An Adaptive Detection of Anomalies in User's Behavior

Artem M. Sokolov

International Research and Training Center of Informational Technologies and Systems

Pr. Acad. Glushkova 40

Kiev 03680

Ukraine

sokolov@ukr.net

Abstract- **We propose an adaptive approach to modeling user's behavior in computer anomaly detection systems. As a base model, Markov chains with variable memory length are used. An adaptive version of the algorithm constructing a model that takes into account changes in user's behavior is introduced. Experimental testing of the proposed approach is also provided.**

I Introduction

Over the last decade it became evident that even advanced security systems are not always capable of protecting computer systems from intrusions. The majority of systems make use of common security mechanisms: identification and authentication, access restriction, and cryptographic mechanisms. Drawbacks of traditional approaches include: vulnerability from own malevolent users, an often impossible clear distinction between "inside" and "outside" users, comparative ease of guessing semantic passwords, hampering user's work due to imposed access restrictions (e.g., in e-business). Hence, a demand has emerged for mechanisms that can resist attacks even if standard protection systems have been already passed through.

Intrusion detection systems are intended to meet that demand. Usually, they are classified into misuse detection and anomaly detection systems. The former are based on comparing patterns of known attacks with log traces. Such systems are effective dealing with known attacks, however, in case of an unknown attack they usually fail. Therefore, a large, continuously updated database of attack patterns and their variations is needed. Anomaly detection systems are more flexible when dealing with unknown attacks. Their basic assumption is that intruder's actions necessarily differ from the behavior of a legitimate user, i.e., they are an anomaly. At the first stage of their functioning, a representation of normal activity is built. Building patterns of abnormal behavior is difficult because of the rarity of labeled intrusive training data. Therefore, training should often be carried out only on positive examples, which is much more difficult. Two types of mistakes are possible: normal behavior is wrongly taken for ill-intentioned (false positive) or an attempt of an illegal penetration into a system is taken for normal activity (false negative).

In this work, we propose an approach to adaptive modeling of user's behavior using Markov chains with variable memory length [1]. In section II, we give some introductory considerations that lead to the selection of the base model. Section III briefly describes the base model from [1]. Section IV provides our modified adaptive algorithm. Experimental testing and conclusions are given in sections V and VI.

II Modeling user's behavior

Starting from the pioneering work in the field of anomaly detection [2], many approaches to the problem have been proposed. Almost all of them can be divided into four categories: instance-based learning [3], frequency-based learning [2], neural-network approaches [4,5], and finite automata approaches [6,7,8].

The majority of data available from audit systems are of sequential and non-deterministic nature. And for many data sources, it is typical that the probability of the next symbol depends on previous symbols, and frequently only on few of them. It suggests modeling such data with Markov chains [2,9,10,11] or with Hidden Markov Models (HMM) [6]. Though these statistical models can model a variety of sequences, they have their drawbacks: the number of states of Markov chains grows exponentially with the increase of their order. Hence, only models with very small order are of practical value, and they may give bad approximations. As to HMM, there are theoretical results concerning the difficulty of their learning (see [1] for references and details).

Another consideration for not using HMM is that unlike the situation in pattern recognition tasks, the observed audit events are usually not distorted. In case of computer audit events, we usually observe events their creator meant to create. Hence, it is not obligatory to introduce unobservable states having a meaning of "what really occurred", as in HMM [6], or to use advanced structure-processing methods that handle sequence components possessing a gradual degree of similarity [12].

Moreover, for sequences of user commands, the probability of next command is usually determined by a variable length context. Therefore, there is no need to take into account all possible previous contexts, but only those on which next

commands really depend. For this work we chose a model that can be thought of as Markov chains with variable memory length [1].

The majority of approaches to modeling of audit sequences described in literature are non-adaptive. It is often offered to periodically retrain models for adaptation to varying behavior. This delayed tuning results in performance and reliability decrease, because the model becomes inaccurate as time passes after retraining. The only work known to us where an adaptation method similar to ours was proposed is [9]. However, a Markov chain of the first order was used there and dependencies on contexts longer than one command could not be considered.

III BASE MODEL

Following [1], we consider a finite alphabet (commands, in our case) Σ, with the set of words over Σ denoted by Σ^*, finite set of automaton states Q, where every state $q \in Q$ has a label $s \in \Sigma^*$. Probabilistic Suffix Automaton (PSA) is defined as a 5-tuple $< Q, \Sigma, \tau, \gamma, \pi >$, where $\tau : Q \times \Sigma \to Q$ is the transition function, $\gamma : Q \times \Sigma \to [0,1]$ is the next symbol probability function, and $\pi : Q \to [0,1]$ is the initial state probability distribution over starting states.

