Proactive Detection of Insider Attacks

Benjamin Liebald, Dan Roth, Neelay Shah, and Vivek Srikumar

University of Illinois, Urbana-Champaign, USA

Abstract. Insider attacks are a significant threat to IT infrastructures and are difficult to detect. The problem is exacerbated if the attacker explicitly tries to masquerade as a legitimate user and evade detection. In this paper, we describe a novel approach for detecting these attacks, where the intrusion detection system (IDS) proactively influences the user's perception of the system. The IDS does so by switching among a set of situational contexts and observing the user's reaction to these changes. This is done in a way that poses no significant problem to legitimate users, but creates difficulties for attackers that have learned the system in specific contexts, and therefore cannot improvise well enough to avoid being detected. We present a framework for a generic proactive IDS that shows promising experimental results, suggesting that this method can indeed be effective in detecting masquerade attacks in a variety of domains. We also present an implementation of this idea in a behavioral biometrics domain, where we show that making the IDS proactive enables detection of masquerades.

1 Introduction

Insider attacks are a serious threat to the security of information. This is because hostile insiders have both the motivation and opportunity to impersonate legitimate users and access unauthorized information. This threat gets aggravated if, in addition to having motivation and opportunity, the intruder also *learns* to masquerade as a legitimate user. Conventional anomaly detection systems attempt to solve this by learning a model of the legitimate users' normal behavior. Any user whose actions sufficiently deviate from the expected behavior is classified as an intrusion ([1], [2]). These ideas are applied for detecting masquarades by Schonlau and his colleagues in [3], which compares six methods for masquerade detection. Futher improvements in masquerader detection were presented in [4] and [5].

Intrusion Detection Systems (IDS) can be evaded by a masquerader who expressly learns to pretend to be a legitimate user. Consider the following scenario – Alice is an engineer and Bob is a accountant in a company. Bob wants to steal engineering plans and manages to get Alice's password. If Bob accesses the system using Alice's password, then the anomaly detection system will detect this as an intrusion because Bob's actions will be different from Alice's. However, if Bob observes Alice's behavior and learns to imitate her, then an IDS that looks for anomalous behavior may not report Bob's activities as intrusive. Bob knows

how to achieve his goal by crafting a typical set of actions that looks like normal behavior to the IDS. Thus, in order to beat an anomaly detector, the intruder should learn to masquerade as a legitimate user. This scenario is formalized in Sect. 2.

However, it is reasonable to assume that even if Bob manages to masquerade as Alice, he need not necessarily be able to handle abnormal situations the same way as Alice does. For example, in the event of a database error, Bob's actions need not be similar to the ones of Alice, who is an engineer. This motivates the idea of an IDS that proactively presents uncommon situations to the users and observes their reaction. If done right, it can allow the IDS to observe users' behavior in other situations than those they "prepared" for, and thus with minimal interruption to normal work, keep the system secure.

In this paper, we introduce the idea of a Proactive Intrusion Detection System that is based on this intuition (Sect. 3). To be in a realistic masquerading situation, we assume that the users, and therefore potential masqueraders, have access to at least some of the logs of system interactions and can learn the normal behavior of other users. The underlying idea behind proactive intrusion detection is to expose attackers and legitimate users of the system to multiple contexts, in a way that poses no problem to legitimate users, but creates significant difficulties to attackers who learned to use the system in a specific context, and therefore cannot improvise well enough to avoid being detected. The different situations that the IDS presents to the user, called modes of the IDS, will depend on the system that is being protected. One of the modes will be the normal mode of operation. The ability to change modes allows the IDS to influence the user's actions and observe their reactions to the changes.

If the IDS switches between modes, then the problem of choosing which mode to present to the user becomes an important one. On one hand, the IDS must pick modes such that even the smart masquerader will be identified. On the other hand, the legitimate users should not face excessive disruption in normal activity. These two goals are often diametrically opposed. We address the problem of mode selection in this paper in Sect. 4 and present algorithms for proactive systems that differ in the way they choose modes.

The performance of the algorithms is experimentally verified. We present the results of two sets of experiments. The ideas were first verified via simulation to show that proactive intrusion detection does indeed detect smart masqueraders who learn to behave like a legitimate user. We also show that this idea is feasible in a real world situation in a behavioral biometrics domain. Using data collected from real users, we constructed a strong masquerading attack that is not detected by a conventional anomaly detector and show that the proactive IDS detects it. The design and results of the experiments are discussed in Sect. 5. The proposed system has wide ranging applications. We suggest ways to apply proactive intrusion to detect masquerades in the UNIX command shell environment, transaction systems and even online games (Sect. 6).

2 Problem Definition

Intrusion detection is a binary classification problem – given the user's actions, a classifier has to decide whether the user is an intruder or not. The classification is done over a sequence of actions generated by the user.

Let *Alice* be a user. Let the set of user actions be

$$\mathcal{X} = \{X_1, X_2, \cdots, X_{|\mathcal{X}|}\}$$

where each X_i is an individual user action and the user can choose from $|\mathcal{X}|$ such actions. The user's actions are influenced by her eventual goal and also by feedback from the system.

