat detection is that the malicious behavior of the insider is only a small portion of the subject's actions. The malicious actions take place alongside other normal behavior that an insider performs as part of his job. This enhances the imbalanced nature of the dataset, since not only are there a small number of malicious subjects, but for these subjects only a small portion of the behavior is malicious.

The goal of this paper is two-fold. First, we give a detailed and broad overview of techniques to detect malicious insiders. Our survey is different from a 2008 survey [7] in that our survey covers a broad swath of multidisciplinary territory — insider threat has been studied over the years by psychologists, criminologists, electrical engineers, and of course, computer scientists. We try to cover all of these efforts in our survey. In addition, we also cover more recent work that was published after the 2008 survey of [7]. Once we complete our survey, we present BAIT (Behavioral Analysis of Insider Threat), in which we develop bootstrapped algorithms that try to learn separators between malicious insiders and honest users under the following conditions: (i) the training data sets are highly imbalanced, (ii) there is very little training data, (iii) the attacks are carried out by real humans similar to employees in an organization without regard to past attacks reported in the literature, and (iv) The attacks are performed alongside other normal behavior.

In order to address these new scenarios, the combination of which has not yet been reported in the literature, we present the BAIT framework. In BAIT, we considered 7 algorithms in total (5 that build on top of Support Vector Machines [8] and 2 that leverage the Multinomial Naive Bayes algorithm). Of these 7 algorithms, 2 are completely obvious, but the remaining 5 represent different ways of generating a larger and more balanced training set by "bootstrapping" on top of a very imbalanced and small training set. In order to test out our algorithms, we designed a BAIT game. The game was tested on Amazon Mechanical Turk and used a set of 795 carefully vetted users from the USA. Past work on insider threat detection has ofteny used real data from an organization to show organizational activity, but injected threat data artificially. This was done in DARPA's ADAMS project [9], in ELICIT[6], and in related efforts. In contrast, our work complements this exactly — our injection of threat is real, but we were unable to test it within a real organization.

The game mimicked the "insides" of an organization and assigned all users a role (malicious insider vs. honest user) as well as a set of possible actions they can take. These actions are similar to those in other studies in the literature. Based on these, we define a set of 28 features that can be used by our SVM-based and Naive-Bayes-based methods. We evaluate the efficiency of the 7 BAIT algorithms via a series of experiments in order to determine which algorithm exhibits the best accuracy. As usual, precision and recall tradeoff against one another, but our best algorithm is able to deliver a recall of 0.6 and a precision of 0.3 (for a total F-measure of 0.4). This means that we were able to correctly identify 60% of the malicious insiders, while guaranteeing analysts that about one of three suspects presented to them would indeed be a malicious insider. However, if we wish to make a different tradeoff in order to increase recall (which is a major factor), then another one of our algorithms yields 70% recall, but only a 7% precision. This means that if we are willing to live with only about 1 in 15 suspects being a true malicious insider, then we can identify an extra 10% of malicious insiders. Whether this tradeoff is worth it in a real application (e.g. within an FBI-like organization with 35K+ employees) depends upon the resources available to investigate insider threat. We suspect they would rather have a 60% recall and a 30% precision, meaning that about one of three individuals the system flags (and that they subsequently investigate manually) is truly malicious.

We also present findings about behaviors of users that are statistically significant indicators of being insider threat risks — our findings on the behaviors that distinguish between malicious insiders and benign insiders include some new behavioral results and independently validate some prior results of other researchers. In the case of two findings of other researchers, our independent experiments leave open questions: specifically, while we noticed trends similar to some prior reports, they were not statistically significant in our study.

2 Related Work

This section will survey methods from various domains that consider the insider threat detection methods and challenges. We begin with a selection of studies which motivated the study and suggest features to consider for insider threat detection, such

as system usage, user behavior etc. We then proceed to describe psychological and social theories, which profile malicious insiders, and propose systems that use these profiles for insider threat detection. This is followed by a discussion on anomaly detection. Although many of the systems using anomaly detection are aimed at external threat detection rather than insider threat detection (because of the shortage of data available), these systems may inspire the insider detection problem. Honeypots and Graph-based systems are commonly used for threat detections, and are surveyed next. Finally Game theory approaches are presented. Most of the studies surveyed are presented briefly; a small number of studies that we found to be closest to our work are presented in detail.

2.1 Motivating Studies

Magklaras and Furnell [10] present an insider threat prediction tool which performs three major tasks:

1) *Monitoring aspects of file and directory location* in order to account for the fact that certain types of misuse of a computer system are connected to placing certain files in certain directories.
2) *File content analysis* in order to check for certain patterns such as virus signatures within files.
3) *File integrity checks* in order to check if a file has been compromised, e.g. system files or boot files.

They then develop Evaluated Potential Threat (EPT) metrics which characterize user behavior features such as their knowledge of a file system, the content of files in their workspace and the way in which they interact with the network (e.g. histogram of traffic types they receive and/or send). However, no empirical results are reported on the effectiveness of these metrics.

Myers et al. [11] state that insider threats involve two scenarios:

1) *Unauthorized use of privileges* in which a malicious insider tries to access data that he is not authorized to access (e.g. accessing compartments of data that are not relevant to his mission) and/or uses authorized resources in inappropriate ways (e.g. emailing a file he is authorized to access to a person who may not be authorized to see it).
2) *Automated insiders* such as bots, programs that map the internal network and probes that try to identify weaknesses in the system.

Jabbour and Menasce [12] propose the Insider Threat Security Architecture (ITSA) to identify and mitigate insider threat. They present a hypothetical example of a database administrator of an insurance company who has "gone bad" and is trying to steal money from his employer. To do this, he injects a bogus $100K claim into the database, but the security policy in force prevents the payment of the claim because all claims of $100K or more must be audited according to the policy. As a consequence, the DBA changes the claim to $99,999 (or a similar number) and then deletes all traces of this change from the logs. In the ITSA framework, however, there is a "defense" layer that records the DBA's actions and would alert some appropriate entity in the organization of the DBA's attempt to inappropriately change the status of the claim.

Bishop et al. [13], [14] propose a graduated notion of *insiderness*. They introduce a hierarchy of policy abstractions, and argue that the discrepancies between the different layers of abstraction are useful for identifying insider threats. They also present a methodology for analyzing the threat based upon these definitions. They introduce Attribute Based Group Access Control that allows any attributes to define a group. The attributes that they study include job function (e.g., Help Desk) and building access (e.g., after 5pm). They applied this to the insider threat by defining groups based on access capabilities, and used those groups to identify users with a high level of threat with respect to high-risk resources. Again, no empirical studies are reported on the effectiveness of the proposed methodology.

Hunker and Probst [15] present an overview of definitions of insiders and insiders threats and discuss a number of approaches from the technological, the sociological, and the socio-technical domains. Their main conclusion is that tackling insider threats requires a combination of techniques from these domains in order to detect and mitigate insider threat.

Sinclair and Smith [16] discuss the challenges of prevention of attacks using access control. In particular, they discuss the difficulty to balance between allowing the members of organizations in financial, health care, and other enterprise environments to fulfill their tasks successfully and efficiently and the deployment of access control technology to prevent insider threats that interfere in their activities. They conclude by saying that prevention complements (not replaces) detection efforts since better prevention can restrict the problem space that detection must address.

2.2 Psychological and Social Theories

In this section we first present some studies that describe which indicators are likely to appear in malicious insiders. We then survey tools and systems that detect insider threats using psychological and social theories.

2.2.1 Behavioral Indicators

Schultz [17] defines five behavioral indicators that are apparently predictive of an insider seeking to attack a system.

1) *Deliberate markers* refer to the fact that various users may engage in deviant behavior online. They provide an example of a disgruntled employee flooding his supervisor's mail box with threatening emails from an anonymous source.

2) *Meaningful errors* in which a user makes mistakes. For example, a user trying to download various proprietary files may erase relevant log files, but may forget to erase relevant error logs which may contain traces of errors the user made (e.g. mistyped access command) that can help identify him later.
3) *Preparatory behavior* in which a user might use a range of system level commands such as nslookup, whois, and so forth to perform surveillance on a system before carrying out an insider attack.
4) *Correlated usage patterns* in which a user might exhibit a certain behavior on multiple loosely coupled sub-networks or sub-systems which separately do not show a suspicious pattern, but collectively do show a suspicious pattern.
5) *Verbal behavior* in which a user uses hateful language about a supervisor or a company in general.
6) *Personality traits* such as introversion are assumed to be correlated to the likelihood of a user posing an insider threat.

In a similar vein, Wood [18] tries to identify various aspects of users who pose an insider threat. These users are assumed to have the skills needed to attack the system and to be risk averse, usually working on their own — this last aspect is similar to the introversion criterion identified by Schultz [17]. Wood [18] goes on to say that people who mount attacks either have a character defect or work for a competing organization. Warkentin and Willison [19] point out that most paper about insider threat do not consider the behavior of the attacker. In a related paper, Willison and Warkentin [20] propose an "Organizational Justice" model which seeks to understand how various corporate factors can shape feelings of disgruntlement amongst employees. Disgruntled employees are frequently mentioned as potential insider threats [5], [21].

2.2.2 Sytems using Psychological and Social Theories

An excellent paper by Greitzer and Frincke [22] presents a comprehensive view of psychological approaches to detecting insider threat, together with a computational approach that uses Bayesian nets. Based on many years of work, they identify the following factors as predictors of insider threat.

1) *Disgruntlement.* As mentioned earlier, disgruntled employees can often become malicious insiders as in the case of Shakuntala Singla, a former US Coast Guard employee who in 1997, masqueraded as another user and crashed several Coast Guard computers[pages 182-183][23] after becoming disgruntled that her complaints about sexual harassment were not being taken seriously.
2) *Accepting criticism.* This is related to disgruntlement. Users who may not accept criticism of their behavior, work ethic, or quality of work can quickly become disgruntled.
3) *Anger management.* Users who end up getting very angry, sending abusive emails, using foul language both in a verbal and a written (e.g. email or SMS) setting can end up eventually becoming insider threats.
4) *Disengagement.* A user who does not seem to interact much with others in the organization and becomes withdrawn also is believed to pose an insider threat risk.
5) *Disregard for authority.* A user who ignores normal, even non-computer related workplace rules, obviously poses a risk. A person who ignores one set of rules may ignore rules about computer use.
6) *Performance.* Clearly, people who have received poor performance appraisals can end up with an incentive to become malicious insiders.
7) *Stress.* Users who are under stress (e.g. a divorce, financial stress, health related stress) can fall prey to third parties who might leverage their psychological vulnerabilities in order to turn them into malicious insiders.
8) *Confrontational Behavior.* This is similar to the anger management category. Users who are overly aggressive are usually aggressive because they have some level of dissatisfaction with things or individuals in their organization. Thus, confrontational behavior might be a symptom of disgruntlement.
9) *Personal issues.* A user who is unable to separate his/her personal life from their work life may have personal problems causing stress. Thus, this category is related to the stress category listed above.
10) *Self-centeredness.* A user who only thinks about his needs, not the needs of his colleagues at work, is perhaps not going to spend much time thinking about his company's needs either except in so far as they support achieving his own goals. Such a user may be tempted by outside organizations that seek to turn him into a malicious insider by offering him incentives that meet his needs (e.g. money, sex, etc.).
11) *Lack of dependability.* This corresponds to the case where a user makes promises, but is unable to keep them. The rationale here is that if the user cannot keep promises made to his colleagues (e.g. to finish a project on time), then his level of trust within the organization is not high.
12) *Absenteeism.* A user who is chronically absent or late exhibits a poor work ethic and may not care much about the company.

It is important to note that these 12 "predictors" are not likely, by themselves, to be good predictors of insider threat because they would induce very high false positives. For instance, a person who is stressed because of issues with a spouse is unlikely to turn into a malicious insider at his work. Therefore, these indicators merely serve as "signals" of possible future wrongdoing, together with the knowledge that most likely they are wrong.

Based on these intuitions, Greitzer and Frincke [22] proposed a Bayesian net-based predictive model. The details of the model are not spelled out (and we will discuss other Bayesian models due to [5] later in this paper). The authors report that they conducted two types of experiments. In the first, they compared the results of the Bayesian model with the above indicators using two human resources experts who served as adviser to the development of the model and obtained an R^2 value of 0.94, showing an excellent fit. They also asked 3 additional human resources experts to validate the model on a set of 50 cases, but here, the R^2 value was much lower, 0.29. They also tested using synthetic data for 100 employees by injecting "bad guys". Their Contingency Coefficient Analysis yielded a C value of 0.76, indicating a strong correlation between their predictions and the ground truth. In a follow-up experiment [24], they asked 10 experts to rank 24 cases according to predictors. The inter-rater agreement on the 24 scenarios was high. Then they evaluated the Bayesian model using a round robin procedure, leaving out the 24 cases from one rater for testing while the 24 cases from each of the other nine raters were used in the Genie's expectation maximization algorithm in order to learn the probabilities in the conditional probability tables in the network. The performance of the Bayesian model was $R^2 = 0.598$ which was similar to that of the Linear Regression method ($R^2 = 0.592$) and the Artificial Neural Network $R^2 = 0.606$. Nevertheless, the Bayesian model has many advantages over the other models including the ability to work with missing values.

