

Design, Implementation, and Validation of a Self-Learning Intrusion Detection System

Davide Adami¹, Christian Callegari², Stefano Giordano², Giada Landi², and Michele Pagano²

¹CNIT - Reasearch Unit Dept. of Information Engineering, University of Pisa, ITALY

E-mail: davide.adami@cnit.it

²Dept. of Information Engineering, University of Pisa, ITALY

E-mail: {[callegari](mailto:callegari@netserv.iet.unipi.it),[landi](mailto:landi@netserv.iet.unipi.it),[pagano](mailto:pagano@netserv.iet.unipi.it),[giordano](mailto:giordano@netserv.iet.unipi.it)}@netserv.iet.unipi.it

Abstract—The use of Intrusion Detection Systems (IDSs) has emerged as a key element in network security. In this paper we present the design and the validation of an Anomaly based Network IDS, named Self Learning Intrusion Detection System (SLIDS), able to identify new ad hoc attacks. SLIDS has a modular structure, that not only provides flexibility and extensibility features, but also permits to deal with a wide range of attacks, exploiting various vulnerabilities of TCP/IP stack.

The paper presents the system architecture, as well as experimental tests carried out to evaluate the performance of SLIDS both with the well known DARPA data set and with actual traffic, highlighting its effectiveness in detecting different kinds of attacks.

I. INTRODUCTION

IN the last few years, Internet has experienced an explosive growth. Along with the wide proliferation of new services, the number and impact of security attacks have been continuously increasing. Indeed, the knowledge required to carry out an attack has been decreasing, since software tools for this aim are largely available on Web sites all over the world [1][2].

Recent advances in encryption, public key exchange, digital signatures, and the development of related standards have set a foundation for network security. However, network security goes beyond, because it must include security of computer systems and networks, at all levels, top to bottom.

To this aim, the use of an Intrusion Detection System (IDS) is of primary importance to reveal ongoing intrusions in a network or in a system. An IDS [3] is a software/hardware tool designed to reveal an intrusion while it is in act, or after it has occurred. IDSs are usually classified on the basis of two distinct aspects: the scope (Network IDSs vs. Host IDSs) and the detection technique (Misuse or Signature based IDSs vs. Anomaly based IDSs)[4][5].

State of the art in the field of intrusion detection is mainly represented by misuse based IDSs [6]. Considering that most attacks are realized with known tools, downloaded from the Internet, a signature based IDS could seem a good solution. Nevertheless the most dangerous attacks are those prepared ad hoc. Hence, a misuse based IDS is completely unable to block such kind of attacks.

In this paper we present the design, the implementation, and the validation of a network IDS, based on anomaly detection techniques. The system, which has been designed

as an original modification of well-tested approaches, relies on supervised learning techniques. Given a training dataset, the IDS is able to build a model of the normal behavior of the network, which will be used, during the running of the system, to classify network activity either as normal or anomalous. Mainly for this aspect, the system is named Self-Learning Intrusion Detection System (SLIDS).

The paper is organized as follows: in section 2 we describe the architecture and the functionalities of SLIDS. Section 3 consists of two sub-sections, which are devoted to the performance analysis of our IDS both in an emulated network and in a real one. Finally section 4 concludes the paper with some final remarks and provides suggestions for future works.

II. SLIDS

In this section we provide a detailed description of SLIDS architecture. SLIDS is a software anomaly based IDS, composed of several modules. The modular implementation has been chosen, because it guarantees scalability and extensibility. To provide more flexibility, network administrators can select and configure the modules to be used.

The major feature of SLIDS architecture (see Fig.1) is that, differently from most of the current state of the art IDSs, more than one approach for detecting anomalies has been used.

At first, SLIDS needs to perform a training phase to learn the normal behavior of the network. During this phase the system analyzes some traffic, which is supposed to be attack-free. As a result of this analysis, it builds some rule-sets to be used in the detection phase.

In the next subsections we will describe the behavior of each module both during the training and the detection phase.