The task of the IDS is to identify whether a block of actions were performed by a user or not. Let the size of the block under consideration be $\mathcal{N}$. Let x_i represent the i^{th} action in the block. For notational convenience, let $x_1^{\mathcal{N}}$ be the set of all the actions in the block. Then, IDS has to learn a classifier $\mathcal{C}_{Alice}$ that identifies whether $x_1^{\mathcal{N}}$ were performed by *Alice* or not.

The classifier is a function that, given $x_1^{\mathcal{N}}$, extracts features from it and based on these features, assigns a label to the block that states whether it represents a legitimate user or an intruder. Existing literature in masquerade detection ([3–5] and others) focuses extensively on the types of classifiers that can be used to define the intrusion detection system and their relative merits and demerits. However, recent work in machine learning ([6]), shows that all classifiers used today – both discrimiative and probabilistic – can be thought of in terms of a conditional probability distribution. This allows us to limit our discussion to a probabilistic classifier that labels a window of actions as intrusive if the probability of those actions given the current user is less than a threshold. Equivalently, a block of actions $x_1^{\mathcal{N}}$ is labeled as an intrusion if

$$-\log P(x_1^{\mathcal{N}}|Alice) > \theta \tag{1}$$

If the condition is not satisfied, the block is labeled as a legitimate one. θ is the threshold that may be learned by cross-validation over the training data.

Many anomaly detection systems in the literature (for example, [7, 8]) models the user's actions as a Markov model. The assumption made is that the present data is indepenedent of the future, given the past. Making this assumption allows us to simplify the probability of actions as follows –

$$\begin{aligned} -\log P(x_1^{\mathcal{N}}|Alice) &= -\log\left(\prod_{i=1}^{\mathcal{N}} P(x_i|x_1^{i-1}, Alice)\right) \\ &= \sum_{i=1}^{\mathcal{N}} -\log P(x_i|x_1^{i-1}, Alice) \end{aligned}$$

Thus, a block of actions is labeled as an intrusion if the following holds –

$$\sum_{i=1}^{\mathcal{N}} -\log P(x_i|x_1^{i-1}, Alice) > \theta \tag{2}$$

Equation 2 defines the classifier $\mathcal{C}_{Alice}(x_1^{\mathcal{N}})$.

Let *Bob* be another user who wants to masquerade as *Alice*. In order to do so, *Bob* learns his own model of *Alice*'s actions and uses this to generate actions as *Alice*. Note that, in addition to obtaining *Alice*'s login credentials, *Bob* learns to masquerade as *Alice* by observing her usage of the system. This means that the intruder is smarter than the ones in [3] and other similar work, where the intruders only know the login credentials, but do not transform their input to match the user's input patterns. While this may seem like a strong assumption, we show the feasibility of this in the experiments in Sect. 5.2.

3 Proactive Intrusion Detection

In the scenario described in Sect. 2, it is clear that if *Bob* can perfectly impersonate *Alice* (a practically impossible task), then no IDS will be able to distinguish between *Alice* and *Bob*. However, it is reasonable to assume that while *Bob* can masquerade as *Alice* in normal situations, he will not have enough data to learn her behavior in all situations. This assumption is true if *Alice* is an "expert" in her domain and *Bob*, who is not one, has learned to behave like her. Therefore, in uncommon situations, it is unlikely that he will behave the way she does. Am intrusion detection should take advantage of such distinctive responses of users to specific situations.

We introduce the notion of a Proactive Intrusion Detection System which is based on this idea. In this proposed architecture, an anomaly detector observes a user's interaction with a system over a variety of different situations, yielding a set of context specific classifiers. During the training phase, it is assumed that the IDS observes the users behavior in all situations without contamination by masqueraders. When deployed, the IDS actively switches between different situations and classifies the observed behavior based on the context. The different situations are called the **modes** of the IDS. The actual semantics of modes will depend on the system.

For example, if the system being protected is a database system, then the a sample set of modes could be the ones listed in Table 1. For each user, the IDS learns a separate classifier for each mode. When deployed, the IDS does not affect the system's normal operation. However, it does affect the way the user interacts with the system – the output seen by the user depends on the current mode of the IDS. The most common mode would be Mode 1, where the IDS does not alter the output. Occasionally, the IDS will change the mode to one of the other modes based on a mode selection policy that will be discussed later. If the current mode is Mode 2, then the IDS makes it appear as if there is a deadlock in the object that the user is requesting for. If the current mode is Mode 3, the

IDS displays an error that says that an integrity constraint has been violated. For each mode, the IDS observes the user's reaction in response and performs its classification task based on the reaction. If the masquerader has seen only the normal behavior of the users, then the responses of the masquerader to other modes will be detected by the respective classifiers.

Table 1. Example set of modes for an interactive database system. The proactive IDS in this system learns how the users behave in response to the introduction of these situations. Note that the IDS does not change the way the database works. It only impacts user's observations. When deployed, the IDS occasionally introduces these situations to verify that the user's responses to the changes are consistent with behavior that has been learned.