While these experiments represent an excellent first step, they leave several questions unanswered. First, how did they determine the values of the 12 signals listed by them for each user? One would imagine that some of these signals would require natural language processing techniques or video processing techniques, but the source of this data is not clearly revealed in their paper. Another is the nature of the study. All experiments used synthetic data with no real human subjects, so it is hard to say how well these numbers will hold up in the real world. Nonetheless, the paper presents very important signals that need to be monitored in order to ensure that potential [22] malicious insiders are quickly identified.

In another psychologically oriented paper, Theoharidou et al. [25] present several different theories from criminology and related social science fields on the behaviors of insiders. In particular, they present:

- *General Deterrence Theory.* This theory in criminology suggests that people make decisions on the value of the perceived utility of their actions and the costs involved, not unlike work in game theory. The focus is on deterrence. A study by Straub and Welke [26] suggests that misuse of computers by insiders can be achieved by a Security Action Cycle consisting of 4 steps. In the *Deterrence* step, education and outreach inform all of the penalties for misuse. In the *Prevention* step, the goal is to take measures (such as with appropriate authentication mechanisms) to prevent computer misuse. In the *Detection* step, sophisticated algorithms are used to detect abuse while it is occurring (such as the anomaly detection algorithms listed below). In the *Remedies* step, actions are taken against offenders.
- *Social Bond Theory.* This theory suggests that the likelihood that someone will engage in criminal misuse of computational facilities depends upon his social bonds (i.e. with whom he associates). The logic is that a person is more likely to be a criminal if he associates with a lot of criminals [27]. The authors state that capturing social bonds can be done by monitoring a person's interest in his environment (e.g. colleagues, work projects), his commitment to achieving social status (e.g. that of being a valued colleague) and his involvement in normal activities in which most people are engaged (such as spending time with his family and kids, playing sports).
- *Social Learning Theory.* This theory, similar to that above, says that a person is more likely to commit crimes if he associates with those who do so [28], [27], [29].
- *Theory of Planned Behavior.* This theory [30] distinguishes between intentions (which occur first) and execution. It says that a person's intentions are shaped by his view of how a given behavior will be viewed by others, subjective norms which are social factors that may support or inhibit a user from behaving in a certain way, and control factors which look at whether the person believes that he can control how his behavior will occur and whether he can realize his objectives. In the execution phase, the user waits for an opportunity to act upon his intentions.
- *Situational Crime Prevention.* This theory states that crimes (cyber crimes or others) occur when a person has both motive and opportunity — so by either removing motive or by denying a malicious user an opportunity, we can help prevent crime [31].

In the case of insider threats within organizations, much of this work is applicable. For instance, con-

sider the case of Shakuntala Singla, a former US Coast Guard employee who in 1997, masqueraded as another user and crashed several Coast Guard computers[pages 182-183][23]. She later said she did not expect her actions to have the degree of impact that they had. Nonetheless, many of these theories may be difficult to realize in certain insider threat scenarios. For instance, CIA spy Aldrich Ames and FBI spy Robert Hansen would probably not have been caught by any of the first three theories above. The situational crime theory is a good theory but its realization in practice in a cyber-security malicious insider threat setting poses a challenge — denying attackers the opportunity they seek has been the holy grail of computer security in general, but has been impossible to achieve.

A related highly technical approach of Martinez-Moyano et al. [32] to insider threat detection classifies all actors in an organization into three categories — information workers, security officers, and malicious insiders. The goal of this work is to identify malicious transactions, not users. The authors define a "base" rate to be the percentage of employees who are malicious insiders and decision thresholds that decide the level of suspicion in a user's behavior that triggers an alert. The authors use Signal Detection Theory or SDT [33], [34] from psychology to learn curves in which the x-axis represents the level of suspiciousness of a threat and the y axis represents a probability that a random person will be a true threat, given that level of suspicion. The authors argue that this is a bell curve. They then plot a similar curve where the y-axis represents the probability that a random person is not a threat, given that the suspicion score says he is not a threat. This is also asserted to be a bell-shaped curve. The distance between the means of these two normal distributions is then a measure of how accurate the classifier is.

Based on this idea, Martinez-Moyano et al. [32] set up the problem as a standard feedback control system in which the system is assumed to be in a state $x(t)$ at all times t. This state is a vector. In addition, there are some extraneous events $u(t)$ occurring at time t (e.g. noise in the system). If $x(t_0)$ denotes the initial state vector, then they capture the derivative $\frac{dx}{dt}$ as some function of $x(t), u(t)$ and $x(t_0)$. Given a computational event $e = (e_1, \ldots, e_n)$ where the e_i's are attributes of the event, they assume that the information workers are the ones who will "turn in" suspicious transactions. To model this, they associate a weight w_i with each attribute, measuring the competence of the workers in asessing those arguments. The authors then state that they assume information workers will flag the transaction e as suspicious if the weighted sum $\Sigma_{i=1}^{n} w_i e_i$ exceeds some threshold. In this case, an audit may be conducted by the organization. The paper goes on to add more sophisticated estimates such as the attention span of information workers and their ability to remember things to the model. Based on this, they state, like the work described earlier in General Deterrence Theory above, that information workers (and security officers) try to maximize utility while minimizing cost. In this case, the costs are those associated with false positives and the utility is captured in terms of true positives.

Martinez-Moyano et al. [32] report the results of detailed simulation experiments in which they assume that malicious insiders first launch probes to carry out attacks, consistent with the behaviors observed in a prior study conducted by the US Secret Service and CERT [1]. These simulations look at three scenarios — perfect information where the malicious user's knowledge of the system is very good, alignment training where the organization improves its security training (to all people in the organization) — with this, in their simulations, malicious insiders never went into attack mode. In the third policy they tested, called consistency training, they assumed that information workers would respond consistently to security cues (perhaps through additional training). Here, as expected, the attack rate went down.

These simulations shed important light on the types of security policies that may apply to humans in an organization but not to the network or hosts themselves. However, from the perspective of using this model to deploy an "online" malicious insider threat detection algorithm, the work has some challenges to overcome. First, it is unclear how to get good quantitative judgments of the level of saving (the weights w_i) that users in the organization have in detecting cyber-threats. Second, in many real-world malicious classified environments, only a small number of people can see what a given user is doing because of "compartmentalization" restrictions. Third, it is hard to tell whether the linear threshold model ($\Sigma_{i=1}^{n} w_i e_i$ exceeds some threshold) used to flag someone as a malicious insider actually works. What should the threshold be? What are the false positive and false negative rates? Nonetheless, this work offers important value in understanding how external policies within an organization can lead to better cyber-health for the organization.

In a recent paper [35], Martinez-Moyano et al. expand their previous work on the behavioral aspects of insider threat identification focusing on the learning process of security officers and decision-makers. They explicitly say that the proposed model is not thought of as a substitute for multiple layers of automated security mechanisms. Their goal is to provide additional considerations and processes whose identification by security officers and decision-makers may be important to mitigating the risk of the defense mechanisms being compromised as a result of novel and not-yet-fully-known threats and attacks. The proposed learning model was evaluated by 12 experienced decision scientists who played the role of security officers

over a two-week period facing synthetic data. The results showed that the model accurately captures the variability of the human-generated results without bias. However, further work is necessary to use the proposed model for training security officers in order to improve their performance in identifying malicious insiders.

Probst and Hunker [36] present a thought-provoking discussion on the difficulty of estimating the risk of insider threats. They state that the expected loss function from the deployment of a counter-insider threat policy is the probability of an insider threat to occur, times the probable damage of a certain amount or type, times the probability that the policies imposed will be unsuccessful in blocking that threat. Policy and security officials can imagine all sorts of threats, but estimating the relevant probabilities is extremely difficult. Attempting to counter each possible threat (assuming they will occur with probability 1) is very costly. Furthermore, they believe that for more complex trust relationship-based insider threats, the policies deployed to counter the threat may not be effective (i.e., the probability the probability that the policies imposed will be unsuccessful in blocking that threat is relatively high). Based on their observations, they conclude that it is important to understand how better to change the motivations of insiders, instead of merely focusing on controlling and monitoring their behavior.

2.3 Anomaly Detection Approaches

A number of approaches to insider threat detection try to build a model of normal behavior and then attempt to detect anomalies. These include a broad overview of anomaly detection methods for external threats (often termed intrusion detection) by Patcha and Park [37]. Detection of external threats is different to insider threats, as the threat is easier to detect. For example, by looking for infiltrations to the system, or finding other unusual behavior as we would expect an external intruder to behave differently from insiders. Insider threat poses more of a challenge, since the malicious user has access to internal systems, can spread his malicious actions among legitimate actions etc. However some of the methods used for external threats can be used or adapted to insider threats, and are therefore mentioned here. Anomaly detection is based on the assumption that activity deviating from normal activity will raise an alarm. Anomaly detection is more beneficial for detecting insider attacks than previously proposed signature systems assuming the insider is a masquerader, or has deviated from his regular behavior. Anomaly detection techniques include: statistical anomaly detection, machine learning based anomaly detection and data mining methods. One drawback of anomaly detection is that the system must define 'normal' activity; this is a challenging task. Although hybrid systems that incorporate anomaly detection with signature detection would seem to produce a much stronger detection system, the resulting hybrid systems are not always better. Patcha and Park [37] mainly described methods that focus on the external threat problem in thier survey. Insider threats are described by Patcha and Park as a great challenge not yet solved. In this paper, we focus on two of the anomaly detection approaches so as to give a flavor of how these approaches work — an outlier detection based approach, and a classification based approach.

2.3.1 Insider Threat Detection via Supervised Outlier Detection

In this section, we discuss the work of Liu et al. [38] who operate at the operating systems level by adapting methods used to detect external threat to the case of insider threat. They track user activities at the operating systems level for two reasons. First, by monitoring actions at the OS level, they ensure that *all* activity of any user is tracked, not just user activity detected by higher level entities such as application logs. Second, by tightly integrating insider threat detection with the operating system, there is less chance that the insider threat detection engine itself gets compromised by a smart adversary. They then proceed as follows. They define a set of system level "features". They define three sets of features using n-grams, histograms, and parameter-based approaches. Once the features are defined, they use a training set of known "normal" events in conjunction with a k nearest neighbor algorithm to detect outliers. We will describe their approach in greater detail below.

First, the basic types of events captured by Liu et al. [38] involve system calls generated by the following types of higher level activities: browse, db_admin, email, misc, open_office, software_dev. In addition, the authors identify 7 types of exploits. These exploits include:

1) privilege_escalation in which the user tries to gain root access using any available privileges and local applications;
2) removable_media in which the user tries to copy data and files to a removable device such as a hard drive, USB stick or CD;
3) export_via_email in which the user tries to email data;
4) change_file_extension in which the user tries to change file extensions in order to trick any network monitoring code;
5) encipher, decipher operations in which the user tries to computationally modify a secret document;
6) unusual search in which the user looks for documents he is not authorized to access;
7) malware_installation in which the user tries to install malware on the system.

The features defined by Liu et al. [38] involve three types of high level features: n-grams, histograms and a parametrized system call information.

2.3.1.1 n-grams: An n-gram is merely a sequence of n system calls. For example, a 4-gram is a sequence of system calls of length 4. An *n-gram record* is any single n-gram. The authors generate n-grams via a sliding window - for instance, in a system call log of calls $c_1, \ldots, c_r$, the first n-gram consists of $c_1, \ldots, c_n$, the next n-gram consists of $c_2, \ldots, c_{n+1}$, the next consists of $c_2, \ldots, c_{n+2}$, and so forth. The authors leverage the use of n-grams from prior work of Hofmeyr et al. [39] on identifying external security threats, and suggest using $n = 5$ based on their experience and tests.

2.3.1.2 Histograms: Leveraging prior work of Liao and Vemuri [40], for the external threat problem, the authors first create a "window size" (e.g. 30 calls is suggested). For each window w_i, they count the number of each system call type executed within that window. The authors then overlap windows (they suggest an overlap of 5). Thus, suppose once again that our system call log sequence is $c_1, \ldots, c_r$. In this case, the first window w_1 consists of the calls $c_1, \ldots, c_{30}$. Because w_2 must overlap with the last 5 calls in w_1, w_2 consists of $c_{26}, \ldots, c_{55}$. w_3 then consists of $c_{51}, \ldots, c_{80}$, and so forth.

2.3.1.3 Parameter-based system calls: The previous two paragraphs merely track the "type" of system calls (e.g ."open" a file) but do not keep track of the parameters invoked by the system call (e.g. the file name). In this class of features, the authors keep track of the exact types of parameters used. They then keep track of the number of system calls (with associated parameters) of each type. They provide an example where the system call is "open" with parameters

(java,30,homeuser1aaaasensB.class,438,32768)

corresponding to the Process Name, the return code, the full path name, the file create mode, and the file mode.

In the next phase, Liu et al. use these features in order to identify anomalies. They do this by first using a training set TS consisting solely of normal (non-malicious) records of user activity (an n-gram and a window histogram corresponding to a user record).

