Intrusion Detection Systems

Advances in Information Security

Sushil Jajodia
Consulting Editor
Center for Secure Information Systems
George Mason University
Fairfax, VA 22030-4444
email: jajodia@gmu.edu

The goals of the Springer International Series on ADVANCES IN INFORMATION SECURITY are, one, to establish the state of the art of, and set the course for future research in information security and, two, to serve as a central reference source for advanced and timely topics in information security research and development. The scope of this series includes all aspects of computer and network security and related areas such as fault tolerance and software assurance.

ADVANCES IN INFORMATION SECURITY aims to publish thorough and cohesive overviews of specific topics in information security, as well as works that are larger in scope or that contain more detailed background information than can be accommodated in shorter survey articles. The series also serves as a forum for topics that may not have reached a level of maturity to warrant a comprehensive textbook treatment.

Researchers, as well as developers, are encouraged to contact Professor Sushil Jajodia with ideas for books under this series.

Additional titles in the series:

VULNERABILITY ANALYSIS AND DEFENSE FOR THE INTERNET edited by Abhishek Singh; ISBN: 978-0-387-74389-9

BOTNET DETECTION: Countering the Largest Security Threat edited by Wenke Lee, Cliff Wang and David Dagon; ISBN: 978-0-387-68766-7

PRIVACY-RESPECTING INTRUSION DETECTION by Ulrich Flegel; ISBN: 978-0-387-68254-9

SYNCHRONIZING INTERNET PROTOCOL SECURITY (SIPSec) by Charles A. Shoniregun; ISBN: 978-0-387-32724-2

SECURE DATA MANAGEMENT IN DECENTRALIZED SYSTEMS edited by Ting Yu and Sushil Jajodia; ISBN: 978-0-387-27694-6

NETWORK SECURITY POLICIES AND PROCEDURES by Douglas W. Frye; ISBN: 0-387-30937-3

DATA WAREHOUSING AND DATA MINING TECHNIQUES FOR CYBER SECURITY by Anoop Singhal; ISBN: 978-0-387-26409-7

SECURE LOCALIZATION AND TIME SYNCHRONIZATION FOR WIRELESS SENSOR AND AD HOC NETWORKS edited by Radha Poovendran, Cliff Wang, and Sumit Roy; ISBN: 0-387-32721-5

PRESERVING PRIVACY IN ON-LINE ANALYTICAL PROCESSING (OLAP) by Lingyu Wang, Sushil Jajodia and Duminda Wijesekera; ISBN: 978-0-387-46273-8

SECURITY FOR WIRELESS SENSOR NETWORKS by Donggang Liu and Peng Ning; ISBN: 978-0-387-32723-5

MALWARE DETECTION edited by Somesh Jha, Cliff Wang, Mihai Christodorescu, Dawn Song, and Douglas Maughan; ISBN: 978-0-387-32720-4

ELECTRONIC POSTAGE SYSTEMS: Technology, Security, Economics by Gerrit Bleumer; ISBN: 978-0-387-29313-2

Additional information about this series can be obtained from http://www.springer.com

Intrusion Detection Systems

Edited by

Roberto Di Pietro
Università di Roma Tre
Italy

and

Luigi V. Mancini
Università di Roma "La Sapienza"
Italy

Springer

Editors:
Roberto Di Pietro
Università di Roma Tre
Dip.to di Matematica
L.go S. Leonardo Murialdo, 1
00146 - Roma
Italy
dipietro@di.uniroma1.it

Luigi V. Mancini
Università di Roma "La Sapienza"
Dip.to di Informatica
Via Salaria, 113
00197 - Roma
Italy
lv.mancini@di.uniroma1.it

Series Editor:
Sushil Jajodia
George Mason University
Center for Secure Information Systems
4400 University Drive
Fairfax VA 22030-4444, USA
jajodia@gmu.edu

ISBN: 978-0-387-77265-3 e-ISBN: 978-0-387-77266-0

Library of Congress Control Number: 2008924099

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Foreword

In our world of ever-increasing Internet connectivity, there is an on-going threat of intrusion, denial of service attacks, or countless other abuses of computer and network resources. In particular, these threats continue to persist even on account of the flaws of current commercial Intrusion Detection Systems (IDS). These flaws are the result of the concurrence of several shortcomings such as excessive resource requirements, limited precision, lack of flexibility, and scope limitation. The aim of this book is to present the contributions made by both academia and industry to thwart those threats. In particular, this book includes a selection of the best research outcomes on intrusion detection of the Italian FIRB WEB-MINDS project (Wide scalE, Broadband MIddleware for Network Distributed Systems), plus several international contributions.