Mode	**Situation represented**
Mode 1	Normal usage
Mode 2	Simulate "Deadlock found"
Mode 3	Simulate "Integrity constraint violation"

There are two advantages of a proactive IDS over a passive one –

1. The user's behavior becomes more distinctive in each specific situation. This makes the task of classification easier.
2. If the modes are chosen such that reaction to them is at an instinctive level rather than at a conscious level, then learning to impersonate the legitimate user in different modes becomes very difficult.

In general, let the set of modes of an IDS be denoted by

$$\mathcal{M} = \{M_1, M_2, \cdots, M_{|\mathcal{M}|}\}$$

The mode M_1 will be designated as the normal mode, where the IDS will not transform the output seen by the users. In all other modes, the IDS will transform the output in some way. Since all modes other than M_1 are disruptive, we may wish to quantify and limit the disruption in some way. Let $d(M_j)$ represent a non-negative real number representing the disruption caused the IDS being in mode M_j. Note that $d(M_1) = 0$ because that represents the normal mode. For all other modes, the disruption need not be zero.

As with the user actions, let the i^{th} mode within a block be m_i and let all the modes in a block be denoted by $m_1^{\mathcal{N}}$. We extend the definition of the classifier defined in 2 to include the modes. The classifier has the same form as defined earlier; it has additional input to make its decision. The classifier $\mathcal{C}_{Alice}(x_1^{\mathcal{N}}, m_1^{\mathcal{N}})$ is redefined to report an intrusion if –

$$\sum_{i=1}^{\mathcal{N}} -\log P(x_i, m_i | x_1^{i-1}, m_1^{i-1}, Alice) > \theta \tag{3}$$

Since the IDS proactively chooses modes after being deployed, we must specify a procedure for the selection of modes. Let the policy for picking modes be called $ModeSelectionPolicy$. We will discuss different mode selection policies in Sect. 4.

A generic Proactive IDS is represented in pseucode in Algorithm 1.

Algorithm 1 Proactive Intrusion Detection System

```
1: Set m_1 = M_1
2: while TRUE do
3:    Observe x_i
4:    if C(x_1^N, m_1^N) == 1 then
5:       Intrusion
6:    else
7:       Choose the next mode using ModeSelectionPolicy
8:    end if
9: end while
```

Clearly, from Algorithm 1, we can see that the IDS is completely defined by specifying the classifier $\mathcal{C}$ and $ModeSelectionPolicy$. These are two independent aspects of the proactive IDS. The choice of classifiers is an extensively studied problem in the field of intrusion detection. From the point of view of machine learning, classification of sequences is a well understood problem. Instead of explicitly specifying the type of classifier, we focus on the $ModeSelectionPolicy$ in this paper. When we describe our experiments in 5, we give brief information about the specific classifiers used.

4 Choice of Modes

In this section, we examine different policies for the selection of modes.

4.1 Passive IDS

It is useful to note that a **passive IDS** (that is, an anomaly detector that does not use different modes) can be thought of as a proactive IDS where $m_i = 1$ for all i. That is, a passive IDS is one that always operates in the normal mode. This view of a passive IDS shows that the set of proactive intrusion detection systems is a strict superset of the set of passive intrusion detection systems.

4.2 Random Proactive IDS

Another policy for the selection of modes is to pick each mode randomly and independently. While this may seem naïve, experiments show that this does indeed perform quite well.

4.3 Constrained Random Proactive IDS

The random proactive IDS does not place any constraints on the disruption caused by the modes. We may to limit the total disruption caused by the IDS per block to a value – D_{limit}. Extending the random proactive IDS, a **constrained random proactive IDS** picks modes randomly while ensuring that the total disruption for the whole block is not more than D_{limit}.

4.4 Decision Theoretic Proactive IDS

While the random proactive systems do offer improvement over the passive IDS, we would like to control the choice of modes in a better way. Here, we present a decision theoretic policy for the choice of modes. This approach is based on the worst-case assumption that the IDS and the user have an adversarial relationship.

The implicit goal of the intruder *Bob* is to make the classifier of the IDS believe that he is *Alice*. In our current setting, his goal will be to make the left hand side of Equation 3 less than the threshold θ. In further discussion, the left hand side of Equation 3 will be referred to as the *score* of the classifier.

The additive form of the score can be interpreted as follows – each term in the summation makes an incremental contribution of

$$-\log P(x_i, m_i | x_1^{i-1}, m_1^{i-1}, Alice)$$

to the final score. At the i^{th} step, the goal of *Bob* is to pick an action x_i such that its incremental contribution is minimized. The assumption of an adversarial relationship between the IDS and the user will imply that the proactive IDS should pick mode m_i such that the mode's incremental contribution to the final score is maximized for any action of *Bob*'s. This gives us the following **decision theoretic policy** to pick modes –

$$m_i = arg_M \left(\min_X \max_M -\log P(x_i, m_i | x_1^{i-1}, m_1^{i-1}, Alice) \right) \tag{4}$$

4.5 Mixed Strategy Proactive IDS

The adversarial notion presented above can be analyzed as a zero sum game between the IDS and the user. In the decision theoretic analysis presented above, the IDS picks a single mode whose incremental contribution to the final score is maximized for all possible actions of the user. Instead of picking a single mode, we can derive a probability distribution over the modes such that the *expected incremental contribution* to the final score is maximized for all possible user actions. If $p = (p_1, p_2, \cdots, p_{|\mathcal{M}|})$ is a probability distribution over the modes where p_j is the probability that the IDS will pick mode M_j, then we wish the following to hold –

$$p = \min_X \max_p E_p \left[-\log \left(P(x_i, m_i | x_1^{i-1}, m_1^{i-1}, Alice) \right) \right] \tag{5}$$

where E_p denotes the expectation over the distribution p. Once we have the optimal p, the next mode is picked by drawing from the distribution.