For the validation set VS, they find the distance between a record $r \in VS$ and the records in TS. The authors choose the neighbors which are at the k^{th} greatest distance from the records in the training set. In the case of the n-gram approach, they use Hamming distance to measure the distance between two n-grams. In the case of the histogram-based representation, the distance between two records $(r_1, \ldots, r_n)$ and $(r'_1, \ldots, r'_n)$ is simply set to $\Sigma_{i=1}^{n} |r_i - r'_i|$. For the case of the parameter-based system call representation, for any two calls, they simply calculate the number of parameters on which the two calls vary (this is the Hamming distance between the two records).

Liu et al. conduct experimental evaluation using synthetic data and present ROC curves showing that the n-gram approach does not do very well. However, the histogram approach does better, while the parameter-based methods perform the best. *Note that this work identifies anomalous records, not anomalous users*. To find anomalous users, one would presumably need to aggregate the levels of anomaly exhibited by the different system calls generated by that user.

Liu et al. extended their work in [38] and tried applying their methods for external threat detection to insider threats [41]. They found that the transition does not work well and suggest that this may be a result of different assumptions made in the different domains. For example external threats are assumed to be outliers when compared to normal activity, whereas insiders may perform some normal activity and some malicious activity and be harder to detect. This comparison further motivates our work by indicating that simply using methods developed for external threats is not sufficient for insider threats, and new methods must be considered.

2.3.2 The ELICIT System

Maloof's ELICIT system focuses on detecting insiders within an organization who violate the "need to know" principle. In most security organizations (such as the FBI), each user has a security clearance level. These may include clearance levels like secret, top_secret as well as top_secret_sci where the sci stands for "Sensitive Compartmented Information". SCI clearances allow a user to see certain "compartments" of information but not other "compartments." The "need to know" principle requires users to only look at information they need to look at, not go snooping. Maloof and Stephens [5] mention the case of an FBI analyst who was arrested for downloading/printing information about the Philippines that he had no "need to know." Recent episodes involving Bradley Manning (who leaked information to Wikileaks) and Edward Snowden (who did the same) also involve their downloading classified information for which they had no "need to know." ELICIT was tested on a real-world corporate intranet during a 284 day period involving 3900 users. A red team generated synthetic insiders on the basis of attacks that had been carried out in the past by real world spies such as Aldrich Ames, Robert Hansen, and other. They conducted experiments with this large data set and ended up with an ROC curve with an AUROC (area under ROC curve) of 0.92 which is amazing.[1] Because

1. A ROC (Receiver Operating Characteristic) curve is a classic way of measuring the accuracy of a classification algorithm — in this case, the algorithm that classifies people as "posing a threat" vs. "not posing a threat." A ROC plots the false positive rate on the x-axis and the true positive rate on the y-axis. The area under the ROC curve (or AUROC) is a measure of the accuracy of a classification algorithm — the maximal possible area is 1, so an AUROC of 0.92 is outstanding.

of the excellent results achieved by ELICIT, we will elaborate on the techniques used therein. ELICIT uses four steps:

1) *Data collection.* In the first step, ELICIT transforms network traffic packet-level data (that the system monitors via sensors placed within a corporate intranet) into what the authors call "information use" (or IU) events. This transformation of raw packets into IU events is done via programs called packet decoders. In addition to the packet level data, ELICIT also contains contextual information (such as where a person's office is, where printers are located, etc.). In addition ELICIT tries to attribute events to actual users on the system.
2) *Anomaly Detectors.* In this critical stage, ELICIT developed 76 "detectors". A detector tries to anticipate one type of activity that an attacker might engage in that is anomalous. These detectors involve three techniques — rules written by experts, as well as parametric and non-parametric density estimators.
3) *Bayesian Ranking.* As each of the 76 detectors can flag 76 different types of anomalous activity, for each user we must amalgamate the alerts raised about his activity and merge them into a single score capturing the likelihood that he is a malicious insider.

We describe these five steps in further detail below.

2.3.2.1 Data collection: An *IU-event* is an atom $a(t_1, \ldots, t_n)$ where a represents an action, and $t_1, \ldots, t_n$ represents the parameters associated with an action. Thus, an IU-event is nothing but an action as represented for many years in AI planning [42]. As usual, different actions can have different numbers of arguments. ELICIT looks at packet level data and extracts 8 types of high level actions. These actions captured by ELICIT are list,delete,read,write,move,print,query,send. Because ELICIT captures arguments, its actions are similar to those captured by Liu et al. [38] except that they do not consider either sequences of actions (n-grams) or histograms. The arguments of each of these actions are drawn from the following "fields": protocol (such as HTTP, SMTP, FTP, etc.), file name/path, start/stop time, client/server IP address, user name, bytes, original file name, printer_id, pages, search phrase and e-mail headers. Different actions have varying subsets of these "fields" as attributes.[2] The conversion of packet level data to these high level IU-events was done using a software package called `Ethereal`.[3]

In addition to IU-events, ELICIT also captures various auxiliary types of information. These include various attributes about each user such as their office, title, the projects they were working on, and so forth. By examining emails, ELICIT was also able to build a social network of users within the organization. By understanding the projects a user was working on, they were able to identify the types of documents the person should be looking at (e.g. a Nigerian analyst should perhaps not be looking at documents about the Philippines unless the documents involved both Nigeria and Philippines). Likewise, a person would be expected to use printers near his office, not ones far away.

Last but not least, the authors wrote "event attribution" code which tied IU-events to specific users. Of the 91M IU events tracked by ELICIT, the authors note that 14.7% were directly attributed to a user (i.e. the user authenticated himself prior to the event), 2.3% were indirectly attributed (e.g. the user's email was included), while the remaining events were not attributed. In order to attribute the 83% of unattributed events, the authors used two techniques.

Suppose $e_1 : t1, e_2 : t_2, \ldots, e_n : t_n$ represents a sequence of events e_i occurring at time t_i. We assume $t_1 < t_2 < \cdots < t_n$, i.e. the events are listed in ascending chronological order. Note that some of the e_i's may be attributed to a user, while others may not. Let $user(e_i)$ be the ID of a specific user when e_i is attributed and $\perp$ (or unknown) otherwise. Suppose now that $user(e_i) = \perp$ and we want to infer the correct user associated with e_i. The authors propose two methods, one of which is discussed below.

In the first event attribution method, let $L = \max\{j \mid j \leq i \wedge user(e_j) \neq \perp\}$ and $U = \min\{h \mid h \leq i \wedge user(e_h) \neq \perp\}$. Thus, e_L is the nearest attributed event prior to the event e_i we are trying to attribute, while e_U is the nearest attributed event after the event e_i we are trying to attribute. We set $Best = L$ if $|t_i - t_L| \leq |t_i - t_U|$ and to U otherwise. In other words, $Best$ is the attributed event (of e_L, e_U) that is temporally closer to the event we are trying to attribute. The event e_i is then attributed to e_{Best}.

2.3.2.2 Anomaly Detectors: ELICIT contains 76 detectors that each track one type of activity. Not all activities mentioned above are used — the authors focus on browsing, searching, downloading, and printing activities. Given a window w of time, ELICIT tries to capture anomalies during that window of time.

Like Liao and Vemuri [40], ELICIT associates a histogram in order to capture the activity of individual users. In addition, they develop histograms to capture the activities of users having the same job title. Given a window w of time, a user u and a property p, let $\mu(u, p, w)$ be a measure of how much user u engaged in property p during time window w. Likewise, given a group G of users, let $\mu(G, p, w)$ denote the same

2. The authors note [5, p.149] that "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."

3. www.ethereal.com

quantity, applied to a group of users. The 76 detectors varied these properties, but they included things such as the number of "sensitive" search terms (e.g. "proprietary"), the number of pages and/or documents printed to a printer that is not located near the person's office, the number of documents downloaded, the number of search queries issued, and so forth. In addition, ELICIT also used a social network of users within the organization (built from email traffic) to identify the percentage of documents that a user retrieved that were retrieved by people outside his immediate neighborhood (i.e. people who were two or more links away from him in the organizational social network). The rationale here is that the user might be expected to work on things similar to those that people he is directly connected to work on, but is less likely to have work-related interests in documents not authored by people near him in the network. For all of these types of "detectors", ELICIT typically used experts to define anomaly thresholds. When a user's activity during a given time window exceeded the anomaly threshold, then an alert would be generated.

In short, at the end of the anomaly detection phase, each user u has an associated set of (upto 76) alerts generated for the most recent time window w.

2.3.2.3 Bayesian ranking: The set of generated alerts is then shipped to a Bayesian ranking engine that tries to assign a "threat score" to each user. This is done by a 3-layer Bayesian network. The root is the Malicious_Insider node. The next two levels have nodes corresponding to each of the 76 properties detected by the detectors. Given a node N coresponding to a detection property p_N, node N intuitively captures two conditional probabilities: $\mathbb{P}$(u is malicious|u generated an alert for p) and $\mathbb{P}$(u is malicious|u did not generate an alert for p). For each node at the third (leaf) level, we have two probabilities again. $\mathbb{P}$(detector will generate an alert for p| p occurs) and $\mathbb{P}$(detector will generate an alert for p| p did not occur). The probabilities for the top two levels were generated by experts, while the probabilities for the last level seem to have been captured using historical data.

In order to assess the score of a single user during a given time window, ELICIT first sets to 1, all nodes at the leaf level (3rd level) for which the user generated an alert. It then propagates probabilities up the Bayesian network till a probability is assigned to the root node, Malicious_Insider, specifying the probability that the insider is malicious.

Based on these steps, the authors conducted experiments using one month's network traffic data. A "red team" generated 8 evaluation tests, each corresponding to a real world insider scenario that had occurred in the past (e.g. Aldrich Ames, Robert Hansen, and others). The authors report that their false positive rate was 1.5% which is outstanding, and that the area under the ROC curve was 0.92. These are outstanding results. However, the study suffers from a few problems.

1) First, a number of "past malicious insiders" were studied by both the ELICIT team and the Red Team, so the training and validation data sets effectively overlapped which is somewhat strange.
2) As the red team only used the behaviors of past malicious insiders to inject threats, ELICIT shows that detecting known *past* behavior can be done efficiently and with a lot of work. This explains the high AUROC of 0.92 obtained by the ELICIT team. It is silent on the topic of detecting *unknown* new behaviors. For instance, a spy working for a foreign state may well use sophisticated new techniques disclosed to him by his control operators that are designed to be smarter than those who were caught in the past (clearly people who were caught in the past were not smart enough not to get caught!). The BAIT system proposed in this paper helps address this.
3) Third, by requiring that all past anomalous behaviors be encoded in some form or fashion into the detectors (which are often hand-coded), it is hard to say how ELICIT will translate to different settings.

Nonetheless, ELICIT represents a good effort to identify malicious insiders. The 76 detectors they built track different types of known malicious behavior and must form part of any reasonable insider threat detection system.

To answer these problems, in 2007 the ELICIT team conducted a fascinating study of more than 50 participants, holding various positions at MITRE including management, technical, and administrative staff [6]. The participants were randomly assigned to one of two conditions: benign user (control group) or malicious user (experimental group). Both roles describe a person who has fallen on the same hard financial times and must find and deliver the most valuable information in order to improve his or her financial situation. In the benign condition, the person was given an opportunity to participate in a high-profile team and had it explained to him that outstanding performance would likely lead to a promotion and a pay increase. In the malicious condition, the person was given an opportunity to start a new, higher-paying job, but the offer was conditional on bringing inside information from the old company, thereby providing a competitive advantage in a major federal acquisition.

They monitored the participants' activities. Their preliminary analysis revealed interesting and significant patterns in malicious behavior. Some of the patterns they found for distinguishing between benign and malicious behavior are related to the actions

we enabled in BAIT such as fetching documents, document editing and saving documents to CD. They do not use any Machine Learning in their approach. ELICIT's list of detectors is very valuable and must be deployed — but it is unclear how it will apply to threats never seen before which is the focus of our work. Nonetheless, ELICIT's findings support our approach in BAIT that role playing is a necessary methodology to tackle the insider threat challenge and that given such data, machine learning techniques should be used to build malicious insider predictors.

2.3.3 Other Relevant Anomaly Detection Work

ADAM [43] uses frequent sequence mining for external threats. The model is built in two stages, first ADAM uses attack-free feature vectors to find frequent sequences associated with normal behavior. In the next stage, unlabeled data is mined for frequent sequences that are compared with those found in the first stage. Frequent sequences that occur in the unlabeled data that are not frequent in the first set are tagged as suspicious. Another similar external threat detection algorithm is proposed by Denning [44] where the attack-free feature vectors are profiled, and malicious insiders are viewed as anomalies that do not fit the model.

Lee et al. [45] provide a detailed discussion on the unique characteristics of external threat detection such as: how to measure accuracy in external threat detection, or what features to extract. The data they use have no examples of anomalies (i.e. malicious insiders). The authors overcome the absence of anomaly data by presenting a technique for artificial anomaly generation.

Related efforts include the use of stochastic timed automata to recognize unexplained (though not necessarily malicious) activity in video [46] — this model was subsequently adapted to identifying unexplained sequences of events in network traffic as well [47], [48].

