Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 641 – 649
ISSN 1896-7094

Anomaly Based Intrusion Detection Based on the Junction Tree Algorithm

Evgeniya Petrova Nikolova, Veselina Gospodinova Jecheva

Burgas Free University, Faculty for Computer Science and Engineering
62 Str. San Stefano, 8100 Burgas, Bulgaria,
enikolova@bfu.bg, vessi@bfu.bg

Abstract. The aim of this paper is to present a methodology for the attacks recognition during the normal activities in the system. Since the proposed approach uses the graphical representation method, we apply the junction tree algorithm (JTA). Some results from the accomplished simulation experiments are submitted as well.

1 Introduction

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices. An intrusion detection system (IDS) is a system that automates the intrusion detection process. The basic functions of IDSs include recording information related to observed events, notifying the system administrators of important observed events and generating reports [1].

The task of an intrusion detection system (IDS) is modeled as a classification problem in a machine-learning context. It monitors system data (network or host) to distinguish intrusions and attacks or normal user activity. Based on the intrusion detection method IDS can be categorized into two main categories: misuse based and anomaly based. Misuse detection IDSs generate the alarms based on specific attack signatures. Despite of the fact they achieve a high level of accuracy, their major drawback is the little possibility of detecting novel attacks [10].

In this paper we direct our attention to the anomaly based IDS. A typical anomaly detection model will analyze data, compare to a preliminarily defined profile, run statistical analysis to determine if any significant deviation is found, and flag the event as a normal activity or an attack [2]. The advantage of anomaly based IDS is the potential to detect novel attacks or unknown attacks, variations of known attacks, and the ability to detect insider attacks or identity theft. Another benefit of this approach is that learning to describe normal activity can be automated.

Analyzing program behavior profiles for intrusion detection has emerged as a viable alternative to user-based approaches to intrusion detection [5]. Program behavior profiles are built by capturing system calls made by the program under analysis under normal operational conditions. If the captured behavior represents a compact and adequate signature of normal behavior, then the profile can be used to

detect deviations from normal behavior such as those that occur when a program is being misused for intrusion.

In this paper we are interested in finding methodology for the attacks recognition during the normal activities in the system, i.e. to discern the normal activity patterns from abnormal ones. For that purpose we apply the junction tree algorithm (JTA). The algorithm takes the form of message passing on a graph referred to as a junction tree, whose nodes are clusters of variables. Each cluster starts with one potential of the factorized density. By combining this potential with the potentials it receives from its neighbors, it can compute the marginal over its variables. Except finding marginal probabilities, JTA helps to find the most likely state of the distribution and to define the anomalies in the obtained sequence. For completeness, in Section 2 we give basic facts about JTA. In Section 3 we give a brief exposition of the applied methodology. Finally, some of our experimental results are provided in Section 4.

2 The Junction Tree Algorithm

Let $X=(X_1, X_2, \ldots, X_n)$ be a random vector with density p and $G=(V,E)$ be an undirected graph for which each node s has an associated random variable X_s. Denote $X_A=\{X_s, s \in A\}, A \subseteq V$. A subset of vertices all joined by edges is called clique.

The graph G can be used to enforce the random vector X (or on the distribution p) in different ways:

Markov property: X is Markov with respect to G if X_A and X_B are conditionally independent given X_S whenever S separates A and B.

Factorization: The distribution p factorizes according to G if it can be expressed as a product over cliques:

$$p(x) = \frac{1}{Z} \prod_C \psi_C (x_C),$$

where $Z = \int \prod_C \psi_C (x_C) dx_C$ and $\psi_C (x_C)$ is the potential for clique C.

Cluster nodes within cliques of graph with cycles to form a clique tree.

A clique tree with **running intersection property:** all nodes on the unique path between clique nodes C_1 and C_2 contain the intersection $C_1 \cap C_2$, is called a **junction tree.**

The goal of the junction tree algorithm (JTA) is to define a potential representation of the graph such that, coupled with a suitable algorithm to modify these potentials, will result in the marginals of individual or groups (cliques) could be obtained directly from the modified potentials.