A. Traffic Filter

This module is responsible for traffic filtering. First of all it drops all non-IP packets and all packets addressed towards external destinations. Subsequently, the filter checks the protocol field of the remaining packets and forwards them to the corresponding module. Packets, whose protocol field differs from 1 (ICMP), 6 (TCP), or 17 (UDP), are not processed by the system.

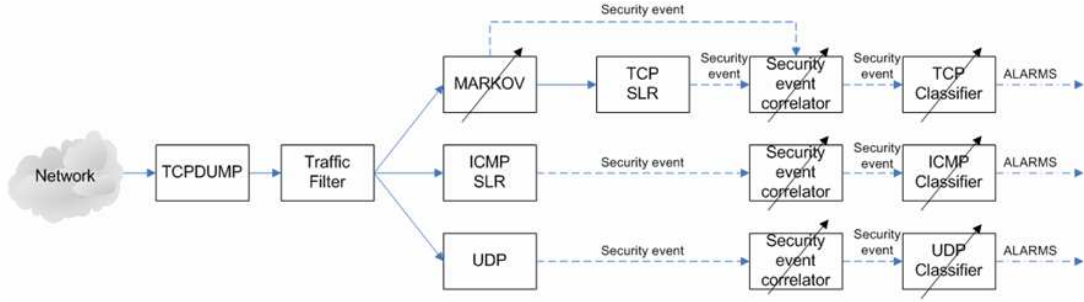


Fig. 1. SLIDS Architecture

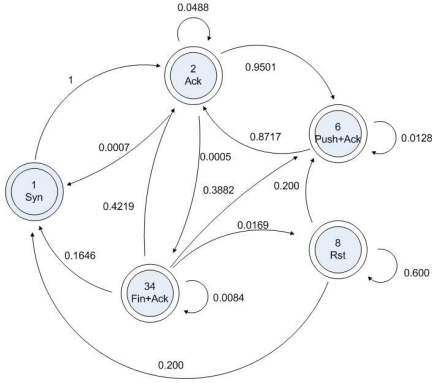


Fig. 2. SSH Markov Chain

B. TCP/Markov Module

The TCP/Markov module is based on the idea that TCP connections can be modeled by Markov chains [7][8]. It is state-full module since it takes into account information related to previous packets of the same flow.

The use of Markov Chains for modeling a TCP flow is a well-known method in the field of intrusion detection. Nevertheless our implementation is based on an original modification of this approach. Indeed, experimental results, obtained from the analysis of real traffic traces captured in our laboratory, have shown that the Markov chains associated to distinct application are strongly different. For example, figure 2 and 3 show the Markov chains, derived for SSH and FTP connections respectively.

Therefore SLIDS calculates one distinct matrix for each application (identified on the basis of the destination port number). Before starting the training, the module performs an additional filtering phase. By default, it only analyzes packets belonging to SSH, HTTP, and FTP connections, but in the tuning phase the network administrator can choose to monitor other types of applications. All other packets are directly forwarded to the TCP/SLR module without been processed. This filtering is also performed in the detection phase.

The module only considers a few fields of the packet headers, more precisely the IP destination address, the destination port number, and the TCP flags. The IP address and the port number are used to identify a connection, while the value of the flags is used to identify the chain transitions.

During the training phase, the module reconstructs TCP connections and associates a value S_p to each packet, according to the following relationship:

$$S_p = syn + 2 \cdot ack + 4 \cdot psh + 8 \cdot rst + 16 \cdot urg + 32 \cdot fin$$

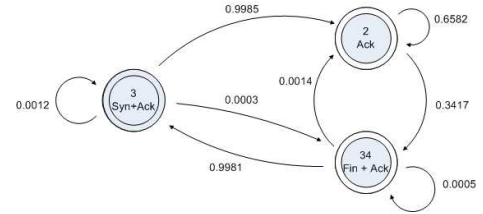


Fig. 3. FTP Markov Chain

Thus, each connection is represented by a sequence of symbols S_i . These symbols can be considered as the states of a Markov chain. Hence, the modules calculates the transition probabilities matrix A , where

$$a_{ij} = P[q_{t+1} = j | q_t = i] = \frac{P[q_t = i, q_{t+1} = j]}{P[q_t = i]}$$