Shavlik and Shavlik [49] proposed a machine learning algorithm that uses Winnow [50] to create an anomaly detection system. This system creates statistical profiles of normal usage for a given computer running Windows. Deviations from the normal pattern may be indicators for external threatening behavior. Examples of features are network activity, file access, CPU load and many more. Detection rates of nearly 95% were found for the data set of 16 subjects collected. A similar study of Richardson et al. [51] used keyboard and mouse activity to profile users, and attempted to differentiate between a legitimate user and a masquerading user.

Ray and Poolsapassit introduce an interesting approach to insider threat detection [52]. They present a method to amplify existing external threat detection methods with a system that targets suspicious insiders. They assume that it is possible to enumerate the attacks that can be launched against a system, and use this to derive a list of all actions that can be performed against the system. Users are asked to submit a description of their intended usage of the system, and deviations from this plan, that are considered attacks are reported and raise alarms.

Senator et al. [9] presented new approaches to detect insider threats. The system uses structural and semantic information in a variety of anomaly detection algorithms. They suggest a large variety of methods including using behavior of suspected insiders, indicator of unusual activities, statistical patterns or normal and suspicious behavior etc. For example, one of the studies presented in [9] and expanded in [53] describes how using domain knowledge creates a good starting point to select appropriate features for use in anomaly detection, which are associated with insider threats. In [53] Senator et al. also presented a visual language to describe anomalies relevant to insider threats. Nance and Marty [54] presented methods of classifying and visualizing insider behavior to establish a pattern of acceptable actions based on workgroup role classifications. The proposed methods are demonstrated via simplified examples showing how visualization can be used to help detect insider threat. The work of Senator et al. [9] demonstrates the feasibility of detecting insider threats. The authors describe a very impressive number of results using various methods for analyzing the data from the ADAMS project[4]. however, they do not suggest a specific method as being superior to others and state that additional research is necessary in order to enable the real usage of these methods. Moreover, in their work, the attacks are synthetically generated and not real, even though these fake attacks are embedded in real transaction data.

A special type of malicious insider is a masquerader - a person who uses somebody elses computer account. Masqueraders are mostly insiders. Schonlau et al. [55] studied whether statistical methods can find anomalies in the behavior of masqueraders. Being aware of the difficulty of finding data to detect masqueraders, they developed a detailed methodology for simulating masqueraders. They collected UNIX command data from 70 users: 50 users served as intrusion targets and the remaining 20 users served as the masqueraders and they interspersed their data into the data of the 50 users. Using this data they showed that it is possible to detect masquerades by formulating hypotheses and applying statistical theory to test them.

2.4 Honeypots

There has been a tremendous amount of work done on the use of honeytokens [56], honeypots [57], and honeynets [58]. Honeypots have been used to identify

4. en.wikipedia.org/wiki/Anomaly_Detection_at_Multiple_Scales

spammers [59], build collections of malicious code [60], identify credit card fraud and identity thieves [61], to name a few applications. All of these methods are designed to lure a person with malicious intent towards an apparently valuable piece of information with the goal of identifying who he is and what he wants. People identified using honeypots gain no secret information (at least through the honeypot) and can be further investigated by other means such as traditional law enforcement mechanisms (in addition to ongoing online surveillance of their activities). Informally speaking:

1) A *honeytoken* is a computational entity such as a database record, a login/password combination, a name/social security number combination, a file such as a corporate spreadsheet or a corporate document designated to lure a malicious user to access it.
2) A *honeynet* is an entire network of computers whose sole purpose is to lure potentially malicious users inside a fishbowl where their activities can be comprehensively monitored, their identity established, and where it is possible to establish their motivations and intent.
3) A *honeypot* is a decoy (e.g. a file or a server) that is indistinguishable from other similar objects (e.g. real files or real operational servers) intended to trick a malicious user into thinking it has significant utility for him (e.g. as a launching pad for accessing additional information or for launching spam).

Of course, all the "honey" type objects mentioned above contain spurious content which aim to cause a malicious user to reveal himself. We will use the word "honeypot" to collectively refer to all of these entities in the rest of this section. Though there are many excellent papers on honeypots, we will focus only on a few that capture the main features of honeypots.

In a very practical paper, Spitzner [62] suggests that various honeypot methods may not succeed against users who are merely sniffing the network. In order to address this, he suggests that honeytokens be embedded in network traffic. As a consequence, instead of waiting for a malicious insider to "come to the honeypot", the "honeytokens" fly around the network. A malicious user sniffing the network, for example for userid-password combinations, will likely sniff out the honeytoken. Once he attempts to use it, the system, knowing that this is a honeytoken, will be in a position to monitor the user's actions. Moreover, it allows the organization's security managers to identify where in the network the honeytoken was combed.

A second example provided by Spitzner [57] suggests that honeytokens can be inserted as spurious email messages in the mail boxes of potential targets within an organization. Suppose, for instance, a secure organization like a large defense contractor, believes that a person X is a high value target for a foreign state because he works on sophisticated missile programs. In this case, Spitzner [57] suggests inserting into X's mail box, a spurious email that contains a honey token that purportedly gives access to some high value data on the missile program (or some other high value data). If X's email is compromised and if the external actor chooses to utilize the honey token, then the organization's system managers immediately know about the intrusion due to the attempt to access the honeypot. Similar methods can also be utilized to monitor web searches and the like.

The fact that a person encounters a honey token may not suggest any malicious intent. For instance, if person Y were to search his corporate intranet for information on Iranian missile programs and one of the returned web pages includes a honey token, then the fact that he is shown the honey token does not imply malintent. Even accessing the honey token may not denote malintent — it could simply be a case of curiosity. But curiosity killed the cat and in this case, it would probably trigger a deeper investigation into the person's motivations.

Bowen et al. in their excellent paper [63] provide methods to generate and deploy decoy documents so that they closely resemble normal documents, making it hard for a malicious user to distinguish between real and decoy documents. They developed different types of decoys as listed below.

1) *Hash Method Authentication Codes.* One class of decoys include a keyed "hash method authentication code" or HMAC in it. The idea is that these codes can be inserted into documents. If all documents within an enterprise have such HMACs embedded in them, only a user who has the key can identify the decoy documents from the real documents. The authors developed methods of embedding such HMACs into PDF documents so that the insertion is not visible to people viewing the documents. There are a couple of flaws with HMACS. The first is that all documents (including all legacy documents) within an organization must have embedded HMACS in them — this can be a tall order for an organization. The second is that an enormous amount of the code within an organization may need to be re-engineered in order to account for the HMACs. For instance, an analyst working on Nigeria would need to receive only real (not decoy) documents.
2) *Trap-based Hosts.* These decoys include honeytokens such as login and password information of fake Gmail accounts. Users who access such trap-based hosts can be easily monitored — and for instance, if they try to illegally use the fake information, then they are subject to additional appropriate law enforcement actions.
3) *Beacon Decoys.* These are malware effectively

embedded in documents that a person should not be downloading. An organization might, for instance, place an Excel spreadsheet somewhere with names and social security numbers in it (all falsified) as a decoy. Beacon decoys have pieces of code embedded in documents so that when opened (after the document is downloaded), they reveal a variety of information to the organization that placed the beacon in the document. For instance, suppose a person downloads the above Excel file from his enterprise onto a mobile phone. He is most likely a malicious user — otherwise why would someone want to download names and social security numbers? When he opens the document, the document may contact a remote server controlled by the organization and send all kinds of information to that organization. This may include a GPS location, the IP and MAC addresses of the phone, information on the phone SIM card, a video stream using the phone's camera, and so forth. Bowen et al. [63] specified how they built beacon decoys in Word documents by including remote images (i.e. images that must be retrieved from a remote site). When the document is opened, the attempted remote retrieval is a signal. They also built beacon decoys in PDF documents.

In order to identify masquerading users (malicious users masquerading as legitimate users), Bowen et al. [63] build a behavioral model of a user's search actions. As in the case of Liao and Vemuri [40] and Maloof and Stephens [5], the authors first built a "baseline" model of each user. To do this, they tracked a number of system related activities such as activities related to the Windows registry, actions accessing various dynamic linked libraries (DLLs), creation of processes, kills of processes, and so forth. Such activities can be considered independent variables. For each user, they gathered the values of the independent variables over 10 second windows for many different 10 second windows. They then used a one-class class support vector machine (OcSVM) such as the one presented by Schlkopf et al. [64]. Suppose the behavior of user u during time window w_i is captured by a vector v_i where the vector v_i contains one value for each independent variable. Let $W = \{w_1, \ldots, w_r\}$ be the set of all time windows. The OcSVM tries to find a hyperplane h in this n-dimensional space where n is the length of the vector such that the probability that a randomly chosen point within the n-dimensional space is in W is less than or equal to some user specified threshold probability p. In addition, the number of errors must be bounded. Finding such a hyperplane can be posed in a straightforward way as a quadratic programming problem. Bowen et al. [63] used ocSVMs, which use features such as those listed above, to build a model of normal behavior of a user u. They then used the ocSVM to identify users whose current activity (e.g. in a time window w_{r+j} for $j > 0$ does not fall on the same side of the hyperplane h as the training data. According to their paper, they were able to use this method to detect all masquerading activity with 100% accuracy with a false positive rate of 0.1% which are very impressive results.

Kandias et al. [65] present an interesting model that integrates several of the techniques we have discussed, combining technical solutions with approaches that draw upon psychology. They use a user taxonomy, psychological profiling, real time usage data, honeypots and a decision algorithm in order to identify potentially dangerous users. Unfortunately, no experiments were performed to test and evaluate their interesting model.

2.5 Graph-Based Approaches

A number of efforts also look at graph-based models.

Chinchani et al. [66] introduced the concept of a *Key Challenge Graph* (KCG), extending the well known notion of an attack graph [67]. The vertices in a KCG corresponds to some physical entity such as a host computer or a server. An edge is drawn from entity v_1 to entity v_2 if there is a communication channel from v_1 to v_2. The framework Chinchani et al. presented in their work [66] allows multiple edges to exist from entity v_1 to entity v_2. A *key* is a computational object present on a vertex — there is a mapping specifying what resources are available at which vertices. Keys could include passwords, data, documents, programs, and other objects. An edge may have zero or associated *key challenges*, specifying a hurdle that must be cleared by the user trying to traverse an edge from v_1 to entity v_2 in order to access the keys present on v_2 — for example a key challenge could ask for a userid and password combination or may ask the user a question (e.g. "What is your mother's maiden name?") and wait for a correct answer. In addition, we assume the KCG specifies some set of vertices that are starting points of attacks and some set of vertices representing possible targets. The goal of the malicious insider is to reach all of the targets.

A successful attack on a target is basically a sequence of vertices that the insider can traverse, starting from vertices that he initially has access to, which eventually include all the vertices in the target set. Of course, only edges in the graph where the user can address the key challenge successfully can be used to build this sequence. The authors defined a cost for traversing edges. They show that the problem of checking whether a key sequence with cost below some threshold exists is NP-complete. They then provided algorithms to compute successful near minimal cost attacks and showed that their heuristic algorithm runs in a reasonable amount of time on very small graphs (upto 300 vertex graphs) – and they reported

that on such graphs, computing time can take 15-20 minutes.

Though Chinchani et al. [66] provided an elegant notion of key challenge graphs, their work has two major shortcomings. First, there is no empirical evidence showing that they were able to really catch insiders as the framework was not tested on real data. Second, the run times are prohibitive. In real world enterprises, the graphs are likely to have millions of vertices, leading to unacceptable run times.

Eberle et al. [68] proposed a method for detecting insider threats by developing graph-based anomaly detection algorithms. They proposed GBAD, a graph-based anomaly detector. They studied anomalous insertions, modifications and deletions w.r.t. a graph. Given a body of transactional data (e.g. IP logs), they followed three steps:

1) Discovering network activity that is unexpected. This can be achieved by standard anomaly detection methods.
2) Create graphs focused solely on the times when the anomalies occurred and the relevant users. Within a given time frame, they can eliminate vertices in various ways, e.g. people whose machines were compromised.
3) Predict who is behaving in an anomalous manner. For this, they built a graph containing "movement" data for each employee identified in the preceding step (only for the days when anomalous activity was reported). The nodes in the movement graph involve location nodes (e.g. outside, building, classified_space, network, etc.) and an edge is drawn from one node to another if the user did something that involved moving (either him or data) from the first location to the second. For instance, we may have an edge from "network" to "printer4" with the label "send" or "number of docs" to indicate a print job by the user.

   Using a set of training data (graphs), the GBAD tool identified what they call "normative" patterns. These appear to be closely related to frequent subgraphs or common subgraphs in the training data. They then measured the distance between a particular user's "movement" graph and the "normative" pattern. They looked for the top-k most distant users and flagged them as the most anomalous. They test their method on data from the VAST (Visual Analytics Science and Technology) insider threat data which consistsed of 60 people. Using an ensemble of three different methods, they were able to narrow this number down to 2. On a synthetic graph with 1M vertices, their algorithm took about 3 days to run.

   Unlike Chinchani et al. [66], the work reported by Eberle, Graves and Holder [68] was tested on realistic sized synthetic data for run time. Three days may seem like a lot to computer scientists, but for security organizations seeking to find malicious insiders, having a computer run for 3 days is perfectly acceptable if it can generate good results. The accuracy tests they performed, however, need more work. The VAST data set, contrary to its name, is not vast, but tiny with only 60 users in it — and the data is all synthetic.

