# About Some Applications of Hidden Markov Model in Intrusion Detection Systems

Veselina Jecheva

***Abstract:*** *Intrusion detection systems (IDS) protect the computer networks such as a burglar alarm system against unauthorized access. The present paper introduces an approach to anomaly IDS based on Hidden Markov Models. The point is to process the sequences of system calls in order to distinguish the normal traces of system calls from abnormal ones. Simulations on Unix system data were accomplished and the results are represented as well.*

***Key words****: Intrusion Detection, Anomaly Detection, Hidden Markov Model, Viterbi Algorithm.*

## INTRODUCTION

Intrusion detection systems protect the networks from attacks and/or theft of their valuable data on the part of authorized users or outsiders. Like regular alarms the IDS can produce false alarms, referred to as false positives. A false positive occurs when the IDS generates an alarm from normal user activity. IDS approaches could be broadly classified into two basic types [1]: anomaly detection and misuse detection. Misuse based IDS (sometimes referred to as signature-based IDS) generate alarms based on specific attack signatures. These attack signatures encompass specific traffic or activity that is based on known intrusive activity [5]. Misuse IDS are effective in discovering of known attacks, but work unsatisfactory in the case of unknown attacks.

The present paper analyses the anomaly based IDS. The anomaly based IDS deal with a profile for each user group on the system describing the actions allowed for the group [6]. These profiles can be built automatically or created manually. The generated profiles are used as a baseline to define normal user activity. If any user activity deviates too far from this baseline, then the IDS generates an alarm. Anomaly based IDS have many advantages:

- Since the unauthorized access discovery is not based on certain signatures in user traffic, the system can potentially detect an unknown attack the first time it appears;
- The system has the potential to discover insider attacks or user account theft, since such activity would deviate too much from the user profiles defined;
- Since the unauthorized access detection is based on the user profiles defined, it is very difficult for an attacker to know with certainty what activities he can do without setting off an alarm.

[4] describes an approach that uses normal behaviour definition for the privileged processes, which can be executed on behalf of the root account, instead of normal user activity description. The user's normal behaviour is defined using a program specification language, describing the admissible activity (system calls and their parameters) for each process. [2] suggests a simplified version of this approach, which examines the results of the normal process execution and analyses "local (short) range ordering of system calls", named n-grams. A database containing all possible sequence call patterns of all processes that need to be protected is built as a result. This database is used for examination of the running process behaviour.

There are many methods in the practice to process the system call data gathered for the current process behavior examination. One of the simplest approaches ([2], [3]) is to compare the current process traces with normal traces from the database. In the case of non-coincidence the current sequence is assumed to be anomalous. A different approach ([7], [9]) uses finite state automata for normal user activity recognition. All these approaches, however, require one-to-one mapping between the patterns examined and

the normal user activity patterns. The presence of too many false positives is a described method drawback, since it is very difficult to create a comprehensive database containing all possible normal data patterns.

To avoid the mentioned disadvantages probabilistic methods for the recognition of unauthorized access patterns will be applied. The present paper suggests a method of intrusion detection identification, which is based on a preliminarily composed database of normal user activity patterns. The purpose is to examine the current user activity and to calculate the probability the current activity is normal user activity, using the Hidden Markov Model (HMM) [8]. If the probability found is less than a preliminarily specified threshold, the system assumes an unauthorized access is present and raises an alarm.

## 1. THE SYSTEM MODEL.

We will consider the unauthorized access recognition problem as a decoding problem in the following way. Assume we have a normal activity pattern set $S = (s_1, s_2, ..., s_n)$ given and a sequence of observations $Z = (z_1, z_2, ..., z_T)$ we have to examine. The set Z consists of current user activity patterns, which are the elements of the set $V = (v_1, v_2, ..., v_M)$. We are in a search for a sequence $\hat{Q} = \hat{q}_1 \hat{q}_2 ... \hat{q}_T$ which is closest to Z, where each $\hat{q}_i$ can take the value of some $s_j$ (i=1, 2,...,T; j=1, 2,..., n). In other words we look for the most probable normal user activity sequence $\hat{Q}$ for a sequence of observations Z given. If the probability calculated is less than a preliminarily defined threshold L, we will assume an unauthorized access is present.