In the detection phase the TCP/Markov module considers each packet as part of a connection. For each connection, the module uses a sliding window mechanism to process the packets, where the dimension T of the sliding window can be configured. Thus, when processing the packets, the module computes T symbols $S = \{S_{R+1}, S_{R+2}, \dots, S_{R+T}\}$ and estimates the probability $P[S|A]$, where A is the matrix obtained in the training phase. Actually, the system calculates the logarithm of the Likelihood Function ($LogLF$) and its “temporal derivative” D_W (where the default value for the parameter W is 3).

$$LogLF(t) = \sum_{t=R+1}^{T+R} Log(a_{S_t S_{t+1}})$$

$$D_w(t) = \left| LogLF(t) - \frac{1}{W} \sum_{i=1}^W LogLF(t-i) \right|$$

A sequence of non-expected symbols produces a low probability: an anomaly determines a rapid decrease in the $LogLF$ and a peak in the D_W . If the D_W is bigger than a predefined threshold, a security event is generated. The security event has an anomaly score, which is calculated as $Log(D_W)$.

This module is very efficient in terms of memory and processing requirements. Given the nature of TCP, the number of actual flags configurations, identified during the training phase, is usually less than ten, which implies the storage of a matrix composed of less than one hundred elements (400B) for each application. It is then necessary to store, during the detection phase, T bytes corresponding to the T flags values inside the sliding window (simulations have shown that a small T , around 30, is necessary to reveal an intrusion quickly). The

ease in the computation of the Likelihood Function, clearly shows that this method can be applied to on-line detection.

After this phase the packets are forwarded to the TCP/SLR Module.

C. TCP/SLR Module

The TCP/SLR module constructs a protocol-specific rule-set, by analysing the first 64 bytes of each training packet [9], i.e. IP and TCP headers plus some bytes of the payload.

Before starting the training phase, the TCP/SLR module performs an additional filtering phase. Indeed, the module only analyzes the SYN packets, the first hundred packets of each connection, and the FIN packets, while discards all the others. This filtering allows to reduce the processing time without degrading system performance as highlighted by preliminary experimental tests.

The first 64 bytes of the training packets, considered 2 by 2, are called attributes. An attribute is defined as A_i and its value is called v_i . The training phase of this module consists of constructing a random set of conditional rules of the form

$$\begin{aligned} \textit{if} \quad & A_1 = v_1, A_2 = v_2, \dots, A_k = v_k \\ \textit{then} \quad & A_{k+1} \in \mathbf{V} = \{v_{k+1}, v_{k+2}, \dots, v_{k+r}\} \end{aligned}$$

where $A_1 = v_1, A_2 = v_2, \dots, A_k = v_k$ is the antecedent, while $A_{k+1} \in \mathbf{V} = \{v_{k+1}, v_{k+2}, \dots, v_{k+r}\}$ is the consequent.

At first the module selects a random subset S of the training data set, composed of one hundred packets. At this point the module randomly selects packets two by two up to obtain one thousand pairs and then up to four rules, which must be satisfied by both packets, are generated for each pair. In this way a maximum of four thousand rules are generated.

For each rule the system also computes the value of n/r , where n is the number of packets that satisfy the antecedent, while r is the number of values the attribute can assume. Subsequently the module analyzes the computed rules to discard the redundant ones. The system also discards rules with a low value of n/r , because a low number of packets matches these rules or the consequent can assume a high number of different values. Hence in both cases the rule does not significantly describe the training data set. Finally the module tests the remaining rules on the complete training set, discarding those who generate false alarms.

During the detection phase, the TCP/SLR analyzes the first 64 bytes of each packet and checks if they break some rules. If so, the module calculates an anomaly score for each broken rule. The total anomaly score is the sum of all those calculated for each rule.

The anomaly score is calculated as $t \cdot \frac{n}{r}$, where t is the time since the rule was last violated. If an anomaly score has been calculated, the module generates, for that packet, a security event which is passed to the event correlator.

As for the previous module, this approach is very efficient in terms of memory and processing requirements. Experimental results have shown that, after the pruning procedure, a total of about one hundred rules are generated, which means that the memory occupation is of the order of few KBs. Moreover, once

the rule set has been constructed, the system works exactly as a rule-based IDS, which is suitable for on-line detection.