Other graph based methods use stream mining to detect insider threats, with both unsupervised [69] and supervised [70] methods. The unsupervised approach used by Parveen et al. [69] adapts GBAD to work with unbounded streams that have evolving patterns (with an accuracy of 56%, false positive rate of 54%, and false negative rate of 42%). The supervised approach [70], which implements a one class SVM that handles unbounded streams (with an accuracy of 71%, false positive rate of 31% and false negative rate of 0%) was found to be superior to the unsupervised approach.

2.6 Game Theory Approaches

Game theory is widely used in modeling security problems [71], [72], [73], [74]. Liu et al. [75] proposed the insider game -- a two-player zero-sum stochastic game to model the interaction between the insider and the system administrator. Their goal was that the game would facilitate the understanding of malicious insiders' motives and their decision-making process. Also, the insider game helped infer the strategy that the insider would take and determined the defender's best counter-move. They demonstrated the applicability of the game by using it to model and analyze a real-life incident where the administrator cannot recognize some states [76]. While the accurate prediction of an insider's moves was the main motivation, only Nash equilibrium analysis has been used to capture the insider's future actions, in order to respond properly. It is well known that people do not follow Nash equilibrium strategies [77], [78] including intelligent security people [79]. In addition, the game is a one-shot game, and a discussion on repeated games was not included.

To address these two shortcomings, an extensive form game played repeatedly between the malicious insider and the Intrusion Detection System (IDS) of the organization was provided by Kantzavelou and Katsikas [80]. They deployed the logit Quantal Response Equilibrium (QRE) to capture the players' bounded rationality and to model the insiders' behavior following other security systems that deploy game theory methods [79], [81]. They used the QRE results to predict insiders' future behavior. Using these predictions, they suggested reactions to the IDS against this behavior in order to protect the system. To demonstrate the applicability of their approach, they implemented an IDS. However, no experimental

evaluation was conducted. Some of the concerns with respect to this approach include: (i) the time complexity of the QRE calculations and (ii) whether the assumptions made when defining the game are realistic. In particular whether the defined set of possible actions of the game is large enough and the whether the system will be able to identify them.

Tang et al. [82] extended Kantzavelou and Katsikas' model [80] in order to handle shortcomings of traditional IDSs. They proposed to use a dynamic Bayesian network (DBN) structure [83] to infer the insider's actions before calculating the QRE equilibrium. They compared their approach to Kantzavelou and Katsikas' model by means of simulation experiments. They showed that while their computation cost was higher, they obtained improved convergence and precision.

Zhang et al. [84] discussed an interesting idea. They observed that an attacker's strategy depends not only on his own goals but also on the toughness of the defender (i.e., the defender's willingness to enforce security, even at the expense of false positives). They studied how to establish the defender's toughness reputation in anomaly detection against insider attacks. They considered both naive attackers who attack regardless of the defender's reputation and actions, and smart insiders who learn from the outcomes of past attacks in order to avoid detection. Based on a game-theoretic formulation, they proposed two generic reputation-establishment algorithms for systems consisting of only smart insiders and also those with both smart insiders and nave attackers. They performed experiments using the syntactic data of [38]. The experimental results and the theoretical analysis indicate that their proposed algorithms can significantly improve the tradeoff between the detection rate and the false positive rate.

3 THE BAIT FRAMEWORK

In this section, we present BAIT (Behavioral Analysis of Insider Threat) — a framework developed by the authors to use behavioral cues to identify insider threat. In particular, BAIT represents one of the first studies of real-world users attempting to compromise a system — a smaller scale study was reported by Caputo, Maloof and Stephens [6]. BAIT contains bootstrapped algorithms that try to learn separators between malicious insiders and honest users for an environment with the following properties:

- Honest and malicious insiders' data is gathered from real human players. We took a number of steps (detailed later) to ensure that only high-quality subjects dedicated to their mission were considered in our study.
- The attacks are carried out by real humans similar to employees in an organization without regard to past attacks reported in the literature. In particular, our results are not biased by the possibility of both attackers and defenders studying past cases of insider attacks (which has biased many previous studies).
- Malicious insiders are directed to hide their actions.
- There is almost no labeled training data and the training data set is highly imbalanced.
- The system does not have any data on users' past behavior.
- The system may only observe the players' actions on a high-level and does not have any access to specific features of the documents which the players are handling or should handle.

Table 1 shows that BAIT is different from the related work described earlier (i.e., the papers we described earlier) because it was explicitly designed to ask real users to carry out attacks without exposing these real users to data about past insider threat attacks. BAIT carried out real experiments with a 35 round game in which 795 users were recruited on Amazon Mechanical Turk.

3.1 The BAIT Game

The BAIT game is deployed on Amazon Mechanical Turk and helps us learn behaviors of real world users who both act normally and who are charged with exfiltrating data from an organization. Players were told that they have to select tasks that they would perform in a high security facility on various topics. They do not actually perform these actions, but rather select them from a list of predefined possible actions. The players were told that they are required to gather information on several assigned topics (e.g. an analyst might be tasked to collect information on piracy incidents off the Horn of Africa), edit this information and send it (or hand it) to the topic's requester.

The game is a single player game (i.e., an "open" game) and is composed of 35 rounds. The player may either be an *honest* worker or a *malicious insider*. Both types of workers receive a list of between 3 to 5 classified[5] topics. The players are told to gather information on each of these topics, edit this information and send it to the topic's requester. The players are told to assume that all the information that they are dealing with is classified. The players are given an additional topic ("personal interests"), to which they may attribute all actions not strictly related to their work. Such actions are a proxy for the fact that some analysts may check basketball data while on their classified machines or ballet performances, reflecting things in which they are interested. All players are told that there is a system monitoring their actions. Malicious insiders are given an additional topic t

5. Of course, we do not use real classified data, just specify topics the users are told may represent classified information.

approach	honest & malicious user source	system observation	supervised /unsupervised /semi-supervised	assume user history	strictly distinguished train/test
Greitzer & Frincke [22]	synthetic data	psychological (stress, anger, etc.)	unsupervised	no	no
Liu et al. [38]	synthetic data	full knowledge of user actions	supervised	yes	no
ELICIT [5]	genuine honest, synthetic malicious ("red" team)	full knowledge of user actions	supervised	yes	no
GBAD [68]	large synthetic data	transactional data	unsupervised	yes	yes
MITRE [6]	50 human players 25 malicious	full knowledge of user actions	statistical analysis only	no	no
BAIT	795 human players 7 known malicious	player high-level actions	semi-supervised	no	yes

TABLE 1
Comparison of different approaches for insider threat detection

(e.g. design plans for a new missile) and told that their job is to come up with actions that exfiltrate that data, while minimizing the probability of detection by the surveillance system and/or human cybersecurity personnel. This type of worker is told that the topic t is his top priority and that he must try to hide/disguise any actions that he performs related to this topic. Note that the extra malicious topic is performed alongside the other tasks that the player performs. This simulates the behavior of malicious insiders, who perform both regular tasks and a small amount of malicious tasks.

In every round of the game, each player may choose to perform one of the following 13 actions regarding each of the topics assigned to him. These actions are similar to actions studied by other researchers in the past such as Maloof and Stephens' [5] ELICIT system. Note that later in the paper, we will develop features used by our bootstrapped machine learning algorithms on top of these actions; thus these actions are not features.

1) Do nothing.
2) Fetch a document from the confidential internal data-base related to [a selected topic].
3) Ask a colleague to send a document related to [a selected topic].
4) Fetch a document from a CD / DVD related to [a selected topic].
5) Fetch a document from the USB drive related to [a selected topic].
6) Fetch unclassified data from the internet / email, related to [a selected topic].
7) Send an email with encrypted data related to [a selected topic] to the topic's requester.
8) Save data to a CD / DVD related to [a selected topic], and deliver it to the topic's requester.
9) Save data to the USB drive related to [a selected topic], and deliver it to the topic's requester.
10) Send unencrypted data related to [a selected topic] to the topic's requester via email/internet.
11) Print information related to [a selected topic], and hand the pages to the topic's requester.
12) Send encrypted data over the internet related to [a selected topic] to the topic's requester.
13) Edit information on [a selected topic].

Note that the number of actions we allow the user to perform is actually much greater than 13 because 12 of the actions are each parametrized by a topic. Thus, BAIT allows the user to select any of the $12T+1$ topics where T is the total number of possible topics.

Each player needs to perform 35 actions (of which he may do nothing as many times as he wants). Every time a player performs an action related to a topic, that action is listed under the associated topic.

The system does not see the topics, but only knows what actions were performed. We term the actions related to transferring data as "transfer actions". The transfer actions include: save to CD, save to USB, print and 3 "send" type actions: email, internet, unencrypted. For the three send actions, the system can also determine whether the actions were internal or external, resulting in 6 possible "send" type-destination combination actions. The internal destination "send" operations send information within the organization. The external destination send operations send data out of the organization, and can be used by either honest users (e.g. for following/communicating their personal interests) or by malicious users (e.g. to exfiltrate data). However the system does not know if the external destination operations were used by honest users for their personal reasons or by malicious users to send sensitive data to people outside the organization which is strictly forbidden.

3.2 Background on Semi-Supervised Learning and Imbalanced Data

The data available in our domain is generally composed of a very small amount of labeled data and large amounts of unlabeled data. This motivates the

use of Semi Supervised Learning. An extensive survey of semi supervised learning can be found in Zhu's survey [85]. We briefly describe two of the method types suggested in the survey and their applicability to our problem. The oldest methods include generative models that identify distributions in the unlabeled data, and then use the small labeled data set to match a label to each distribution. This method is not suitable when the number of malicious insiders in the data is very small. Another family of methods is the Self-Training family. These methods first classify the labeled data and then use the model to classify the unlabeled data. The newly classified unlabeled data is then added to the labeled data set and a new model on all the labeled data is built. This process can be performed iteratively. Several of the algorithms we propose adapt this idea.

Aside from the need to use Semi Supervised Learning our data is also characterized by being very imbalanced. There are many more honest people than there are malicious insiders. And most of the actions taken by the malicious insiders may be benign with a relatively small number of malicious actions hidden within a larger number of benign actions. This leads us to search for algorithms that learn from imbalanced data. Two comprehensive surveys on imbalanced data [86] and rare data [87] describe variants of imbalanced data. Sometimes the issue of imbalanced data can easily be solved by sampling methods that either undersample the large class or oversample the small class. However in domains where the small class is also rare (the number of episodes is very small) undersampling is of no help and oversampling may lead to overfitting [86]. Another approach suggests assigning different weights to the cost of misclassification of episodes in the learning algorithm. However these techniques are developed for specific paradigms and there is no unifying framework for general cases [86].

A related problem that is also close to our problem is that of partially supervised classification [88], where there is a labeled set of data for the positive data and a mixed data set that has both positive and negative documents but no labeled negative documents. They use an iterative algorithm that first learns from the positive data. Next they select those most likely to be negative from the unlabeled set and add them to the labeled data to build a new model. This stage is repeated iteratively. We implemented a similar method in one of the variations we tested. However our initial model is built on both positive and negative examples (see Algo. 3 for details).

As we perform semi supervised learning on imbalanced data, we seek to combine methods for semi supervised learning with imbalanced data learning. As stated by Haibo and Garcia [86] "...the issue of semi supervised learning under the condition of imbalanced data sets has received very limited attention in the community.". This motivated us to develop the BAIT Algorithms.

3.3 BAIT Algorithms

The BAIT system includes a suite of seven algorithms that are built on top of two classical machine learning algorithms — Support Vector Machines [8] or SVM and Multinomial Naive Bayes [89].

In all SVM methods, we determine the best kernel by using cross validation on the labeled data. We considered linear kernels, polynomial kernels of degree 2 and 3, and the Radial Basis Function (RBF) kernel. We use cross validation on the labeled data also when considering the best method for dealing with imbalanced data. We consider oversampling of the minority group (the malicious insiders), undersampling of the majority group (the benign users) and cost weight adjustments. We propose 5 methods for building the model. In methods 2-5 we use a confidence measure. The confidence is defined as the greatest distance from the separator hyper-plane computed by SVM.

3.3.1 Algorithm BAIT_L - Use Only Labeled Data.

The BAIT_L algorithm is extremely simple and serves as a baseline. Given a set L of labeled data, it merely uses SVM to compute a separator hyperplane and then uses this as a classifier. Nothing particularly intelligent is done other than the use of SVM as a classifier.

3.3.2 Algorithm BAIT_{LUMI} - Use Labeled Data + Unlabeled Malicious Insider.

The BAIT_{LUMI} algorithm (Algo. 1) is smarter than the baselineBAIT_L. It takes as input, a highly imbalanced and very small labeled data set L with very few labeled malicious players as well as an unlabeled data set U, and an integer $k \geq 0$.