Linear programming is a common technique to solve zero-sum games (see, for example, [9]). Computing the optimal set of probabilities p can be done by converting the game to a linear program.

Let the actual expected incremental contribution of x_i and m_i be λ. The IDS tries to maximize this value, giving us the objective function of the linear program –

$$\max \lambda \tag{6}$$

Additionally, the user wants this value to be less or equal to the expected incremental contribution for any action X_k that he performs (which is equivalent to minimizing the quantity for all actions). This gives us the first set of constraints for the the linear program –

$$\lambda \leq E_p \left[-\log \left(P(x_i = X_k, m_i | x_1^{i-1}, m_1^{i-1}, Alice) \right) \right] ; \forall X_k \in \mathcal{X} \tag{7}$$

The final constraints for the linear program ensure that the search space is the probability simplex –

$$\sum_{j=1}^{|\mathcal{M}|} p_j = 1 \tag{8}$$

$$p_j \geq 0 \; ; \forall j \tag{9}$$

The Equations 6, 7, 8, 9 together form the linear program whose solution is the optimal probability distribution over the modes. The linear program will always converge to an optimal solution because every matrix game with two players has an optimal solution. The IDS then draws a mode using this probability distribution – called the mixed strategy – and switches to that mode. This gives us the **mixed strategy policy**.

It must be noted that in this analysis, there is a departure from traditional game theoretic analysis for mixed strategies. Traditionally, it is assumed that both the players make their move simultaneously. Clearly this is not the case here. The IDS picks its mode first and the user performs an action based on the move. In a perfectly adversarial situation, the user will not think in terms of probabilities over actions because once he knows the mode, he can pick a single action that minimizes the score (in other words, he can use a pure strategy). Yet, it is shown in the experiments that the mixed strategy is among the best performing policies. This is because, in reality, the user is not really perfectly adversarial. The goal of the user, in addition to thwarting the IDS, is also to perform some malicious activity. Hence, the user will not necessarily use a pure strategy all the time.

4.6 Constrained Mixed Strategy Proactive IDS

We may wish to place constraints on the disruption caused by the Mixed Random IDS. In the linear program representation in Sect. 4.5, this can be done by

limiting the expected disruption to D_{limit}. The new constraint added to the linear program is

$$E_p\left[d(M_j)\right] \leq D_{limit} \tag{10}$$

Adding this constraint will will not make the original linear program infeasible because $d(M_1) = 0$ and hence, picking M_1 with probability 1 will be a feasible solution. Constrained games and their relation to linear programming is discussed in [10]. The IDS that uses this constrained game to pick its mode will be referred to as the **constrained mixed strategy proactive IDS.**

5 Experiments

Two sets of experiments were conducted. In the first one, the different mode selection policies were compared with in terms of error in identifying intrusions and disruption. In the second experiment, we adapted the idea of a proactive IDS to detect masqueraders in a behavioral biometrics domain to show that it is indeed a practical approach.

5.1 Comparison of Mode Selection Policies

The setup of this experiment was similar to that of [3]. The Schonlau dataset provides a sequence of commands used by a set of 50 real users. However, this data cannot be used to evaluate a proactive IDS because the training data in terms of different modes is not known.

For the experiments, we assumed that the set of user actions was of size 15 and we defined 6 modes of operation for the IDS. Since the algorithms for the IDS do not depend on the semantics of the modes, we did not define the semantics of the modes and instead refered to them as modes $M_1, M_2, \cdots, M_6$. As earlier, M_1 is assumed to be the normal mode, which means that its disruption is zero. The other modes were assigned disruptions 1.

$$\begin{aligned} d(M_1) &= 0 \\ d(M_i) &= 1 \; ; 2 \leq i \leq 6 \end{aligned}$$

Since we could not use the data of real users, we used simulated users to test the systems. Following [3] where a Markov model was among the best models for users, users are simulated using a Markov chain. In other words, the users were completely represented by a probability of the next action given the current action and the current mode. Twenty such users were generated randomly. For each user, a session was defined as 50 non-overlapping blocks, with 100 commands in each block. Each block was defined such that all the commands in a block were generated by the same user – either the masquerader or the session's legitimate user. This allowed us to perform classification on a per-block basis.

Given the definition of d, the maximum possible disruption per block is 100 (when all the modes in that block are disruptive). For the mode selection policies that constrain the disruption, we set the value of D_{limit} to 50 per block, (that is, 2500 for the entire session of 50 blocks).

Masqueraders. The smart masquerader *Bob* will be able to pretend to be *Alice* in a normal situation. When exceptional situations are encountered, *Bob* will revert to his own behavior. This is captured by having the masquerader learn the behavior of the user in normal mode using the same data that was used to train the IDS. The masquerader's learning protocol is the same as that of the IDS, except that it is restricted to the normal mode. In all other modes, the masquerader will be modeled by a different markov chain with its own parameters.