For two states $q^1, q^2 \in Q$, and for each symbol $\sigma \in \Sigma$, if $\tau(q^1, \sigma) = q^2$, and q^1 has s^1 as its label, then we require that q^2 has such a label s^2 that is a suffix of string $s^1\sigma$.

The task of building an automaton using examples obtained from some PSA M is posed in [1], and a special subclass of stochastic machines, Prediction Suffix Trees (PST), is chosen as the hypothesis class. A PST T, over Σ, is a tree of degree $|\Sigma|$, where every edge is labeled by a symbol $\sigma \in \Sigma$. There is a pair (s, γ_s) associated with every node, where s is a string, associated with the walk beginning from that node and ending in the root of the tree, and $\gamma_s : \Sigma \to [0,1]$ is the next symbol probability distribution associated with s. For each s that labels a node $\sum_{\sigma \in \Sigma} \gamma_s(\sigma) = 1$. The probability that the PST will generate sequence $\overline{r} = r_1 r_2 \dots r_N$ is

$$P_T^N(\overline{r}) = \prod_{i=1}^{N} \gamma_{s^{i-1}}(r_i), \qquad (1)$$

where s^i is the string labeling the node we reach by performing a walk that corresponds to the sequence $r_j r_{j-1} \dots r_1$.

To measure the quality of approximation of automaton M by some PST T, it is required that the so-called Kullback-Leibler distance between probability distributions P_M and P_T the automaton and the tree induce on strings of length N is not larger then $\varepsilon > 0$ with sufficiently large probability (ε-good hypothesis).

In [1], empirical probabilities $\tilde{P}(s)$ and $\tilde{P}(\sigma \mid s)$ are defined using examples generated by M. If N is the length of string r, L is the maximal length of nodes' labels (bound on the depth of PST), then, defining $\chi_j(s)$ as an indicator (1/0) of s coinciding with a substring r starting from the position $j - \mid s \mid +1$, we put:

$$P'(s) = \frac{1}{N - L + 1} \sum_{j=L}^{N-1} \chi_j(s),$$

$$P'(\sigma \mid s) = \sum_{j=L}^{N-1} \chi_{j+1}(s\sigma) / \sum_{j=L}^{N-1} \chi_j(s). \qquad (2)$$

For some m strings, each of length $N \geq L + 1$, we then have

$$\tilde{P}(s) = \frac{1}{m(N - L + 1)} \sum_{i=1}^{m} \sum_{j=L}^{N-1} \chi_j(s),$$

$$\tilde{P}(\sigma \mid s) = \sum_{i=1}^{m} \sum_{j=L}^{N-1} \chi_{j+1}(s\sigma) / \sum_{i=1}^{m} \sum_{j=L}^{N-1} \chi_j(s). \qquad (3)$$

The algorithm of learning PST starts from an empty tree with one root node labeled by an empty string. Then, a node with some label s is added to it, if $\tilde{P}(s)$ is not negligible and if for some symbol σ the probability $\tilde{P}(\sigma \mid s)$ differs from the empirical probability $\tilde{P}(\sigma \mid suffix(s))$ of having it as the next symbol after $suffix(s)$. This means that s, but not $suffix(s)$, is then the context of σ, on which s depends. The algorithm ends if there are no leaves in the tree for which the above conditions are satisfied, or if it has reached the maximum bound on the depth of the tree L.

It is proved in [1] that, for any PSA M and parameters $\varepsilon > 0$ and $0 < \delta < 1$, the learning algorithm builds in polynomial time a PST $\hat{T}$ such that with the probability not less than $(1 - \delta)$, the tree $\hat{T}$ is an ε-good hypothesis for M.

IV MODIFICATION OF THE BASIC MODEL

A principal peculiarity of anomaly detection task is the necessity of adaptive model construction and it's updating. To supply the base model with adaptability, we modified the calculation of empirical probabilities (3) so that the contribution of earlier examples to cumulative empirical probability decreases at each step.