D. ICMP/SLR Module

ICMP/SLR relies on the same procedure as the previous module. Because of its self-learning nature, the algorithm is effective both in revealing TCP as well as ICMP attacks.

E. UDP Module

This module is much simpler than all the previous ones because UDP headers do not carry enough information to allow a proper classification of the packets.

Therefore, the UDP module only performs the check on well-known ports, revealing an anomaly if a packet is sent to a port never used during the training phase. This kind of approach can be useful for the detection of probe attacks.

F. Correlators and Classifiers

Inputs to these modules are the security events generated by the detection modules. The correlators perform some operations aimed to reduce the false alarms number.

For each security event, they check if an analogous indication (related to the same hypothetical attack) was received in the previous Y seconds, where Y is set by the network administrator. If so, the security event is dropped, otherwise an alarm is generated.

Moreover, in case of TCP packets, if both the TCP/SLR and the Markov modules have generated a security event, the anomaly score is the sum of those contained in the two security events.

The resulting security alarms, containing the time-stamp, the destination address and the anomaly score, are then sent to the classifiers, which generate an alarm if the anomaly score exceeds a threshold, that can be set by means of Monte Carlo simulation.

III. PERFORMANCE ANALYSIS

In this section we discuss the tests carried out to evaluate the performance of SLIDS.

The results highlight that the combined use of different modules detect more anomalies than classic IDSs. In particular, for TCP packets, the two approaches, TCP/SLR and TCP/Markov, detect different anomalies, thus the combined use of both of them takes to a significant improvement in the IDS performance.

The results have been obtained using the DARPA evaluation data set. The DARPA evaluation project [10][11][12], carried out in 1998 and 1999, represents the first formal, repeatable, and statistically-significant evaluation of IDSs.

The results shown in this subsection have been obtained with the 1999 evaluation data set, composed of 5 weeks traffic, collected over an emulated network scenario. Unfortunately, the features of the collected traces are quite different from those of today's traffic, since they are the result of a quite old project. Nevertheless we used this data in our test, because it represents a benchmark for IDS performance evaluation.