It first uses the BAIT_L algorithm above to learn a separator using SVM on the labeled data. Next, it labels all users in U using the SVM learned. It then identifies the top k malicious users (L_{top}) (i.e., the k users with the highest confidence). It then re-learns a new separator, using $L \cup L_{top}$ as a training set. It then associates labels for all users in $U \setminus L_{top}$ using the learned separator (of course, the labeled entries in L_{top} keep their labels.) and denote it L_U. Then it returns the separator learned from $L \cup L_{top} \cup L_U$.

In short, BAIT_{LUMI} constructs a model by classifying the unlabeled data using an SVM learned on the labeled data, chooses the top k malicious insiders from this set in order to reduce the imbalance in the data between malicious and honest users, and then retrains and reclassifies. The idea of choosing the top k malicious users is that these users are the most likely to have been correctly classified in the first step.

Algorithm 1 BAIT_{LUMI}
Input: labeled data L, unlabeled data U, number k
Output: BAIT_{LUMI}

$\mathsf{BAIT}_L \leftarrow$ Build model using SVM on L.
$L_U \leftarrow$ label U using classification of BAIT_L on U
$L_{top} \leftarrow$ top k malicious insiders (highest confidence) from L_U
$M_{L-top} \leftarrow$ Build model using SVM on $L \cup L_{top}$.
$L_U \leftarrow$ label U using classification of M_{L-top} on U
$ANS \leftarrow$ Build model using SVM on $L \cup L_U$.
Return ANS.

3.3.3 Algorithm BAIT_{LHUMI} - Use Labeled Data + Honest and Unlabeled Malicious Insider.

In the BAIT_{LHUMI} algorithm (Algo. 2), we extend the idea in BAIT_{LUMI}. As before, we first learn an SVM model M from the labeled data set. But instead of just looking at the top k users labeled as malicious by M in the unlabeled data set, we identify both the top k malicious users and the top $\ell \cdot k$ honest users and add them to the training set and then recalibrate the model M. The reason for adding only the top k malicious insiders but not adding a much larger number of honest users is to account for the imbalance in the data as in both the real-world and in our data set, the percentage of malicious insiders is very small.

3.3.4 Algorithm $\mathsf{BAIT}_{I-LHUMI}$ - Use Labeled Data and Iteratively Update Unlabeled Data with Malicious User and Honest Users

The $\mathsf{BAIT}_{I-LHUMI}$ algorithm (Algo. 3) adapts the BAIT_{LHUMI} algorithm by iteratively processing the unlabeled data with the most recently learned SVM model, then adds the most likely malicious user and a larger number of the most likely honest users back into the labeled data, and recalibrates. The most likely malicious/honest user is added back into the labeled data set, one at a time, not k or $k \cdot \ell$ at a time as in the BAIT_{LHUMI} approach above. By adding each back one at a time, we hope to obtain better models, due to the fact that the model is recomputed after each new labeled user is captured.

3.3.5 Algorithm $\mathsf{BAIT}_{MI-LHUMI}$ - Use Labeled Data and Iteratively Update Unlabeled Data with Malicious User and Honest Users

In the $\mathsf{BAIT}_{MI-LHUMI}$ algorithm (Algo. 4), we extend the BAIT_{LHUMI} algorithm in a different way. We first learn an SVM model M from all of the labeled training data L. We then classify the unlabeled data U using this learned model. We then randomly choose one malicious user from the top r malicious users in U for some value of r and label this person a malicious user — this is different than what occurs in the BAIT_{LHUMI} algorithm. Then, as in BAIT_{LHUMI}, we choose the top ℓ honest users in U. These $\ell + 1$ users (ℓ honest plus one malicious insider) are then added back to the labeled data set and this process is iterated within the inner loop of the algorithm until all U is labeled. The entire inner loop is executed J times for some $J > 0$ in order to smooth out the random choice made in the computation of L_{mi}. After iterating this J times, we get a total of J groups of malicious insiders. We label the s feature vectors which appear most frequently in these groups as malicious insiders and the others as honest and retrain the SVM using both this data and the *original* labeled data (L).

3.3.6 Naive Bayes (BAIT_{NB}) Algorithm

The BAIT_{NB} algorithm is very simple — we merely applied the classical Multinomial Bayes algorithm on the labeled data and used that classifier to label the unlabeled data.

3.3.7 Bayes with Iterative Probability Update (BAIT_{BIPU}) Algorithm

This algorithm (Algo. 5) is based on Expectation-Maximization (EM). Leveraging the idea presented by Nigam, McCallum and Mitchell [90], we first built a classifier model using the labeled data and then we improved the model iteratively using the unlabeled data together with the probabilities provided in the most recent iteration. This is executed iteratively till the model converges.

3.4 Features Used

We developed various types of features based on the key indicators described earlier. All these features were chosen without observing the unlabeled/test data. Some features count the number of times an action was performed — which we term "basic" features. Some features are derived from the "basic" features using linear combinations, and some are derived by the division of one derived feature by another. The full list of features is given below:

- 16 basic features :
  - The number of times each action was performed by the player, where the 3 sending actions are each split into internal and external sending.
- 12 features derived from sending actions:

Algorithm 2 BAIT$_{LHUMI}$
Input: labeled data L, unlabeled data U, number k, number l
Output: BAIT$_{LHUMI}$

$M_L \leftarrow$ Build model using SVM on L.
$L_U \leftarrow$ label U using classification of M_L on U
$L_{top} \leftarrow$ top k malicious insiders and top $\ell \cdot k$ honest (highest confidence) from L_U
$M_{L-top} \leftarrow$ Build model using SVM on $L \cup L_{top}$.
$L_U \leftarrow$ label U using classification of M_{L-top} on U
$ANS \leftarrow$ Build model using SVM on $L \cup L_U$.
Return ANS.

Algorithm 3 BAIT$_{I-LHUMI}$
Input: labeled data L, unlabeled data U
Output: BAIT$_{I-LHUMI}$

$M_L \leftarrow$ Build model using SVM on L.
$L_{conf} = \emptyset$
$U_r \leftarrow U$
while $U_r \neq \emptyset$ **do**
  $L_U \leftarrow$ label U_r using classification of M_L on U_r
  $L_{new} \leftarrow$ most likely malicious user in L_U and ℓ most likely honest in L_U
    ▷ if there are no feature vectors classified as malicious (or honest) users none are added.
  $L_{conf} \leftarrow L_{conf} + L_{new}$
  $U_r \leftarrow U_r \setminus L_{new}$
  $M_L \leftarrow$ Build model using SVM on $L \cup L_{conf}$.
end while
$ANS \leftarrow M_L$
Return ANS.

Algorithm 4 BAIT$_{MI-LHUMI}$
Input: labeled data L, unlabeled data U, number r, number l, number J, number s
Output: BAIT$_{MI-LHUMI}$

$M_L \leftarrow$ Build model using SVM on L.
for $j = 1$ **do** to J
  $U_r \leftarrow U$
  $S_j \leftarrow \emptyset$
  while $U_r \neq \emptyset$ **do**
    $L_U \leftarrow$ label U_r using classification of M_L on U_r
    $L_{mi} \leftarrow$ randomly select 1 malicious insider from top r malicious insiders from L_U
    $L_{honest} \leftarrow$ select l honest feature vectors with greatest confidence.
    $L_{new} \leftarrow L_{mi} \cup L_{honest}$
    $L_{conf} \leftarrow L_{conf} \cup L_{new}$
    $M_L \leftarrow$ Build model using SVM on $L \cup L_{conf}$
    $U_r \leftarrow U_r \setminus L_{new}$
    $S_j \leftarrow S_j \cup L_{mi}$
  end while
end for
$S \leftarrow U$ where the s most frequent feature vectors from all the S_j are labeled as malicious insiders, and the others are labeled as honest
$ANS \leftarrow$ Build model using SVM on $L \cup S$.
Return ANS.

Algorithm 5 BAIT$_{BIPU}$
Input: labeled data L
Output: BAIT$_{BIPU}$

$L \leftarrow$ labeled data
$M_L \leftarrow$ Build model using MNB on L
repeat
  $L_U \leftarrow$ label unlabeled data with probability of being a malicious insider obtained from M_L (E step)
  $M_L \leftarrow$ Build model using MNB on $L \cup L_U$ M step
until converge
$ANS \leftarrow M_L$
Return ANS.

- There are 3 types of "send" actions. For each of these types we define a feature that is the number of all internal sends (for this type) divided by the total number of "send" actions (internal and external for this type). This provides 3 features, one for each type.
- Sum of all "fetch" actions (fetch from internal DB, ask colleague for documents, fetch a document from CD, fetch a document from USB).
- Sum of all "transfer" actions.
- Sum of all "send" actions.
- Sum of all "send" actions divided by the sum of "transfer" actions.
- Sum of all external "send" actions.
- Sum of all external "send" actions divided by the sum of internal and external "send" actions.
- Sum of all "transfer" actions that are not "send" actions (save to CD, save to USB and print).
- Sum of all "unsuspicious fetch" actions. We define "unsuspicious fetch" actions as fetching classified data that does not involve other people: we refer to actions 2 (fetching from DB),4(fetching from CD/DVD) and 5 (fetching from USB) described in section 3.1.
- Sum of all "unsuspicious fetch" actions (defined in the previous item) divided by the sum of all "fetch" actions.

When feeding these features to the SVM algorithms, we normalize the features to be between -1 and 1.

3.5 Implementation and Experimental Results

We ran the BAIT game (described in section 3.1) on Amazon's Mechanical Turk service (AMT) [91][6]. The total number of subjects who participated in all the experiments included 795, all from the USA of whom 49.8% were females and 50.2% were males. The subjects' ages ranged from 18 to 68, with a mean of 33.2, a standard deviation of 10.1 and median of 30. 7.7% of the subjects had either a PhD or a masters degree, 45.7% had a bachelor's degree, 46.3% of the subjects had a high-school education and only $3(0.36\%)$ subjects had only completed junior highschool.

In the labeled data-set, we ensured that exactly 7 subjects were malicious insiders, and in the unlabeled data-set and in the test data-set, each player had a 7% chance of being a malicious insider. The insiders were explicitly told to hide their actions.

We recruited only subjects with a history of at least 97% approval rate on AMT, and who had completed at least 60 tasks on AMT in the past. We further had all subjects pass a quiz ensuring that they were all diligent in following the instructions. In addition to all this, since the game is an "open" and unusual game, we expected some subjects to have difficulty understanding and playing the game. Therefore, after the subjects completed playing the game they were asked whether they understood the instructions and whether they played similarly to how they would have played in real life. We removed $101(12.7\%)$ subjects who either indicated that they did not understand the game, or that they just picked random actions. This ensured that the subject pool was even more serious.

After removing these subjects there were a total of 100 labeled feature vectors (each feature vector contained data from a single subject over 35 rounds), of which 7 were labeled as malicious insiders and 93 as honest players, 394 unlabeled feature vectors and 200 feature vectors to use for testing.

We ran all 7 algorithms described in Section 3.3 on our data. We first describe results for the SVM algorithms. We used the following settings: we set $k = 11$ since we assumed that the unlabeled data was sampled from the same distribution as the labeled data. Because there are 7 (of 100) malicious insiders in the labeled data, we know with a confidence of 95% that there are at least eleven malicious insiders. We set $s = 28$ since we assumed 7% of malicious insiders in the unlabeled data, since we know that we have that proportion in the labeled data. We set $l = \frac{93}{7} = 13.3$, $r = 3$,$J = 50$. Cross validation on the labeled data encouraged us to use a linear kernel and to set the cost weight function at 0.01 for the honest feature vectors

6. For a comparison between AMT and other recruitment methods see [92].

and 0.1 for the malicious. We used all the features described in 3.4 for the SVM models. The results are described in table 2. As presented in the table, all methods except BAIT_L reached a similar recall of $0.5 - 0.6$, however, the precision rate varied from 0.17 in BAIT_{LUMI} to 0.3 in $\text{BAIT}_{MI-LHUMI}$, resulting in an f-measure of 0.4 (in $\text{BAIT}_{MI-LHUMI}$). In all algorithms the recall rate was higher than precision. Indeed, when considering insider threat, recall is more important as it implies that more insiders will be detected, even if the precision rate is relatively low, which implies that more (possibly honest) people will be interviewed and required to explain their actions.

We also ran Multinomial Naive Bayes on our data. The actions played in the game can be seen as analogous to words, assuming internal and external sends are seen as different words. This motivated us to test Multinomial Naive Bayes on the set of "basic" features. Unfortunately the "derived" features cannot naturally be used in Multinomial Naive Bayes. We wanted to compare the Multinomial Naive Bayes results to SVM, However SVM did not succeed in building a model solely using the "basic" features. The results are described in table 3. Interestingly, BAIT_{BIPU} does not seem to work well with imbalanced data. Therefore we can conclude that if it is possible to add additional smart features (the derived sending actions), it is better to use the $\text{BAIT}_{MI-LHUMI}$ method. However, if it is not possible to add additional smart features, it is better to use BAIT_{NB} (which is based on Multinomial Naive Bayes). When considering only the labeled data, the Naive Bayes method without the additional smart features seems to perform at least as well as the SVM with the additional smart features.

Last but not least, we briefly note that we tested the use of one-class SVMs that have been used in prior work [63] on our data. The ocSVMs performed very poorly on our highly imbalanced data with very little past history. Consequently we do not pointlessly use the space in this paper to report the results.