Masqueraders were inserted into user's session in a similar fashion as [3]: If the previous block was not a masquerade, then the current block is not a masquerade with probability p_u. If the previous block is a masquerade, then the current block is a masquerade with probability p_m. In our experiments, we set the values of both p_m and p_u to 0.8.

Proactive IDS. All the IDSs described in Sect. 4 (including the passive IDS) differ only in the way they choose modes. Hence, in order to provide fair comparison, the IDS models and the classifier were kept the same for all the IDSs.

The IDS's internal model of the users was a Markov model. For each user, during training, the IDS computes $P(X|M, X_{previous}, user)$. The classification is done by computing the probability of the actions in the block given the modes. That is, the block is declared as an intrusion if

$$-\log\left(P(x_1^{\mathcal{N}}, m_1^{\mathcal{N}}|user)\right) \geq \theta$$

where the threshold θ was tuned to get equal error rate for the passive IDS when not facing masquerade attacks.

Results. For each IDS, we measure the error in predicting the current user – an IDS is said to make an error both when it predicts that a masquerader is a legitimate user and vice versa. In addition, we also measured the total disruption for the entire user session for each IDS. For each proactive IDS, we defined a metric called *Effectiveness* that captures the need for high accuracy and low disruption. The effectiveness of a proactive IDS is defined as follows –

$$Effectiveness = \frac{1 - Error}{Disruption}$$

The errors in masquerade detection, disruptions and the effectiveness for each five algorithm are shown in Table 2.

An immediate observation is that all the proactive IDSs perform better than the passive IDS. The best performing proactive systems are the random proactive IDS and the mixed strategy proactive IDS. However, looking at the corresponding disruptions shows that the most desirable IDS in terms of disruption is the passive IDS; among the proactive intrusion detection systems, the average disruption per user is best for the constrained mixed strategy proactive IDS. In this case, the total disruption is very close to the expected disruption (2500). The other IDS with constrained disruption – the constrained random IDS – does

not perform as well as the constrained mixed strategy IDS. This is shown more clearly in terms of the effectiveness, where the constrained mixed strategy IDS is the most effective one.

Table 2. Evaluation of different mode selection policies. All the proactive policies have a lower error than the passive IDS. Random selection of modes is the best in terms of average error followed by the mixed strategy IDS. However, constraining the error for the random mode selection policy has makes it perform badly. In comparison, the constrained mixed strategy has a very low error rate when the disruption is constrained to a similar value. Effectiveness, which captures the need for high accuracy *and* low disruption, shows that the most effective IDS is the constrained mixed strategy IDS.

IDS Type	Average Error	Average Disruption	Effectiveness $\times 10^4$
Passive	0.427	0	–
Random Proactive	0.019	4157.3	2.360
Decision Theoretic	0.049	4215.3	2.256
Mixed Strategy	0.027	4168.6	2.334
Constrained Mixed Strategy	0.107	2471.2	3.614
Constrained Random	0.303	2500.0	2.788

In order to caliberate the constrained mixed strategy IDS, we looked at the errors in detection as a function of the parameter $D_{limit}/\mathcal{N}$ - the constraint on the disruption per mode. A plot of the error and the actual disruption as functions of the expected disruption is shown in Fig. 1. As the limit on the disruption becomes lower, the error rate of the IDS gets closer to 50%. That is, if no disruptions are allowed, the proactive IDS will be forced to remain in mode M_1 and the error rate will be similar to that of a passive IDS. As expected, if there is no limit on the disruption, then the error rate is very small just like that of the mixed strategy proactive IDS.

The IDS needs to find the right balance between inactivity and disruptiveness. The expected disruption could be thought of as a parameter for creating an IDS which is a combination of the passive IDS and the mixed strategy proactive IDS. Interestingly, the even if the maximum possible disruption is allowed, the actual disruption (as seen in Fig. 1) is less than the allowed amount.

5.2 Behavioral Biometrics

Behavioral biometric systems identify or authenticate users based on how their current usage of a computer compares to profiles learned from their past behavior. Recent work ([11–13]) has shown the success of systems derived from keystroke and mouse movement dynamics, but has not investigated their success in the face of masquerade attacks. We have applied the idea of proactive intrusion detection to detect masquerade attacks in this domain. As a baseline for comparison, we implemented the system described in [11] which distinguishes users based on their mouse movement.