We fix learning coefficient $0 < \alpha < 1$, and put the empirical probability of a state s at time t to be

$$\tilde{P}_1(s) = P_1'(s),$$

$$\tilde{P}_t(s) = \alpha \tilde{P}_{t-1}(s) + (1 - \alpha) P_t'(s) \qquad (4)$$

where $P_t'(s)$ (2) is the empirical probability of the state in the string that arrived at the moment t. Coefficient α regulates "forgetting" speed: values close to zero will result in fast "forgetting", and values close to unity — in slower.

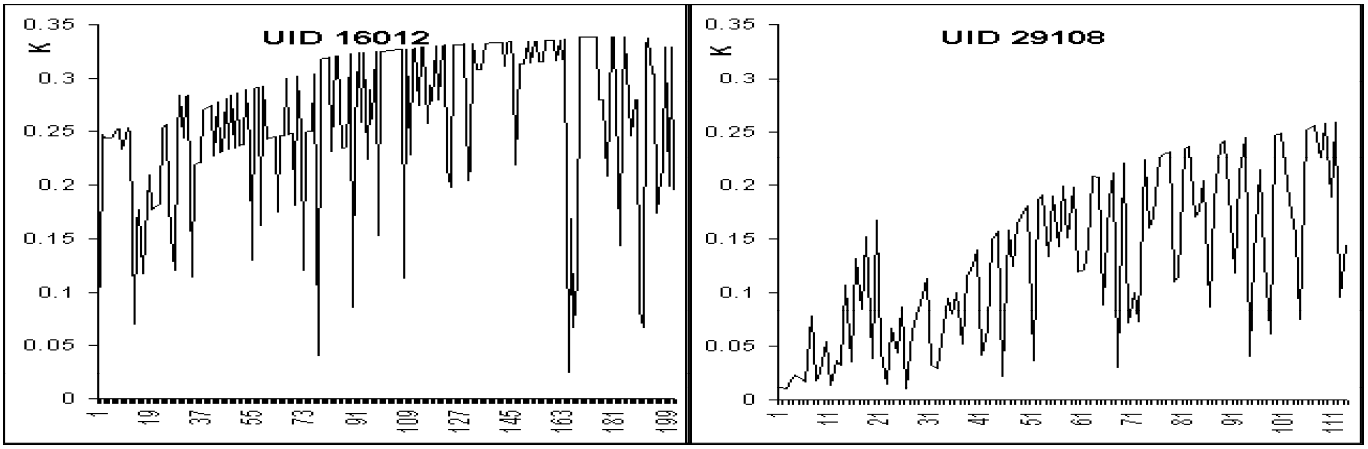

Fig. 1. Typical evolution of anomaly indicator K vs. number of session during the training period.

We obtain the following modified algorithm for adaptive reconstructing a PST under changing probabilities of states:

1. Get $\overline{T}_{t-1}$ from the previous time step, and renew values of empirical probabilities according to (4).
2. Recursively delete all leaves for which
$$\tilde{P}_t(s) < (1-\varepsilon_1)\varepsilon_0.$$
3. Initialize set $\overline{S}$:
$$\overline{S} = \{s \mid s \in \Sigma^*, suffix(s) \in L(\hat{T}_{t-1}), \tilde{P}_t(s) \geq (1-\varepsilon_1)\varepsilon_0\},$$
where $L(\hat{T}_{t-1})$ is the set of leaves of $\hat{T}_{t-1}$.
4. While $\overline{S} \neq \varnothing$ do: pick any $s \in \overline{S}$ and
(a) delete s from $\overline{S}$
(b) if there exists such $\sigma \in \Sigma$ that
$$\tilde{P}_t(\sigma \mid s) \geq (1+\varepsilon_2)\gamma_{\min} \text{ and } \frac{\tilde{P}_t(\sigma \mid s)}{\tilde{P}_t(\sigma \mid suffix(s))} > 1 + 3\varepsilon_2,$$
add to the tree a node that corresponds to s, and (possibly) all nodes on the way from the deepest node in $\hat{T}$, which is a suffix of s, to the node with label s.
(c) If $|s| < L$, then for each $\sigma' \in \Sigma$, if $\tilde{P}(\sigma' s) \geq (1-\varepsilon_1)\varepsilon_0$, add a string $\sigma' s$ to $\overline{S}$.

V Experiments

For training our model, real data obtained from 6 month logs of a UNIX university server were used. All processes which were started on behalf of users were logged by means of FreeBSD accounting mechanisms. Totally, we collected data from more than 800 users. From more than 900 unique process names (commands) that were used during the data collection period, we left only those whose overall use intensity exceeded 200 launches – 350 commands altogether. All other rare commands were replaced with the special command `RARE`. In addition, indicators `ALPHA` and `OMEGA` were inserted to mark the beginning and the end of each session. The resulting PSA (after conversion from PST) contained several thousand states.