### 1.1. The Hidden Markov Model.

Let's consider the finite alphabet $\Sigma$ and the finite pattern set $S = (s_1, s_2, ..., s_n)$, whose elements are finite sequences of the elements of $\Sigma$. We will examine the system behaviour at the discrete moments $t_j$, j=1, 2, ..., T, where the observation sequence Z is given. We will consider the set S as the system states set.

The system model $\lambda$ includes the ordered triple $\lambda$=(A, B, $\pi$), where:

1) The matrix A is the state transition probabilities set: $A = \{a_{ij}\}$, i=1, 2,...,n; j=1, 2,..., n, i.e. $a_{ij} = p(q_{t+1} = s_j \mid q_t = s_i)$, where $q_t$ denotes the system state at the moment t.

The probabilities satisfy the following conditions that are true of probabilities in general:

$a_{ij} \geq 0, 1 \leq i, j \leq n$;

$$\sum_{i=1}^{n} a_{ij} = 1, 1 \leq j \leq n$$

2) The matrix B is the distribution of each of the states $s_j$: $B = \{b_j(k)\}$:

$b_j(k) = p(z_t = v_k \mid q_t = s_j), 1 \leq j \leq n, 1 \leq k \leq M, 1 \leq t \leq T$.

Each $b_j(k)$ satisfies the following inequalities:

$b_j(k) \geq 0, 1 \leq j \leq n, 1 \leq k \leq n$,

$$\sum_{k=1}^{M} b_j(k) = 1, 1 \leq j \leq n.$$

3) The vector $\pi$ is the initial state distribution: $\pi = \{\pi_i\}$

$\pi_i = p(q_0 = s_i), 1 \leq i \leq n$.

We consider the system process as a first order hidden Markov process, described by the model $\lambda$=(A, B, $\pi$). For the sake of computational tractability, the following assumptions are made in the theory of HMMs:

1. The transition probability from the state $s_i$ to the state $s_j$ at the moment t+1 depends only on the state at the moment t (Markov assumption), i.e. $a_{ij} = p(q_{t+1} = s_j \mid q_t = s_i)$.

2. The state transition probabilities are independent of the actual time at which the transition takes place (stationarity assumption), i.e. $p(q_{t_1+1} = s_j \mid q_{t_1} = s_i) = p(q_{t_2+1} = s_j \mid q_{t_2} = s_i)$.

**1.2. The Viterbi Algorithm.**

We used the Viterbi algorithm [10] to find the described problem solution. This algorithm finds the most probable state sequence $\hat{Q}$.

Let's denote an auxiliary variable $\delta = \begin{pmatrix} \delta_1(1) & \delta_2(1) & ... & \delta_T(1) \\ \delta_1(2) & \delta_2(2) & ... & \delta_T(2) \\ ... & ... & ... & ... \\ \delta_1(n) & \delta_2(n) & ... & \delta_T(n) \end{pmatrix}$

where $\delta_t(i) = \max_{q_1 q_2 ... q_{t-1}} p(q_1 q_2 ... q_t = s_i, z_1 z_2 ... z_t \mid \lambda)$.

Consequently $\delta_{t+1}(j) = \max_i \delta_t(i) a_{ij} b_j(z_{t+1})$. We can find the probability of occurrence of the sequence $\hat{Q}$ by calculation of all $\delta_t(i)$, $1 \le i \le n$, $1 \le t \le T$. But since we are interested in the actual state path with this probability, so we will add an additional matrix $\psi$ to keep track of the most probable state index at each step:

$$\psi = \begin{pmatrix} \psi_1(1) & \psi_2(1) & ... & \psi_T(1) \\ \psi_1(2) & \psi_2(2) & ... & \psi_T(2) \\ ... & ... & ... & ... \\ \psi_1(n) & \psi_2(n) & ... & \psi_T(n) \end{pmatrix}$$

where each $\psi_{t+1}(j) = \arg\max_{1 \le i \le n} \delta_t(i) a_{ij}$ and $\arg\max_{1 \le i \le n} \delta_t(i) a_{ij}$ denotes the value of i $\delta_t(i) a_{ij}$ has maximum.