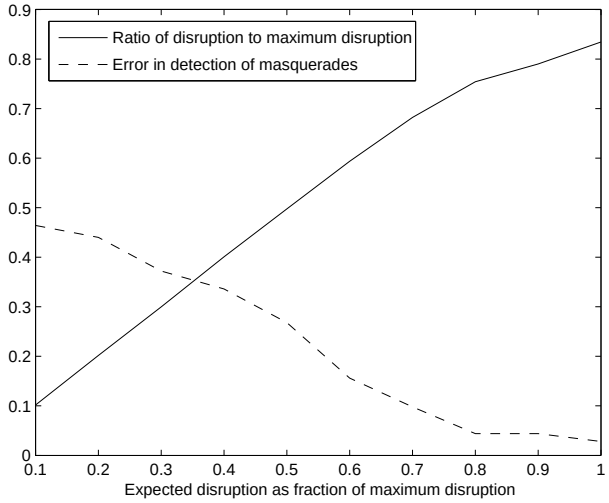


Fig. 1. Error and actual disruption of a constrained mixed strategy proactive IDS as a function of $D_{limit}/\mathcal{N}$ – the constraint on the disruption per mode. The error in detecting masquerades (dashed line) is very close to zero if we allow high disruption and the IDS's performance with very low disruption is similar to that of a passive IDS. This allows us to think of D_{Exp} as a parameter for tuning the constrained proactive IDS as a mixture of a passive and a proactive IDS based on the amount of disruption we can tolerate. Interestingly, the actual disruption (solid line) tapers off for higher values of $D_{limit}/\mathcal{N}$ which shows that even if the IDS is allowed to be disruptive at every step, it does not do so.

System and User Description. Mouse movement data was collected from 10 users interacting with 3 applications {same-gnome, iagno, gnomine} on a Linux workstation. These applications were chosen in part because the mouse is their primary source of input. Although we experimented with segmenting the data by application, we found that all data sets performed similarly and we only report results for an aggregate data set consisting of 1555 strokes per user.

We implemented two intrusion detection systems – the passive IDS and the random proactive IDS. For both of them, experiment we used 90% of the data for training and 10% for testing. During training, a user's mouse movements, {x-coord, y-coord, button-state}, and corresponding timestamp are logged and segmented into *strokes* where a *stroke* is the data collected between two successive mouse clicks. From each stroke an example is created by extracting spatial and temporal features from the original points and from a smoothed representation. Example features include statistics such as the average and standard deviation of the horizontal, vertical and angular velocities, as well as values such as the jitter (ratio between the original path length and the smoothed path length). For a complete description of all features see [11].

Examples formed from the strokes for each user are passed to a learning algorithm to build a classifier for distinguishing strokes generated by user u from strokes generated by others. One classifier is trained per user. While not strictly necessary, in our implementation as well as in the original, a supervised learning algorithm is used such that the strokes from a particular user form the set of negative examples for his classifier and the strokes collected from other users form the set of positive examples, that is intrusions. We experimented with a number of different supervised learning algorithms and found that Support Vector Machines (SVM) [14, 15] using a standard Gaussian kernel and tuned according to the procedure described in [16] performed best.

During evaluation, the output of the classifier when presented a single stroke, is a value between 0 and 1 that can be viewed as the confidence of the classifier that the example *was not* generated by the legitimate user. Given individual classifications for a sequence of strokes the entire sequence is classified as genuine or not by adding the individual scores and comparing their sum to a threshold tuned to achieve equal error rate.

Masquerade Attacks. We assumed that the attacker's goal is to move the mouse from point a to point b and execute a click. To demonstrate a successful masquerade attack against the baseline system, we made the following assumptions –

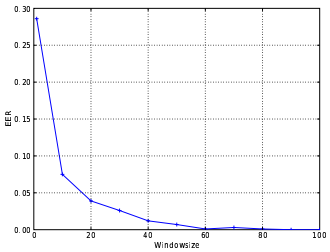
1. The attacker has access to some fraction of the data used to train the system.
2. The attacker knows the learning algorithm used by the IDS and how to extract features from the data.
3. The attacker has the ability to manipulate mouse events in such a way that it can trap and transform strokes before they reach the IDS.

Given these assumptions, we implemented an masquerader script that accepts a classifier and target stroke as input, and outputs a set of mouse movements with the same start and end points as the target but that passes undetected by the specified classifier. Thus, the masquerader is able to transform arbitrary strokes into successful attacks. It does so using a genetic algorithm where the specified classifier is used as a fitness function for potential solutions.

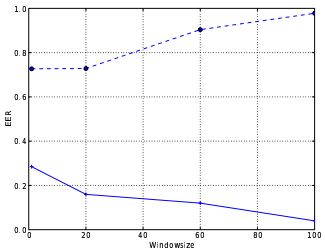
Baseline Passive IDS. First, we replicated the experiments reported in [11] by using other users' data as attack sequences, without using the masquerader script. The results summarized in Fig. 2(a) are consistent with those reported previously [11]. The average single-stroke error rate for the baseline system was 29%, but the error rate drops quickly as longer sequences of strokes are considered reaching 0.7% on sequences of length 50 and 0% on sequences of length 100.

We then implemented masquerade attacks on the IDS by generating attack sequences, using as input, classifiers trained on 25% of the available training data. Target strokes were taken directly from the raw data of other users. The results

summarized in Fig. 2(b) indicate that while the system is able to detect unmodified attacks (strokes taken straight from other users) it fails to detect those same strokes once they've been transformed by the attacker even though the attacker has only 25% of the training data at its disposal. The trend is only strengthened when attacks are mounted with increasing amounts of training data. As far as we are aware this is first time a masquerade attack has been demonstrated against a behavioral biometrics system, and the first time a masquerade attack has been demonstrated where the action space (in this case mouse movements) is distinct from the feature space (that is, there is no one-to-one mapping from examples to strokes).