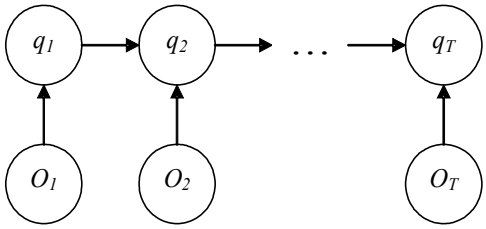
Algorithm: ([9])

- Given an undirected graph G. Form a triangulated graph $G^\wedge$ by adding edges as necessary. (Chord is a link joining two non-consecutive vertices of a loop. An undirected graph is triangulated if every loop of length 4 or more has a chord.)
- Form the clique graph (in which nodes are cliques of the triangulated graph).
- Extract a junction tree.
- Run sum-product on the resulting junction tree.

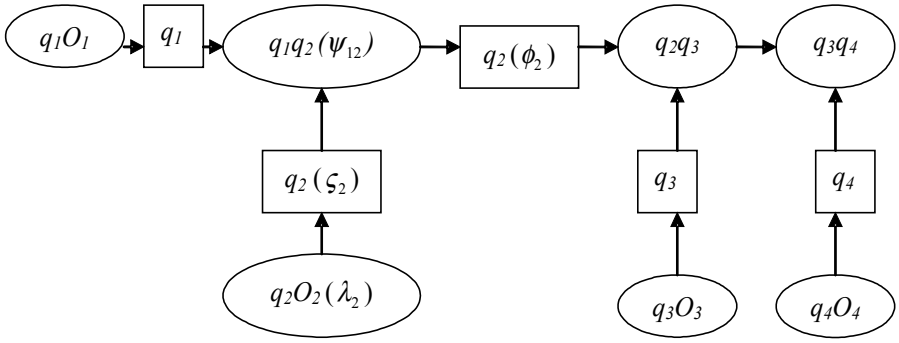
Except finding marginal probabilities, JTA helps to answer another natural question: what is the most likely state of the distribution? Running JTA produces the marginal probabilities. Consider a modified distribution p^* in which we wish to re-weight the states, making the higher probability states exponentially more likely than lower probability states. This can be achieved by $p^* \propto (p)^{\beta}$, where β is a very large positive quantity. This makes the distribution p^* very peaked around the most-likely value of p. In the JTA we need to carry out summations over state. Since in the limit $\beta \to \infty$ only the most-likely state will contribute, hence the summation operation can be replaced by a maximization operation.
More details not introduced here could be found in [7] and [8].

3 Outline of the Methodology

Let's consider the system work in the discrete moments of time $t=1, 2, .. , T$, when the system occupies some of the following N states: $S_1, S_2, ..., S_N$. Let $O=(O_1, O_2, ... O_T)$ is the observation sequence at the moments $t=1, 2, ..., T$. We applied the junction tree algorithm to the graphical model representing the system work, which is shown below



where $Q=(q_1, q_2, ..., q_T)$ is the state sequence at the moments $t=1, 2, ..., T$. Each q_t is one of the elements of the set $S = \{ s_1, ..., s_N \}$. Consider the following junction tree



First we initialize all clique and separator potentials with 1s. Then for each node in the original graph, we multiply its conditional probability distribution onto the clique to which it is assigned. If we assume initially that the nodes, containing observations

O, are hidden, the λ potentials at the node q_tO_t will be all 1s and the potentials along the node q_tq_{t+1} will be as follows:

$$\Psi_{1,2}(i,j) = P(q_2 = s_j | q_1 = s_i)\, P(q_1 = s_i)$$
$$\Psi_{t,t+1}(i,j) = P(q_{t+1} = s_j | q_t = s_i)$$

3.1 Forward steps

We initialize the potentials with evidence

$$\lambda_t^*(i) = P(O_t | q_t = s_i),\ \lambda_t^*(i) = \frac{P(O_t, q_t = s_i)}{P(O_t)} = \frac{\varsigma_t^*(i)}{1} = \varsigma_t^*(i).$$

Run the forward algorithm, we get the following potentials:

$$\psi_{t-1,t}^*(i,j) = P(q_{t-1}, q_t, O_{1:t}) = P(q_t = s_j | q_{t-1} = s_i)\, P(q_{t-1} = s_i, O_{1:t-1})\, P(O_t | q_t = s_j)$$
$$= \psi_{t-1,t}(i,j) \frac{\phi_{t-1}^*(i)}{1} \frac{\varsigma_t^*(j)}{1}.$$