**Viterbi algorithm description:**

1) Initialization:

$\delta_1(i) = \pi_i b_i(z_1)$, $1 \le i \le n$

$\psi_1(i) = 0$

2) Iterative steps:

$\delta_{t+1}(j) = \max_i \delta_t(i) a_{ij} b_j(z_{t+1})$, $1 \le i \le n$, $1 \le j \le n$, $1 \le t \le T-1$

$\psi_{t+1}(j) = \arg\max_{1 \le i \le n} \delta_t(i) a_{ij}$

3) Termination: t=T

$P(\hat{Q} \mid Z, \lambda) = \max_i \delta_T(i)$

$\hat{q}_T = \arg\max_{1 \le i \le n} \delta_T(i)$.

So we can calculate at each step the most probable state $\hat{q}_i$, $1 \le i \le T$, corresponding to the observation $z_i$, $1 \le i \le T$ and the probability of state transition $\delta_t(i)$ using the Viterbi algorithm. After the algorithm conclusion we have to calculate the general probability the user activity examined to be normal user activity. We can assume the average

$\delta_{av} = \frac{\sum_{k=1}^{T} \delta_k(i)}{T}$ is a reliable estimation of the mean of the quantities $\delta_1(i)$, $\delta_2(i)$,..., $\delta_T(i)$. If $\delta_{av} < L$, the IDS will consider the process examined as an unauthorized access result.

**1.3. The Algorithm Complexity.**

For each execution step of the algorithm described $n^2$ calculations of the possible transitions and n comparisons among the $n^2$ results are needed. Consequently the total number of the algorithm operations is $T(n+n^2)$, i.e. the algorithm has $O(Tn^2)$ complexity.

Since for each step execution of the algorithm (2n+1) floating point numbers are necessary, the total memory required is T(2n+1) floating point storage locations.

**2. SIMULATION EXPERIMENTS.**

Simulation experiments, based on the model defined, were accomplished. The experimental data are obtained from Unix based system examination and consist normal user activity patterns of some processes (sendmail, ftp, named and lpr). The methods for pattern generation are described in [2] and [3]. Each pattern is a sequence of system calls, which are the results of the examined process. Each input data file is a sequence of ordered pairs of numbers, one pair per line. The first number in a pair is the process ID (PID) of the process executed, and the second one is the system call number. The correspondence between system call numbers and the actual system call names is stored in an additional file. For example, the number 10 stands for the system call "execv", the number 20 represents "getpid", 132 stands for "mkdir" and so on. Forks are considered as separate processes and their execution results are considered as normal user activity. Table 1 contains some examples of input data:

Table 1. System call data, containing PID and system call number.

| PID | 1393 | 1393 | ... | 1423 |
|---|---|---|---|---|
| System calls | 112 | 19 | ... | 105 |

For the simulations a software in C++ was developed, using the object-oriented programming methods. The experimental data contain normal user activity patterns including main process patterns (sendmail, ftp, named and lpr), as well as their child processes patterns. Since these data are normal user activity patterns, they are the elements of the set S, i.e. the system states. The experimental data also contain abnormal user activity patterns, generated by some common intrusion tools (sm565a, syslog-local, syslog-remote, sscp, decode). These data are put into sequence which the elements are from the observation sequence *Z* with length T from the model. The results shown are the number of the attacks produced by the simulations with the input test data.

We accomplished experiments with the following values of T: 5, 10, 15 and 20 and the following values of L: 0.90, 0.85, 0.80 and 0.75. The result dependences of the number of the unauthorized process executions and the number of the false positives on the input parameters T and L are shown in the tables 2-5 as follows:

Table 2. Number of intrusion signals depending on the value of T.

| intrusion tool | T=5 | T=10 | T=15 | T=20 |
|---|---|---|---|---|
| sm565a | 119 | 158 | 205 | 274 |
| syslog-local | 274 | 295 | 321 | 362 |
| syslog-remote | 220 | 263 | 292 | 331 |
| sscp | 187 | 220 | 273 | 316 |
| decode | 163 | 194 | 248 | 298 |

Table 3. Number of intrusion signals depending on the value of L.

| intrusion tool | L=0.90 | L=0.85 | L=0.80 | L=0.75 |
|---|---|---|---|---|
| sm565a | 104 | 122 | 175 | 203 |
| syslog-local | 255 | 284 | 307 | 351 |
| syslog-remote | 214 | 249 | 278 | 293 |
| sscp | 176 | 204 | 245 | 288 |
| decode | 154 | 175 | 196 | 224 |