(a) Error rate of the baseline system for increasing window lengths. As the sequence length being classified increases the error rate of the system approaches 0%.



(b) Error rate of the baseline system for increasing window lengths under masquerade attack mounted with 25% of training data. While the system is able to detect untransformed strokes of other users (solid line) its error rate approaches 100% when detecting strokes transformed by the attacker (dashed line).

Fig. 2. Evaluation of the passive baseline IDS

Proactive Masquerade Detection. We extended the baseline system to incorporate different modes of interaction during training and testing. We introduced the following six modes and learned a set of classifiers per user (one classifier per mode) –

1. Normal system interaction (no change)
2. Dropping short sequences of mouse movements at random.
3. Introducing a delay between when mouse movements are made and when they appear.
4. Introducing a random amount of jitter to each movement.
5. Slowing down the pointer by reducing the amount of screen movement corresponding to a given physical movement.

6. Speeding up the pointer by increasing the amount of screen movement corresponding to a given physical movement.

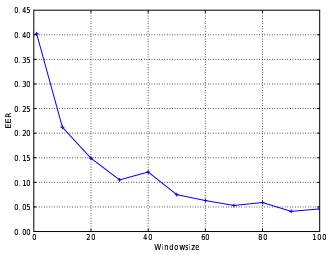
To evaluate the extended system, we collected roughly one hour of mouse movement data from each of 6 users. During each session, modes were chosen at random by the IDS such that for each user we collected roughly 200 strokes per mode. During testing the random mode selection policy was used.

In our experiments, we explored two different methods for learning mode-specific classifiers. The two methods differ in how the sets of positive and negative examples for learning each ⟨user,mode⟩ classifier are formed. In the first method, "user-vs-user", each ⟨user,mode⟩ classifier is trained such that negative examples are drawn from strokes collected from the given user u under the specified mode m, and positive examples are drawn from strokes collected from other users $v \neq u$ under the same mode m. In the second method, "mode-vs-mode", each ⟨user,mode⟩ classifier is trained such that negative examples are once again drawn from strokes collected from the given user u under the specified mode m, but in contrast to the previous method, positive examples are drawn from strokes collected from all other ⟨user=v,mode=n⟩ pairs where either $v \neq u$ or $n \neq m$ or both.

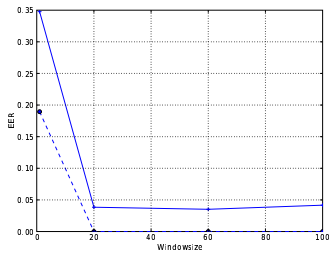
The two approaches for learning mode-specific classifiers, "user-vs-user" and "mode-vs-mode", behave quite differently when applied to natural sequences which have not been manipulated by an attacker. While "mode-vs-mode" classifiers do well in distinguishing between different ⟨user,mode⟩ pairs when the modes are different, they are bad at distinguishing between users under the same mode. On the other hand, while "user-vs-user" classifiers do well in distinguishing between different ⟨user,mode⟩ pairs when the modes are the same, they are in general worse at distinguishing between a single user under different modes.

The opposite biases of the two types of classifiers led us to consider combining them to boost the performance of the system. Finding "mode-vs-mode" classifiers biased towards false negatives we combined the classifiers sequentially: given a stroke we first use the appropriate "mode-vs-mode" classifier to determine if the stroke has been generated from the current mode. If the classifier raises an alarm (a positive classification) the result is passed on for use in sequence classification. If an alarm is not raised the stroke is passed to the appropriate "user-vs-user" classifier to determine if the stroke has been generated by the specified user, the result of this classification is then passed on for use in sequence classification. The results are summarized in Fig. 3(a). The overall effect of using two classifiers in sequence is to force a given example to fit both the current mode and the current user.

To evaluate our proactive detection scheme against masquerade attacks we ran the following experiment. The attacker proceeds as before but is ignorant of the mode since this is only known to the proactive IDS. Because of this it learns a single "average" classifier per user and uses that as the input to the attacker script. When the IDS evaluates each stroke it does so on the basis of the current mode of interaction (which it sets). Thus while the attacker generates strokes



(a) Evaluation of the proactive detection system. Performance is consistent with the baseline- as the sequence length being classified increases the error rate of the system approaches 5%.



(b) Evaluation of the proactive detection system under masquerade attack mounted with 100% of training data. Unlike the baseline system, the proactive IDS is able to correctly identify strokes both before (solid line) and after (dashed line) they are transformed by the attacker (achieving 0% error in the second case).

Fig. 3. Evaluation of a proactive IDS

that pass the "average" classifier these same strokes fail to pass the mode-specific classifiers used by the proactive detection scheme. Clearly, as summarized in Fig. 3(b), the attacker no longer succeeds with any sequence length even though the complete training set is used.

6 Application Domains

The proactive IDS, in the form described in this paper, is not specific to any one domain. In section 5, we show an application of proactive intrusion detection to behavioral biometrics. In this section, we provide other examples showing how the model can be used in different domains. First, we need to outline the properties of a domain to be amenable to the proactive IDS framework.