Probability for each session was calculated using (1). To partially overcome its dependence on the session's length, the anomaly indicator $K = \sqrt[N]{P_T^N(r)}$ was used, where $P_T^N(r)$ is the probability of a given log session of length N. In Fig. 1, the typical evolution of the K parameter is depicted for two users.

A. Cross test

One of the possible ways of anomaly detection is using cross testing of log sessions. I.e., for the current session its probability is calculated as though it was generated not by the owner of the session, but by the PSTs of all other users as well. If the maximum probability is reached not on the real user's PST, an intrusion may be present.

We carried out an experiment to check whether this method works. In Fig. 2, averaged values of K are represented by gray scale – darker cells correspond to their larger values, lighter cells correspond to their smaller values. The diagonal values corresponding to the probability values of "own" sessions using "own" PSTs are approximately three times larger than off-diagonal values.

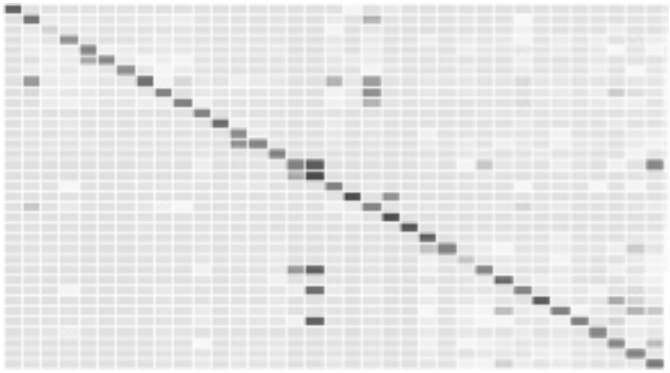
Fig. 2. The cross test for sessions of 35 randomly chosen users. Averaged values of anomaly indicator K are depicted using gray scale.

B. User substitution

We tested the model's ability to cope with the widespread case of intrusions when a password is stolen and another user logs in on behalf of the password owner. Sessions with inserted parts from other users were used for testing. The results of such a simulation are depicted in Fig. 3.

Figures show that the K values for "hostile" parts considerably differ from the values obtained in the legitimate sessions. It is, therefore, possible to introduce threshold classification between normal and abnormal sessions. If the value K exceeds this threshold, the session is marked as normal, otherwise – as anomalous. However, if both users belong to one "class" (i.e. have similar habits), such classification may be not so reliable.

C. Estimation of parameters

We conducted some experiments to investigate the influence of α on the process of adaptation to user's behavior. Fig. 4 shows that smaller values of α give quicker adaptation to changed behavior, while larger ones result in more gradual reaction to changes. It may be necessary to select an individual value of α for each user, because people have individual patterns of changes of their behavior. Although smaller values of α result in faster adaptation, it may lead to the increase of false negative rate because of an early forgetting of past information.

Let us name the value of L optimal if the approximating characteristics of the PST are not improved any more, i.e., further increase of L does not produce higher values of K. Fig. 5 shows that the worst value is $L = 1$, as the model is simplified to the trivial Markov chain of order 1. As L increases, the significant improvement can be observed, as longer contexts are taken into account and a more precise approximation becomes possible. For user UID 5003, optimal L can be set to be equal to five, as its further increase does not substantially increases the values of K (Fig. 5).

VI Conclusions

We presented an application of an adaptive version of Markov model with variable memory length to the anomaly detection task. The modified model can be applied to modeling various sequences. The use of an adaptive adjustment procedure allowed the approach to capture more subtle peculiarities of users' behavior and to continuously refine their models.

Since it is impossible to formalize precisely the anomaly criteria, it may be fruitful to use a set of techniques from which the conclusion will be drawn on the presence of intrusion. In this work, we offered two possible techniques of anomaly detection, namely, threshold classification and cross test technique, and provided their experimental testing.

One of perspective directions of further research is the use of ensembles of probabilistic trees [10], similar to those used in our work. Trees in those ensembles are taken into account with a certain weight when calculating probability of the next event in an audit stream, and there exists an adaptive procedure of weight adjustment.