Marginalizing yields:

$$\phi_t^*(j) = P(q_t = s_i, O_{1:t}) = \sum_i \psi_{t-1,t}^*(i,j).$$

3.2 Backward steps

In the backward steps, we find that the potentials are

$$\psi_{t-1,t}^{**}(i,j) = P(q_{t-1} = s_i, q_t = s_j, O_{1:T}) = P(q_{t-1} = s_i | q_t = s_j, O_{1:t})\, P(q_t = s_j, O_{1:t}, O_{t+1:T})$$
$$= \frac{\psi_{t-1,t}^*(i,j)}{\phi_t^*(j)} \phi_t^{**}(j).$$

Marginalizing yields:

$$\phi_t^{**}(j) = P(q_t = s_j, O_{1:T}) = \sum_i \psi_{t-1,t}^{**}(i,j).$$

If we propagate messages from children to their parents in the rooted junction tree, we have by induction after the forwards steps (after collecting to root), $\psi_{t,t+1}^*(i,j) = P(q_t = s_i, q_{t+1} = s_j)$, $\phi_t^*(i) = P(q_t = s_i)$ and all the other potentials are unchanged (all 1s). After the backward steps we have

$$\psi^{**}_{t-1,t}(i,j) = P(q_{t-1} = s_i, q_t = s_j),$$

$$\varsigma^{**}_t(i) = \lambda^{**}_t(i) = \phi^{**}_t(i) = P(q_t = s_i).$$

As a result from the JTA we obtain the most likely state sequence during the examined period of time. Finally we compute the state transition probabilities for the result state sequence.

Each algorithm step at both forward and backward directions requires $O(M^2)$ operations. Since we have a total of T moments of time, the entire computing procedure will require $O(TM^2)$ operations, where T corresponds to the number of cliques and M^2 is the cardinality of the clique state space.

4 Experiments

The experiment data were obtained from Unix-based and Sun SPARCstations system examination during some period of time. The experimental data include normal user activity traces of some privileged processes that run with administrative rights (synthetic ftp, synthetic lpr, login, named and xlock) and their child processes patterns, as well as intrusion data. The input data files are sequences of ordered pairs of numbers, where the first number is the process ID (PID) of the process executed, and the second one is the system call number. Forks are considered as separate processes and their execution results are examined as normal user activity.

The methods for pattern generation are described in [4] and [5]. They have proved that short sequences of system call traces produced by the execution of the privileged processes are a good discriminator between the normal and abnormal operating characteristics of programs. The experimental results have also confirmed that short sequences of system call traces are stable and consistent during program's normal activities.

Experiments based on the described algorithm, were accomplished using corresponding software in C++. Based on the normal user activity patterns we evaluated the state transition probabilities during the system work in attack absence. The experimental data also contain abnormal user activity patterns, generated by some common intrusion tools (sunsendmailcp, lprcp, buffer overflow, some Trojans, etc.). We considered these data as current system activity data, which could be examined by the JTA. We used a slide window with length T =10 to cross the traces of current user activity. Given an unknown observation sequence, we find the most likely states sequence, as well as its state transition probabilities. Comparing the obtained state transition probabilities with the state transition probabilities of the normal user activity patterns we detect the intrusions presence. Some of the results from this comparison for the enumerated processes are summarized in Figures 1, 2, 3, 4 and 5.

In order to evaluate the false alarms rate we applied a method proposed in [6]. Figures 6, 7, 8, 9 and 10 represent the false positive rate (i.e. classifying normal as an attack) for the following processes: synthetic ftp, synthetic lpr, login, named and xlock. The false positive rate (i.e. classifying normal activity as an anomaly) for the

described method is 8,6%, and the false negative rate (i.e. classifying anomaly as a normal) in this case is 3%. The presented anomaly detection method could accurately verify a given unknown sequence to be normal or anomalous with 88,4% accuracy. Table 1 contains the values of the false positive rate, the false negative rate and the accuracy for the mentioned privileged processes.