In general, we require that the domain has two properties:

1. The domain provides a large degree of interactivity, as indicated by the set of allowed user actions $\mathcal{X}$ and the dependence of the user action X_i on the observations. If the set of user actions is very small, it is not possible to observe enough variation between users, irrespective of the IDS actions. Additionally, if the user's actions don't depend on the observation, then, the assumption that the changing the system to create a different system output will change the user's behavior will not be valid any longer.
2. The domain must have a certain amount of inherent randomness or irregularity. The change of system state must not disrupt the normal activity of the system to a very large extent.

Some domains which satisfy these requirements are listed below.

User interaction with a command line shell This is one of the best studied domains in terms of intrusion detection – [3] and many others discuss this domain. The actions are the inputs by the user to the command line and the observations are the corresponding shell response. A user profile is usually learned by collecting usage statistics over a period of time.

The domain is quite interactive and slight changes in the behavior of the system could be tolerated. Examples of IDS actions could include changing command line internals or disabling certain programs.

Transaction based Systems A transaction based system consists of a number of interconnected databases that can be queried and augmented by the user. The user actions are represented as SQL and the database responses correspond to the observations. A proactive IDS has several possibilities to change the mode of interaction. For example, it can deny access to certain tables or joins.

Online multi-player games and fantasy sports Multi-player online games are fast growing to be the among the most popular online activities. Users assume avatars online to interact with other users. Intrusion and collusion prevention are growing concerns in these games as they become more popular. Collusion is a scenario where more than one person get together to play the game as one avatar. An intrusion (in the context of this paper) is a scenario where a player's account is compromised by another player who pretends to be the first person in the online environment. These and other security concerns in online games are described in [17].

Fantasy sports are online games where people compete against each other by building teams based on real players or teams of a professional sport. Here, each user controls a team based on real world statistics and trends. A masquerade would be a scenario where one player's team is manipulated by another unauthorized user.

Both these domains have a high level of interactivity. Additionally, there is sufficient randomness in the games for the IDS to be proactive without causing too much disruption. In the online games, the IDS could change the environment of the avatar or the characteristics of the avatar itself. In the fantasy sports, the IDS could tweak the statistics or trends so that the players are forced to behave differently. Note that these changes could be made on a per-user basis without reflecting it to the global environment.

7 Conclusion

In this paper, we introduced the novel idea of proactive intrusion detection to detect insider attacks. The key intuition is that while average behavior could be masqueraded, exceptional situations force people to revert to their own behavior and this idea can be used to detect intrusions. Another idea developed in this paper is that the intrusion detection system, instead of just observing

the user's actions, participates in influencing them with its own set of actions called modes. This broadens the capability of the intrusion detection system. These ideas are presented without special emphasis on any specific domain or specific classifiers because they can be used with different classifiers in diverse domains. We experimentally showed that proactive intrusion detection does detect masqueraders and we presented its real life utility by applying the ideas to behavioral biometrics.

References

1. Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, Oakland, CA, IEEE Computer Society Press (1994) 202–212
2. Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A.: A sense of self for Unix processes. In: Proceedinges of the 1996 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press (1996) 120–128
3. Schonlau, M., DuMouchel, W., Ju, W.H., Karr, A.F., Theus, M., Vardi, Y.: Computer intrusion: Detecting masquerades. Statistical Science **16**(1) (2001) 58–74
4. Maxion, R., Townsend, T.: Masquerade detection using truncated command lines. In: Proceeedings of the International Conference on Dependable Systems and Networks, IEEE Computer Society Press (2002) 219–228
5. Yung, K.H.: Using feedback to improve masquerade detection. In: ACNS. (2003) 48–62
6. Niculescu-Mizil, A., Caruana, R.: Predicting good probabilities with supervised learning. In: ICML '05: Proceedings of the 22nd international conference on Machine learning, New York, NY, USA, ACM Press (2005) 625–632
7. Ju, W.H., Vardi, Y.: A hybrid high-order Markov chain model for computer intrusion detection. Journal of Computational and Graphical Statistics **10**(2) (2001)
8. DuMouchel, W.: Computer intrusion detection based on bayes factors for comparing command transition probabilities. Technical Report TR91, National Institute of Statistical Sciences (NISS) (1999)
9. Vaserstein, L., Byrne, C.: Introduction to Linear Programming. Pearson Education, Inc (2002)
10. Charnes, A.: Constrained games and linear programming. Proceedings of the National Academy of Sciences of the United States of America **39**(7) (July 1953) 639–641
11. Gamboa, H., Fred, A.: An identity authentication system based on human computer interaction behaviour. In: Pattern Recognition in Information Systems. (2003)
12. Pusara, M., Brodley, C.: User re-authentication via mouse movements. In: ACM workshop on Visualization and Data Mining for Computer Security. (2004)
13. Gunetti, D., Picardi, C.: Keystroke analysis of free text. ACM Transactions on Information and System Security (2005)
14. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning (1995)
15. Burges, C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery (1998)
16. Hsu, C., Chang, C., Lin, C.: A practical guide to support vector classification. Technical report (2006)

17. Yee, G., Korba, L., Song, R., Chen, Y.C.: Towards designing secure online games. In: AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06), Washington, DC, USA, IEEE Computer Society (2006) 44–48