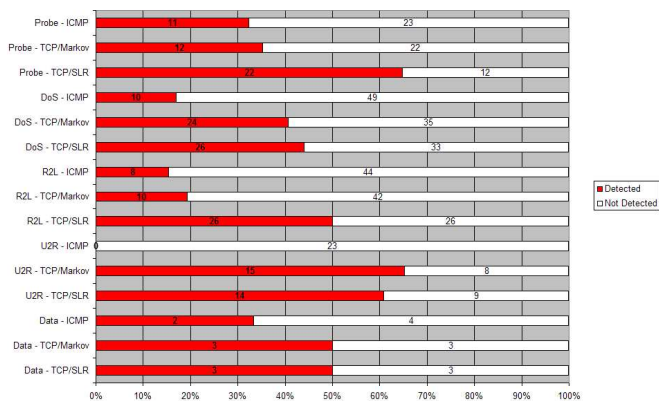


Fig. 4. Attacks detected by the different modules

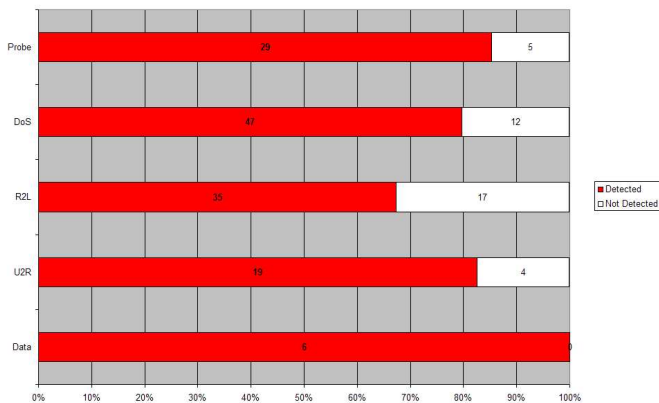


Fig. 5. Attacks detected by SLIDS

The alarm threshold in SLIDS has been set, by means of Monte Carlo simulation, so as to obtain a false alarm rate of about 10%. Figures 4 and 5 show the results of our tests; the values inside the graph bars represent the number of detected/not detected attacks. The first figure shows attacks detected by the different modules, while Fig.6 shows the attacks revealed by SLIDS as a whole. As it can be seen, the use of several modules, working in parallel on the packets, allows us to obtain much better performance than using only a single approach. As an example, let us consider the Data attacks: the use of a single module allows to detect at most one half of the attacks, while the complete system is able to detect all the Data attacks. The same consideration is still valid for the detection rate of all the attacks: the maximum achievable with a single module (TCP/SLR) is 52.3%, while using the whole system it is about 78%.

To evaluate the performance we used a Receiver Operating Characteristic (ROC), which plots true positive rate vs. false positive rate. Fig. shows that the system can achieve really good results, detecting the 80% of the attacks with only 10-15% of false alarms.

Another test session has been carried out using actual traffic traces, captured within our testbed network in November 2005. This analysis was performed to evaluate the capability of SLIDS to face with current security attacks.

The ROC curve obtained in this test session is analogous to the one shown in Fig.6. The test session carried out on

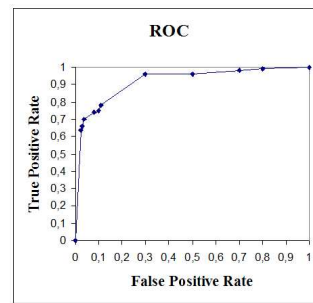


Fig. 6. ROC curve (DARPA data set)

real traffic has confirmed the good performance obtained on the DARPA dataset. The system has detected the 80% of the attacks with a false alarm rate of 15%.

IV. CONCLUSIONS AND FURTHER WORKS

In this paper we presented the design and the implementation of SLIDS, an Anomaly based NIDS. Such IDS has been realized taking into account extensibility and flexibility features. The modularity of the system allows the network manager to customize SLIDS, according to the behaviour of the network. Moreover, parallel use of several modules permits to identify a wide range of attacks.

Experimental results highlighted the effectiveness of SLIDS with both the DARPA evaluation data and data collected within our testbed in presence of up-to-date attacks.

Future work will be mainly devoted to the enhancement of the UDP module and of the correlator block. The idea is to realize a module which is able to find some correlations among the alarms generated by the different modules, so as to perform a more effective detection of the attacks.

ACKNOWLEDGMENTS

This work was partially supported by Euro-NGI Network of Excellence funded by the European Commission.

REFERENCES

- [1] *Packet Storm Home Page*, <http://www.packetstormsecurity.org>
- [2] *AntiServer, Denial Of Service Download Area*, <http://www.antiserver.it/Denial-of-Service>
- [3] Denning D.E., *An Intrusion-Detection Model* IEEE, Transactions on Software Engineering, February 1987.
- [4] Stallings, W., *Cryptography and Network Security. Principles and Practice*, Prentice Hall, 2003.
- [5] G. Giacinto, *Intrusion Detection Systems for Computer Networks*, February 2005.
- [6] Roesch, M., *Snort - Lightweight Intrusion Detection for Network*, Proc. USENIX Lisa, 1999.
- [7] Nong Ye, et al., *Robustness of the Markov-Chain for Cyber-Attack Detection*, IEEE, Transactions on Reliability, March 2004.
- [8] Jha, S. et al., *Markov Chains, Classifiers, and Intrusion Detection*, IEEE 14th Computer Security Foundations Workshop (CSFW), 2001.
- [9] Mahoney, M., *A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic*.
- [10] MIT, Lincoln Laboratory, *DARPA evaluation intrusion detection*, <http://www.ll.mit.edu/IST/ideval/>
- [11] Lippmann, R., et al., *The 1999 DARPA Off-Line Intrusion Detection Evaluation*, DARPA Web Site, 2000.
- [12] Haines, J.W., et al., *1999 DARPA Intrusion Detection Evaluation: Design and Procedures*, DARPA Web Site, 2001.