Table 1 The false alarms rate and the algorithm accuracy

Process	False positive rate	False negative rate	Accuracy
synthetic ftp	5%	1%	94%
synthetic lpr	9%	4%	87%
login	7%	3%	90%
named	6%	1%	93%
xlock	16%	6%	78%

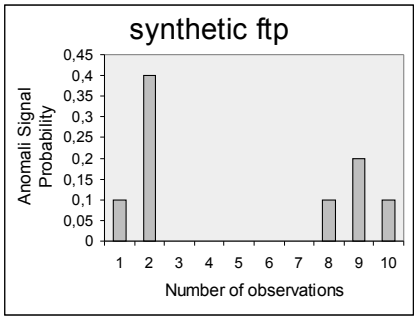


Figure 1

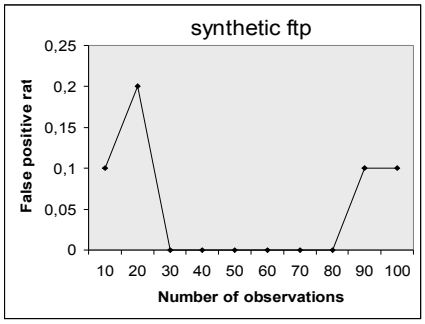


Figure 6

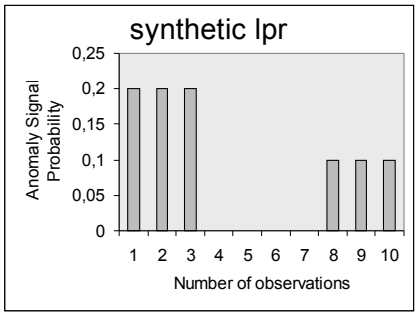


Figure 2

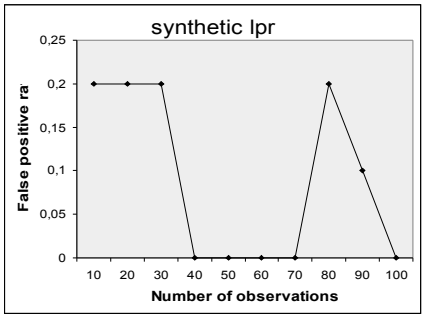


Figure 7

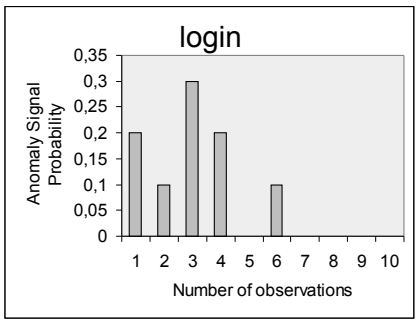


Figure 3

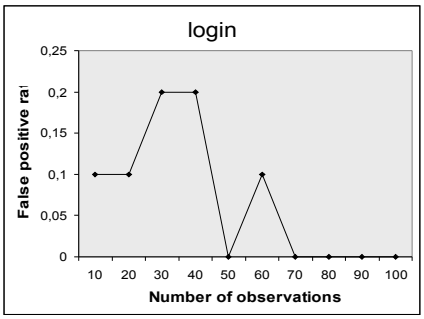


Figure 8

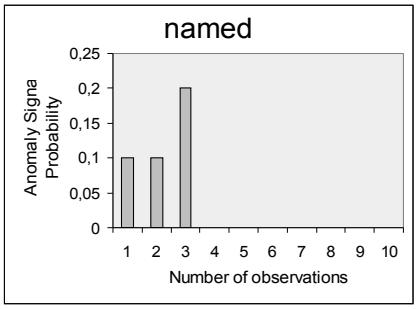


Figure 4

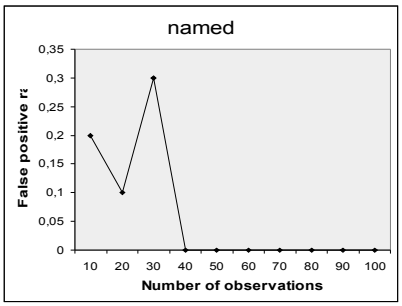


Figure 9

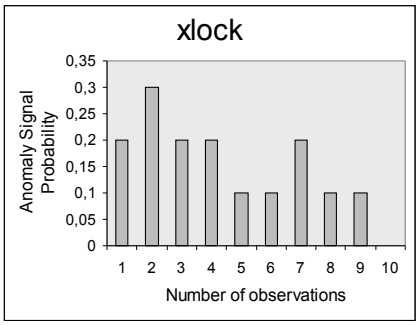


Figure 5

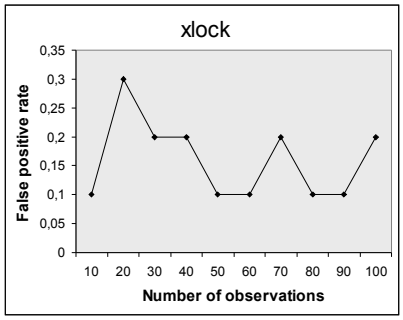


Figure 10

5 Discussions

This paper investigated the capabilities of the JTA in anomaly based intrusion detection. As a result of the JTA we found the joint distribution on the non-obvious variables in a clique with all the obvious variables clamped in their obvious states.

Then calculating the likelihood is easy since we just sum out over the non-obvious variables of any converged potential. From the received sequence it is easy to determine if it is anomalous, comparing to the state transition probability.

The major advantage of graphical representation models, such as the one in JTA, over many other types of predictive models, such as neural networks and decision trees, is that unlike those "black box" approaches, the graph structure represents the inter-relationships among the data set attributes. Human experts can easily understand the network structures and if necessary can easily modify them to obtain better predictive models. Other advantages of graphical representation models include explicit uncertainty characterization, fast and efficient computation, and quick training. They are highly adaptive and easy to build, and provide explicit representation of domain specific knowledge in human reasoning frameworks. Moreover, graphical representations offer good generalization with limited training data and easy maintenance when adding new features or new training data.