3.6 Discussion

Since the BAIT game is an open game it has no right or wrong action and there is no scoring or bonus available. Still we observed that most of the players played seriously. We believe that in situations in which both sides have higher motivation to play their roles, it should be easier to detect the malicious players.

Another parameter which may have a major impact on the detection rate is system observation. In our work, we assumed that the system did not have any indication on the topics of the documents handled by the users. Therefore, it could not, for instance, observe that a user is fetching a document which he isn't supposed to view. We assume that if the system would have cues on the subjects / classification fields of the documents handled by the users, the detection rate would significantly increase. The BAIT game may be expanded to allow such observations by the system. However, since the BAIT game is already complex, such an expansion must be done in a manner that will not make the game much more complex.

We wanted to ensure that we were not influenced in any way by the unlabeled and test data sets while developing the proposed algorithms and selecting features. Therefore we were extremely careful not observe any of the unlabeled and test data before we completed our study. However once we finished the study and wrote the results, we performed a statistical analysis of the players behavior. In total there were 40 malicious players and 654 benign players. Below we describe some of the results of comparing the behavior of the two. We report statistical significance using the t-test with $\alpha = 0.05$. The following results are compatible with the trends found in MITRE [6] which were applicable to our domain:

1) The malicious players were more active and chose the "nothing" action significantly ($p < 0.01$) *less* times (3.77) than the benign players (6.52).
2) The malicious players fetched significantly ($p < 0.05$) *more* "sensitive" information (9.8) than the benign players (8.26). (See section 3.4 for the definition of "sensitive fetches").
3) The malicious players appeared to save *more* data to CD/DVD (1.2) and USB (2.225) compared to the benign players CD/DVD (1.07) and USB(1.82). However, these differences were not found to be statistically significant in our study. These findings vary slightly from those of Caputo et al. [6] which were found to be statistically significant.
4) The malicious players edited the data slightly *less* (3.77) compared to the benign players (3.88). However, these differences also were not found to be statistically significanct. These findings also vary from the findings presented in Caputo et al.s' study [6] which showed the difference to be statistically significant.

In addition, we found the following statistically significant differences between the malicious and the benign players. No similar findings have been reported in prior work.

1) The malicious players sent significantly ($p < 0.001$) *more* information out of the organization with an average of (2.92) times, in contrast to the benign players (0.68).
2) The malicious players fetched significantly ($p = 0.05$) *less* unclassified data (2.87) in contrast to the benign players (3.40).

The similarity between the trends we observed and the trends found in MITRE indicates that the BAIT game is a good platform to simulate insider threat

Algorithm	trueNeg	falsePos	falseNeg	truePos	precision	recall	fmeasure
SVM: BAIT$_{L}$	181	9	7	3	0.25	0.3	0.273
SVM: BAIT$_{LUMI}$	160	30	4	6	0.17	0.6	0.261
SVM: BAIT$_{LHUMI}$	175	15	5	5	0.25	0.5	0.33
SVM: BAIT$_{I-LHUMI}$	171	19	4	6	0.24	0.6	0.342
SVM: BAIT$_{MI-LHUMI}$	176	14	4	6	0.3	0.6	0.4

TABLE 2
SVM Classification results

Algorithm	trueNeg	falsePos	falseNeg	truePos	precision	recall	fmeasure
BAIT$_{NB}$	169	21	6	4	0.16	0.4	0.23
BAIT$_{BIPU}$	97	93	3	7	0.07	0.7	0.13

TABLE 3
Multinomial Naive Bayes Classification results

and analyze malicious behavior.

Note that in this study we do not address the scenario where the threat is performed by multiple people. The situation in which several people work together to create a malicious attack is definitely a complex threat to detect. It would be interesting to extend our work to this type of scenario in the future.

4 Conclusion

Insider threat is a growing problem in many organizations. Although recent episodes in the press such as the leaks caused by the Wikileakers, Bradley Manning and Edward Snowden, have made headlines in the global press, the problem of insider threats has long threatened companies in many different sectors[1].

In this paper, we first presented a detailed survey of research on insider threat compiled from many different disciplines. Specifically, we identified studies of insider threat from social sciences (psychology, sociology and criminology), from electrical engineering, and of course, from computer science. All of these diverse studies have much to offer one another.

After the detailed survey, we described our own work on BAIT (Behavioral Analysis of Insider Threat). We note that insider threat studies are bedeviled by the following problems.

1) The number of malicious insiders compared to normal honest users in most data sets is extremely small, leading to very highly imbalanced data sets. The use of supervised learning algorithms on such data is highly challenging.
2) Past studies have mainly focused on how insider attacks occurred in the past. Detecting known past attacks does not pose a big problem. In contrast, in BAIT, we are interested in new types of attacks that might be used in the future.
3) There is a lack of good models of how insiders may attack systems with new and hitherto unseen attacks.

In order to address these points and to really learn models of how insiders may attack systems in the future, we designed a one-person game (also called an "open game") and recruited 795 people to play the game on Amazon Mechanical Turk — most of whom were assigned the role of *benign* users while a very small number were *malicious*. In keeping with the imbalance in real data, we used a highly imbalanced number of malicious insiders. We further assumed that we know the role of only 100 subjects and that all the rest of the data is either unlabeled data or reserved for testing. We developed 7 algorithms (on top of SVMs and Bayesian methods) and successively (automatically) built increasingly larger labeled data sets by a suite of "bootstrapping" methods designed to increase the size of the training data. We tested the accuracy (precision and recall) of these algorithms. The best algorithms have a recall of 0.6 with a precision of 0.3 — a related algorithm has a higher recall (0.7) but a much lower precision (0.07). The appropriate algorithm to choose for a given organization would depend on the precision the organization is willing to tolerate.

More interestingly, our work casts light on the types of properties that distinguish malicious insiders from benign ones. Specifically:

1) Malicious insiders are more likely to be more active than benign insiders (statistically significant when $p < 0.001$, i.e. at the 99.9% level).
2) Malicious players fetch significantly more "sensitive" information than benign players (statistically significant when $p < 0.05$, i.e. at the 95% level).
3) Malicious players send significantly more data out of their organization (statistically significant when $p < 0.001$, i.e. at 99.9% level) than benign insiders.
4) Malicious players fetched significantly less "unclassified" data (statistically significant when $p < 0.05$, i.e. at 95% level) than benign players.

In addition, our findings only slightly confirm two prior results. The hypothesis that malicious players save more data on removable media (e.g. USB sticks or CDs) is true in our game data, but is without

statistical significance. Likewise, the hypothesis that malicious users edited data less than benign ones was also empirically observed, but not at a statistically valid level.

We note that past work has highlighted the psychological aspects of insider threat — but few works seem to have rigidly tested, even in a synthetic environment, the use of natural language processing for extracting symptoms that are associated with insider threat. Future work may well use sentiment analysis techniques [93], [94]. Merging psychological insights, natural language processing techniques with behavioral signatures may well be an important direction for future work on insider threat.

Acknowledgement. Some authors may have been supported by ARO grants W911NF0910206, W911NF1160215, W911NF1110344, W911NF1410358, an ARO/Penn State MURI award, ONR grant N000140910685, Maryland Procurement Office Contract No. H98230-14-C-0137 and Maafat.

REFERENCES

[1] M. Randazzo, M. Keeney, E. Kowalski, D. Cappelli, and A. Moore, "Insider threat study: Illicit cyber activity in the banking and finance sector," *US Secret Service and CERT Coordination Center/Software Engineering Institute: Philadelphia, PA*, p. 25, 2004.

[2] S. L. Pfleeger and S. J. Stolfo, "Addressing the insider threat," *Security & Privacy, IEEE*, vol. 7, no. 6, pp. 10–13, 2009.

[3] G. Fyffe, "Addressing the insider threat," *Network Security*, vol. 2008, no. 3, pp. 11–14, 2008.

[4] C. P. Pfleeger, "Reflections on the insider threat," in *Insider Attack and Cyber Security*. Springer, 2008, pp. 5–16.

[5] M. A. Maloof and G. D. Stephens, "ELICIT: A system for detecting insiders who violate need-to-know," in *Recent Advances in Intrusion Detection*. Springer, 2007, pp. 146–166.

[6] D. Caputo, M. Maloof, and G. Stephens, "Detecting insider theft of trade secrets," *Security & Privacy, IEEE*, vol. 7, no. 6, pp. 14–21, 2009.

[7] M. Salem, S. Hershkop, and S. Stolfo, "A survey of insider attack detection research," in *Insider Attack and Cyber Security*, ser. Advances in Information Security, S. Stolfo, S. Bellovin, A. Keromytis, S. Hershkop, S. Smith, and S. Sinclair, Eds. Springer US, 2008, vol. 39, pp. 69–90. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-77322-3_5

[8] I. Steinwart and A. Christmann, *Support vector machines*. Springer, 2008.

[9] T. E. Senator, H. G. Goldberg, A. Memory, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, D. A. Bader, E. Chow, I. Essa, J. Jones, V. Bettadapura, D. H. Chau, O. Green, O. Kaya, A. Zakrzewska, E. Briscoe, R. I. L. Mappus, R. McColl, L. Weiss, T. G. Dietterich, A. Fern, W.-K. Wong, S. Das, A. Emmott, J. Irvine, J.-Y. Lee, D. Koutra, C. Faloutsos, D. Corkill, L. Friedland, A. Gentzel, and D. Jensen, "Detecting insider threats in a real corporate database of computer usage activity," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 1393–1401. [Online]. Available: http://doi.acm.org/10.1145/2487575.2488213

[10] G. Magklaras and S. Furnell, "Insider threat prediction tool: Evaluating the probability of it misuse," *Computers & Security*, vol. 21, no. 1, pp. 62–73, 2001.

[11] J. Myers, M. R. Grimaila, and R. F. Mills, "Towards insider threat detection using web server logs," in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*. ACM, 2009, p. 54.

[12] G. Jabbour and D. A. Menasce, "The insider threat security architecture: a framework for an integrated, inseparable, and uninterrupted self-protection mechanism," in *Computational Science and Engineering, 2009. CSE'09. International Conference on*, vol. 3. IEEE, 2009, pp. 244–251.

[13] M. Bishop, S. Engle, S. Peisert, S. Whalen, and C. Gates, "We have met the enemy and he is us," in *Proceedings of the 2008 workshop on New security paradigms*, ser. NSPW '08, 2008, pp. 1–12.

[14] M. Bishop, S. Engle, D. A. Frincke, C. Gates, F. L. Greitzer, S. Peisert, and S. Whalen, "A risk management approach to the insider threat," in *Insider Threats in Cyber Security*. Springer, 2010, pp. 115–137.

[15] J. Hunker and C. W. Probst, "Insiders and insider threatsan overview of definitions and mitigation techniques," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 2, no. 1, pp. 4–27, 2011.

[16] S. Sinclair and S. W. Smith, "Preventative directions for insider threat mitigation via access control," in *Insider Attack and Cyber Security*. Springer, 2008, pp. 165–194.

[17] E. E. Schultz, "A framework for understanding and predicting insider attacks," *Computers & Security*, vol. 21, no. 6, pp. 526–531, 2002.

[18] B. Wood, "An insider threat model for adversary simulation," *SRI International, Research on Mitigating the Insider Threat to Information Systems*, vol. 2, pp. 1–3, 2000.

[19] M. Warkentin and R. Willison, "Behavioral and policy issues in information systems security: the insider threat," *European Journal of Information Systems*, vol. 18, no. 2, p. 101, 2009.

[20] R. Willison and M. Warkentin, "Motivations for employee computer crime: understanding and addressing workplace disgruntlement through the application of organisational justice," in *Proceedings of the IFIP TC8 International Workshop on Information Systems Security Research*. International Federation for Information Processing, 2009, pp. 127–144.

[21] C. Colwill, "Human factors in information security: The insider threat who can you trust these days?" *Information Security Technical Report*, vol. 14, no. 4, pp. 186 – 196, 2009, ¡ce:title¿Human Factors in Information Security¡/ce:title¿.

[22] F. L. Greitzer and D. A. Frincke, "Combining traditional cyber security audit data with psychosocial data: towards predictive modeling for insider threat mitigation," in *Insider Threats in Cyber Security*. Springer, 2010, pp. 85–113.

[23] E. Cole and S. Ring, *Insider Threat: Protecting the Enterprise from Sabotage, Spying, and Theft: Protecting the Enterprise from Sabotage, Spying, and Theft*. Syngress, 2005.

[24] F. L. Greitzer, L. J. Kangas, C. F. Noonan, A. C. Dalton, and R. E. Hohimer, "Identifying at-risk employees: Modeling psychosocial precursors of potential insider threats," in *System Science (HICSS), 2012 45th Hawaii International Conference on*. IEEE, 2012, pp. 2392–2401.

[25] M. Theoharidou, S. Kokolakis, M. Karyda, and E. Kiountouzis, "The insider threat to information systems and the effectiveness of iso17799," *Computers & Security*, vol. 24, no. 6, pp. 472–484, 2005.

[26] D. W. Straub and R. J. Welke, "Coping with systems risk: security planning models for management decision making," *Mis Quarterly*, pp. 441–469, 1998.