It is also worth paying attention to a method of detection of anomalies at the level of system calls [13] as it allows us not to consider too irregular human behavior. However,

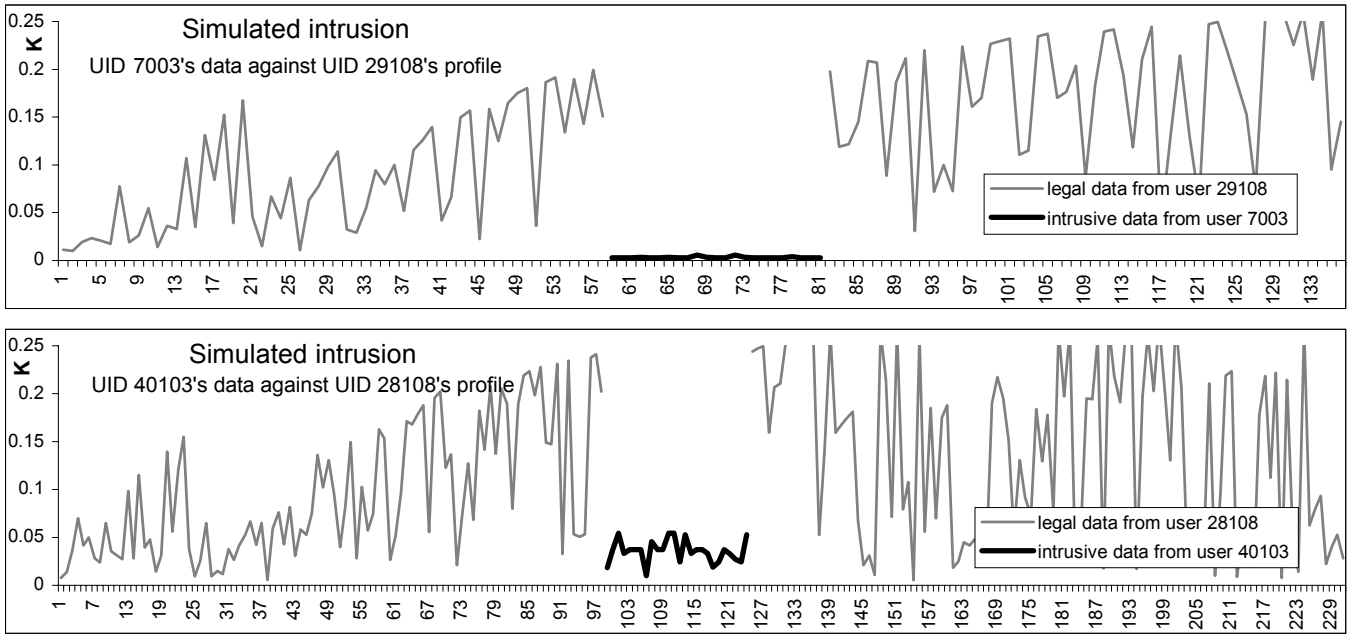

Fig. 3. User substitution examples. Log data from one user was inserted into a log of another, thus modeling a situation when a password was stolen or guessed. A sharp change of values of anomaly indicator K is observed at the time steps corresponding to the inserted (middle) parts of the sessions.

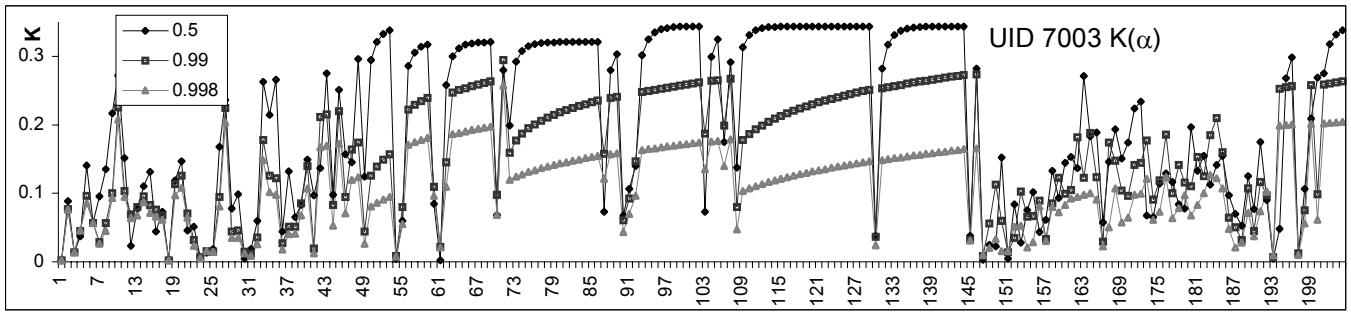

Fig. 4. Typical dependence of the course of adaptation on the value of α vs. number of session. "Monotone" zones correspond to periods of very stable patterns of this user behavior.