Table 4. Number of false positives depending on the value of T.

| intrusion tool | T=5 | T=10 | T=15 | T=20 |
|---|---|---|---|---|
| sm565a | 4 | 9 | 16 | 23 |
| syslog-local | 7 | 12 | 14 | 18 |
| syslog-remote | 13 | 17 | 21 | 24 |
| sscp | 11 | 16 | 20 | 22 |
| decode | 8 | 13 | 15 | 19 |

Table 5. Number of false positives depending on the value of L.

| intrusion tool | L=0.90 | L=0.85 | L=0.80 | L=0.75 |
|---|---|---|---|---|
| sm565a | 5 | 9 | 12 | 14 |
| syslog-local | 11 | 13 | 16 | 18 |
| syslog-remote | 8 | 11 | 15 | 17 |
| sscp | 7 | 10 | 13 | 16 |
| decode | 9 | 14 | 19 | 21 |

## 3. DISCUSSIONS.

The obtained results are analyzed in the following way. When the values of L and T increase, the number of intrusion signals increases, but the false positives rate increases as well. Consequently the values of L and T have to be determined as a result of some compromise in each practical solution.

An advantage of the algorithm is its potential to detect an unknown attack the first time it appears, since it relies on probabilistic methods instead of one-to-one mapping between the current patterns and those in the database. A drawback of the algorithm is its considerable price, as it has $O(Tn^2)$ complexity. Since n is the number of the states, i.e. the number of normal user activity patterns in the database, its value could be significant in the case of a large system. Another disadvantage of the anomaly based IDS in general is the creation of the database containing the user profiles, which could be a task of considerable difficulty.

## CONCLUSIONS AND FUTURE WORK

The present paper proposes a method of unauthorized access recognition in the anomaly based IDS. Since we considered it as a decoding problem, we applied the Viterbi algorithm to distinguish the normal user activity and the intrusion activity. Some experimental simulations were accomplished as well. The purpose of future work could be the algorithm optimization, as well as some different probabilistic method applications and the comparison of the results obtained.

## REFERENCES

[1] Allen J., A.Christie, W.Fithen, J.McHugh, J.Pickel, E.Stoner; State of the Practice of Intrusion Detection Technologies; Technical Report CMU/SEI-99-TR-028, Software Engineering Institute, Carnegie Mellon, January 2000.

[2] Forrest S., S.A. Hofmeyr, A. Somayaji, T.A. Longtaff, A Sense of Self for Unix Processes. In Proceedings of the 1996 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitors, CA, pp.120-128.

[3] Forrest S., S.A. Hofmeyr, A. Somayaji, Intrusion detection using sequencies of system calls, Journal of Computer Securit,y Vol. 6, 1998, pp. 151-180.

[4] Ko C., Fink G., Levitt K., Automated Detection of Vulnerabilities in Privileged Programs by Execution Monitoring, Proceedings of the 10th Annual Computer Security Applications Conference, 1994, pp.134-144.

[5] Kumar S., E. Spafford, A Pattern-Matching Model for Intrusion Detection. Proceedings of the National Computer Security Conference, 1994, pp.11-21.

[6] Lee W., S. J. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems," In ACM Transactions on Information and System Security, Vol. 3, No. 4, 2000, pp. 227–261.

[7] Micheal C. C., A. Ghosh, "Simple, state-based approaches to program-based anomaly detection," ACM Transactions on Information and System Security, Vol. 5, No. 3, August 2002, pp. 203-237.

[8] Rabiner L.R., A tutorial on Hidden Markov Models and selected applications in speech recognition, Proc. IEEE, 257-286, 77, 2, Feb 1989.

[9] Sekar R., M. Bendre, Dhurjati, P. Bullineni, "A fast automaton-based method for detecting anomalous program behaviours," IEEE Symposium on Security and Privacy, 2001, S&P 2001, pp. 144 – 155.

[10] Viterbi A.J., Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Transactions on Information Theory, IT 13, 1967, pp. 260 – 269.

## ABOUT THE AUTHOR

Veselina Jecheva, PhD, Centre of Computer and Technical Sciences, Burgas Free University, Phone: +359 56 900 487, E-mail:vessi@bfu.bg .