Another advantage of the anomaly-based approach in general is its potential to detect novel attacks, insider attacks or account theft. The drawback of the described method is its considerable computational price, since it has $O(TM^2)$ complexity. Since M^2 is the cardinality of the clique state space, 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 and requires some period of time the system is unprotected.

6 Conclusions and Future Work

The present paper proposes a method of intrusions access recognition in the anomaly based IDS. We applied a graphical representation model, precisely the Junction tree algorithm, as well as some probabilistic methods to distinguish the normal user activity from the intrusion one. 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 obtained results.

References

1. Abraham A., Thomas J., *Distributed Intrusion Detection Systems: A Computational Intelligence Approach, Applications of Information Systems to Homeland Security and Defense,* Idea Group Inc. Publishers, USA, Chapter 5, 2005, pp. 105-135.
2. Chan P., M. Mahoney, M. Arshad, *Learning Rules and Clusters for Network Anomaly Detection,* Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection, George Mason University, Technical Report CS-2003-06, 2003.
3. 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.
4. Forrest S., S. A. Hofmeyr, A. Somayaji, *Intrusion detection using sequences of system calls,* Journal of Computer Security Vol. 6, 1998, pp. 151-180.

5. Ghosh A. K., A. Schwartzbard, M. Schatz, *Learning Program Behavior Profiles for Intrusion Detection,* In Proceedings of the 1st Workshop on Intrusion Detection and Network Monitoring, pp. 51–62, 1999.
6. Hoang X. D., J. Hu, P. Bertok, *A Multi-layer Model for Anomaly Intrusion Detection Using Program Sequences of System Calls,* 11th IEEE International Conference on Networks (ICON 2003), Sydney, Australia, 2003.
7. Jensen F. V., F. Jensen, *Optimal Junction Trees,* Uncertainty in Artificial Intelligence, 1994.
8. Lauritzen S. L., *Graphical Models,* Oxford Science Publications, 1996.
9. Lauritzen S. L., D. J. Spiegelhalter, *Local computations with probabilities on graphical structures and their application to expert systems (with discussion),* Journal of Royal Statistical Society, Series B, 50(2), 1988, pp. 157–224.
10. Tran T. P., T. Jan, A. J. Simmonds, *A Multi-Expert Classification Framework for Network Misuse Detection,* From Proceeding (544) Artificial Intelligence and Soft Computing, ISBN 0-88986-610-4, 2006.