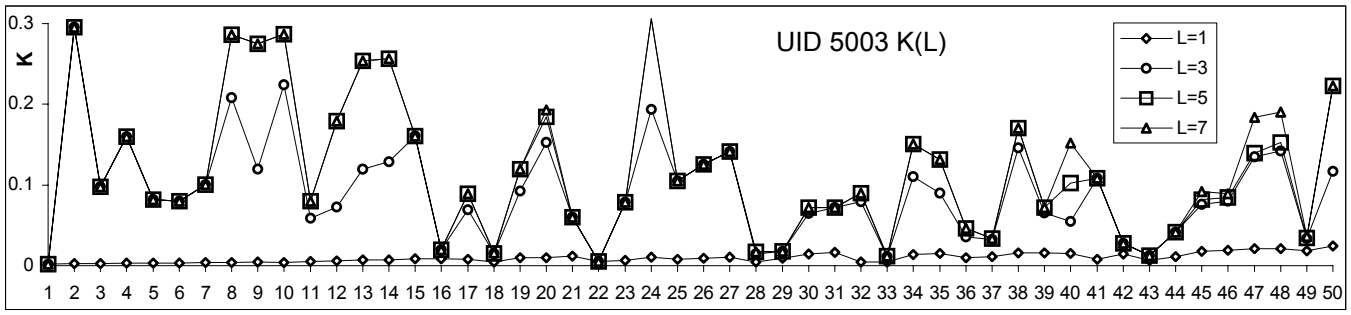

Fig. 5. Typical dependence of the course of adaptation on the value of L vs. number of session. Contexts of length not longer then $L = 5$ appeared to be sufficient to describe this user behavior (as further increase of L does not increase values of K).

completely refusing from analyzing user audit data is not desirable, as only the analysis at this level will allow detecting intrusions that cannot be detected at a level of system calls (e.g., use of stolen password).

REFERENCES

[1] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25 (2-3): 117-149, 1996.

[2] D. E. Denning. An intrusion-detection model. In *Proc. IEEE Symposium on Security and Privacy*, pp. 118-131, 1986.

[3] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. In *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, pp. 120-128. IEEE Computer Society Press, 1996.

[4] K. Tan. The Application Of Neural Networks To UNIX Computer Security. In *Proc. of the IEEE International Conference on Neural Networks,* 1, pp. 476-481, Perth, Australia, 1995.

[5] J. Ryan, M.-J. Lin, R. Miikkulainen. Intrusion Detection with Neural Networks, *Advances in Neural Information Processing Systems*, Cambridge MA: MIT Press, 1998.

[6] T. Lane. Hidden markov models for human/computer interface modeling. In *IJCAI-99 Workshop on Learning About Users*, pp. 35-44, 1999.

[7] C. C. Michael and A. Ghosh. Two state-based approaches to program-based anomaly detection. *ACM Transactions on Information and System Security*, 5 (2), 2002.

[8] D. Wagner and R. Dean. Intrusion detection via static analysis. In F. M. Titsworth, editor, *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pp. 156-169, Los Alamitos, CA, May 14-16 2001. IEEE Computer Society.

[9] B. D. Davison and H. Hirsh. Predicting sequences of user actions. In *Predicting the Future: AI Approaches to Time-Series Problems*, pp. 5-12, Madison, WI, July 1998. AAAI Press. Proceedings of AAAI-98/ICML-98 Workshop, published as Technical Report WS-98-07.

[10] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proc. 17 International Conf. on Machine Learning*, pp. 255-262. Morgan Kaufmann, San Francisco, CA, 2000.

[11] N. Ye. A markov chain model of temporal behavior for anomaly detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics, Information Assurance and Security Workshop*, pp. 171-174, 2000.

[12] D. A. Rachkovskij and E. M. Kussul. Binding and normalization of binary sparse distributed representations by context-dependent thinning. *Neural Computation* 13(2): 411-452, 2001.

[13] A. Somayaji. Automated response using system-call delays. In *USENIX Security Syposium 2000*. pp. 185-197, 2000.