[27] J. Lee and Y. Lee, "A holistic model of computer abuse within organizations," *Information management & computer security*, vol. 10, no. 2, pp. 57–63, 2002.

[28] W. F. Skinner and A. M. Fream, "A social learning theory analysis of computer crime among college students," *Journal of Research in Crime and Delinquency*, vol. 34, no. 4, pp. 495–518, 1997.

[29] R. C. Hollinger, "Crime by computer: Correlates of software piracy and unauthorized account access," *Security Journal*, vol. 4, no. 1, pp. 2–12, 1993.

[30] I. Ajzen, "Perceived behavioral control, self-efficacy, locus of control, and the theory of planned behavior1," *Journal of applied social psychology*, vol. 32, no. 4, pp. 665–683, 2002.

[31] R. Willison, "Understanding and addressing criminal opportunity: the application of situational crime prevention to is security," *Journal of Financial Crime*, vol. 7, no. 3, pp. 201–210, 2000.

[32] I. J. Martinez-Moyano, E. Rich, S. Conrad, D. F. Andersen, and T. R. Stewart, "A behavioral theory of insider-threat risks:

A system dynamics approach," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 18, no. 2, p. 7, 2008.
[33] J. A. Swets, "The relative operating characteristic in psychology," *Science*, vol. 182, no. 4116, pp. 990–1000, 1973.
[34] J. A. Swets, R. M. Dawes, and J. Monahan, "Psychological science can improve diagnostic decisions," *Psychological science in the public interest*, vol. 1, no. 1, pp. 1–26, 2000.
[35] I. J. Martinez-Moyano, S. H. Conrad, and D. F. Andersen, "Modeling behavioral considerations related to information security," *Computers & Security*, vol. 30, no. 67, pp. 397 – 409, 2011.
[36] C. W. Probst and J. Hunker, "The risk of risk analysis and its relation to the economics of insider threats," in *Economics of information security and privacy*. Springer, 2010, pp. 279–299.
[37] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448 – 3470, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138912860700062X
[38] A. Liu, C. Martin, T. Hetherington, and S. Matzner, "A comparison of system call feature representations for insider threat detection," in *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*. IEEE, 2005, pp. 340–347.
[39] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of computer security*, vol. 6, no. 3, pp. 151–180, 1998.
[40] Y. Liao and V. R. Vemuri, "Using text categorization techniques for intrusion detection." in *USENIX Security Symposium*, vol. 12, 2002.
[41] A. Liu, C. E. Martin, T. Hetherington, and S. Matzner, "Ai lessons learned from experiments in insider threat detection." in *AAAI Spring Symposium: What Went Wrong and Why: Lessons from AI Research and Applications*. AAAI, 2006, pp. 49–55. [Online]. Available: http://dblp.uni-trier.de/db/conf/aaaiss/aaaiss2006-8.html#LiuMHM06
[42] M. Ghallab, D. Nau, and P. Traverso, *Automated planning: theory & practice*. Access Online via Elsevier, 2004.
[43] D. Barbará, J. Couto, S. Jajodia, and N. Wu, "Adam: A testbed for exploring the use of data mining in intrusion detection," *SIGMOD Rec.*, vol. 30, no. 4, pp. 15–24, dec 2001. [Online]. Available: http://doi.acm.org/10.1145/604264.604268
[44] D. E. Denning, "An intrusion-detection model," *Software Engineering, IEEE Transactions on*, no. 2, pp. 222–232, 1987.
[45] W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang, "Real time data mining-based intrusion detection," in *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, vol. 1. IEEE, 2001, pp. 89–100.
[46] M. Albanese, C. Molinaro, F. Persia, A. Picariello, and V. Subrahmanian, "Finding unexplained activities in video," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two*. AAAI Press, 2011, pp. 1628–1634.
[47] ——, "Discovering the top-k" unexplained" sequences in time-stamped observation data," *IEEE Transactions on Knowledge and Data Engineering*, 2013.
[48] R. Erbacher, S. Jajodia, C. Molinaro, F. Persia, A. Picariello, G. Sperli, and V. Subrahmanian, "Recognizing unexplained behavior in network traffic," in *Network Science and Cybersecurity (ed. R.E. Pino)*. Springer Science & Business, 2014, p. 39.
[49] J. Shavlik and M. Shavlik, "Selection, combination, and evaluation of effective software sensors for detecting abnormal computer usage," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: ACM, 2004, pp. 276–285. [Online]. Available: http://doi.acm.org/10.1145/1014052.1014084
[50] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Mach. Learn.*, vol. 2, no. 4, pp. 285–318, Apr. 1988. [Online]. Available: http://dx.doi.org/10.1023/A:1022869011914
[51] A. Richardson, G. A. Kaminka, and S. Kraus, "Cubs: Multivariate sequence classification using bounded z-score with sampling," in *ICDM Workshops*, 2010, pp. 72–79.
[52] I. Ray and N. Poolsapassit, "Using attack trees to identify malicious attacks from authorized insiders," in *Computer Security ESORICS 2005*, ser. Lecture Notes in Computer Science, S. Vimercati, P. Syverson, and D. Gollmann, Eds. Springer Berlin Heidelberg, 2005, vol. 3679, pp. 231–246.
[53] W. Young, H. Goldberg, A. Memory, J. Sartain, and T. Senator, "Use of domain knowledge to detect insider threats in computer activities," in *Security and Privacy Workshops (SPW), 2013 IEEE*, 2013, pp. 60–67.
[54] K. Nance and R. Marty, "Identifying and visualizing the malicious insider threat using bipartite graphs," in *System Sciences (HICSS), 2011 44th Hawaii International Conference on*. IEEE, 2011, pp. 1–9.
[55] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi, "Computer intrusion: Detecting masquerades," *Statistical Science*, pp. 58–74, 2001.
[56] L. Spitzner, "Honeytokens: The other honeypot," 2003.
[57] ——, *Honeypots: tracking hackers*. Addison-Wesley Reading, 2003, vol. 1.
[58] J. Levine, R. LaBella, H. Owen, D. Contis, and B. Culver, "The use of honeynets to detect exploited systems across large enterprise networks," in *Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society*. IEEE, 2003, pp. 92–99.
[59] A. Kołcz, A. Chowdhury, and J. Alspector, "The impact of feature selection on signature-driven spam detection," in *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS-2004)*, 2004.
[60] S. Muhlbach, M. Brunner, C. Roblee, and A. Koch, "Malcobox: Designing a 10 gb/s malware collection honeypot using reconfigurable technology," in *Field Programmable Logic and Applications (FPL), 2010 International Conference on*. IEEE, 2010, pp. 592–595.
[61] B. McCarty, "Automated identity theft," *Security & Privacy, IEEE*, vol. 1, no. 5, pp. 89–92, 2003.
[62] L. Spitzner, "Honeypots: Catching the insider threat," in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. IEEE, 2003, pp. 170–179.
[63] B. M. Bowen, M. Ben Salem, S. Hershkop, A. D. Keromytis, and S. J. Stolfo, "Designing host and network sensors to mitigate the insider threat," *Security & Privacy, IEEE*, vol. 7, no. 6, pp. 22–29, 2009.
[64] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection." in *NIPS*, vol. 12, 1999, pp. 582–588.
[65] M. Kandias, A. Mylonas, N. Virvilis, M. Theoharidou, and D. Gritzalis, "An insider threat prediction model," in *Trust, Privacy and Security in Digital Business*. Springer, 2010, pp. 26–37.
[66] R. Chinchani, A. Iyer, H. Q. Ngo, and S. Upadhyaya, "Towards a theory of insider threat assessment," in *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*. IEEE, 2005, pp. 108–117.
[67] S. Noel, M. Jacobs, P. Kalapa, and S. Jajodia, "Multiple coordinated views for network attack graphs," in *Visualization for Computer Security, 2005.(VizSEC 05). IEEE Workshop on*. IEEE, 2005, pp. 99–106.
[68] W. Eberle, J. Graves, and L. Holder, "Insider threat detection using a graph-based approach," *Journal of Applied Security Research*, vol. 6, no. 1, pp. 32–81, 2010.
[69] P. Parveen, J. Evans, B. Thuraisingham, K. Hamlen, and L. Khan, "Insider threat detection using stream mining and graph mining," in *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*, 2011, pp. 1102–1110.
[70] P. Parveen, Z. Weger, B. Thuraisingham, K. Hamlen, and L. Khan, "Supervised learning for insider threat detection using stream mining," in *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, 2011, pp. 1032–1039.
[71] S. Jajodia, A. K. Ghosh, V. Subrahmanian, V. Swarup, C. Wang, and X. S. Wang, *Moving Target Defense II: Application of Game Theory and Adversarial Modeling*. Springer, 2012, vol. 100.
[72] J. Pita, M. Jain, M. Tambe, F. Ordóñez, and S. Kraus, "Robust solutions to stackelberg games: Addressing bounded rationality and limited observations in human cognition," *Artif. Intell.*, vol. 174, no. 15, pp. 1142–1171, 2010.
[73] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, "A survey of game theory as applied to network security," in

System Sciences (HICSS), 2010 43rd Hawaii International Conference on. IEEE, 2010, pp. 1–10.
[74] T. Alpcan and T. Basar, "A game theoretic approach to decision and analysis in network intrusion detection," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 3. IEEE, 2003, pp. 2595–2600.
[75] D. Liu, X. Wang, and J. Camp, "Game-theoretic modeling and analysis of insider threats," *International Journal of Critical Infrastructure Protection*, vol. 1, no. 0, pp. 75 – 80, 2008.
[76] E. Rich, I. J. Martinez-Moyano, S. Conrad, D. M. Cappelli, A. P. Moore, T. J. Shimeall, D. F. Andersen, J. J. Gonzalez, R. J. Ellison, H. F. Lipson *et al.*, "Simulating insider cyber-threat risks: A model-based case and a case-based model," in *Proceedings of the 23rd International Conference of the System dynamics Society*, 2005, pp. 17–21.
[77] S. Kraus, P. Hoz-Weiss, J. Wilkenfeld, D. R. Andersen, and A. Pate, "Resolving crises through automated bilateral negotiations," *Artificial Intelligence*, vol. 172, no. 1, pp. 1–18, 2008.
[78] A. Rosenfeld, I. Zuckerman, A. Azaria, and S. Kraus, "Combining psychological models with machine learning to better predict peoples decisions," *Synthese*, vol. 189, no. 1, pp. 81–93, 2012.
[79] T. H. Nguyen, R. Yang, A. Azaria, S. Kraus, and M. Tambe, "Analyzing the effectiveness of adversary modeling in security games," in *Proc. of AAAI*, 2013.
[80] I. Kantzavelou and S. Katsikas, "A game-based intrusion detection mechanism to confront internal attackers," *Computers & Security*, vol. 29, no. 8, pp. 859 – 874, 2010.
[81] R. Yang, A. X. Jiang, M. Tambe, and F. Ordóñez, "Scaling-up security games with boundedly rational adversaries: A cutting-plane approach," in *Proc. of IJCAI*, 2013.
[82] K. Tang, M. Zhao, and M. Zhou, "Cyber insider threats situation awareness using game theory and information fusion-based user behavior predicting algorithm," *Journal of Information & Computational Science*, vol. 8, no. 3, pp. 529–545, 2011.
[83] K. P. Murphy, "Dynamic bayesian networks: representation, inference and learning," Ph.D. dissertation, University of California, 2002.
[84] N. Zhang, W. Yu, X. Fu, and S. K. Das, "Maintaining defender's reputation in anomaly detection against insider attacks," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 40, no. 3, pp. 597–611, 2010.
[85] X. Zhu, "Semi-supervised learning literature survey," Computer Sciences, University of Wisconsin-Madison, Tech. Rep. 1530, 2008.
[86] H. He and E. Garcia, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 9, pp. 1263–1284, 2009.
[87] G. M. Weiss, "Mining with rarity: a unifying framework," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 7–19, 2004.
[88] B. Liu, W. S. Lee, P. S. Yu, and X. Li, "Partially supervised classification of text documents," in *ICML*, vol. 2. Citeseer, 2002, pp. 387–394.
[89] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, "Multinomial naive bayes for text categorization revisited," in *AI 2004: Advances in Artificial Intelligence*. Springer, 2005, pp. 488–499.
[90] K. Nigam, A. McCallum, and T. Mitchell, "Semi-supervised text classification using em," *Semi-Supervised Learning*, pp. 33–56, 2006.
[91] Amazon, "Mechanical Turk services," http://www.mturk.com/, 2013.
[92] G. Paolacci, J. Chandler, and P. G. Ipeirotis, "Running experiments on Amazon Mechanical Turk," *Judgment and Decision Making*, vol. 5, no. 5, 2010.
[93] F. Benamara, C. Cesarano, A. Picariello, D. R. Recupero, and V. S. Subrahmanian, "Sentiment analysis: Adjectives and adverbs are better than adjectives alone." in *ICWSM*, 2007.
[94] V. S. Subrahmanian and D. Reforgiato, "Ava: Adjective-verb-adverb combinations for sentiment analysis," *Intelligent Systems, IEEE*, vol. 23, no. 4, pp. 43–50, 2008.