The goal of an Intrusion Detection System (IDS) is to monitor network assets in order to detect misuse or anomalous behavior. Several types of IDS have been proposed in the literature, and they can be divided into two broad classes, i.e., network based (NIDS) and host based (HIDS). The former tries to detect any attempt to subvert the normal behavior of the system by analyzing the network traffic, while the latter is intended to act as the last line of defense. The host based IDS strives to detect intrusions by analyzing the events on the local system where the IDS is being run. Host based IDSs are generally classified into two categories: *anomaly detection* and *misuse detection*. Misuse detection systems try to identify behavior patterns that are characteristic of intrusions, but this can be difficult if an attack exhibits *novel* behavior, as it may when attackers develop new strategies. On the other hand, anomaly detectors try to characterize the *normal* behavior of a system so that any deviation from that behavior can be labeled as a possible intrusion. Anomaly detection assumes that misuse or intrusions are strongly correlated to abnormal behavior exhibited by either the user or by the system itself. Anomaly detection approaches must first determine the normal behavior of the object being monitored, then use deviations from this baseline to detect possible intrusions.

Unlike the behavior of a human user or the behavior of network traffic, the behavior of a program ultimately stems from a series of machine instructions. Thus, intrusions can be detected as deviations from normal program behavior. The ques-

tion, however, is how to characterize normal program behavior so as to minimize both false positives (false alarms caused by legitimate changes in program behavior) and false negatives (missed intrusions caused by attackers that mimic benign users).

As can be seen from the short description of the main issues in IDS, the area is far from being fully investigated. Furthermore, there is a need to develop new approaches that could bridge the gap between the flexibility and the precision required by IDS and current solutions. In particular, this book collects several contributions from the research units that are active in the area of Intrusion Detection. The objectives of this book are threefold: to provide an up-date on the state of the art of the research in this area; to indicate possible research directions, and to make practitioners and the industry aware of the most promising IDS technologies.

Contributions and roadmap

Chapter 1 deals with data showing that payload-based approaches are becoming the most effective methods to detect attacks. The chapter describes some approaches for Anomaly-based network detection systems (NIDSs) which focus on packet headers, payload, or a combination of both. The results of the proposed approaches are also discussed. This chapter paves the way for further studies into this segment of interest.

Chapter 2 proposes a methodology for the synthesis of the behavior of an application program in terms of the set of system calls invoked by the program. The methodology is completely automated except for the description of the high level specification of the application program which is demanded of the system analyst. The technology (VSP/CVS) employed for such synthesis minimizes the efforts required to code the specification of the application, thus making the proposal viable for industrial products as well.

Chapter 3 introduces a sophisticated mechanism that can be used to model profiles i.e., Hierarchical Hidden Markov Models. Consequently, abstract process behavior corresponds to probabilistic regular expressions. A learning algorithm built over this abstraction mechanism is proposed; such an algorithm can automatically infer a profile from a set of traces of the process behavior. This chapter pushes forward the application of sophisticated statistical tools and shows promising research directions.

Chapter 4 stems from the observation that the multiple, possibly complementary security devices that are used to defend against computer and network attacks, and that include intrusion detection systems (IDSs) and firewalls are widely deployed to monitor networks and hosts, may flag alerts when suspicious events are observed. Alert correlation focuses on discovering the existing relationships among individual alerts; this chapter gives an overview of current alert correlation techniques. Finally, it also introduces privacy issues in the IDS field.

Chapter 5 describes recent research on attack graphs that represent the known attack sequences which can be used by attackers to penetrate computer networks. This chapter will show how attack graphs can be used to compute actual sets of hardening measures for the safety of given critical resources. Note that the degree

of safety that is provided within the approach proposed in this chapter is tunable and guaranteed.

Chapter 6 introduces a new mechanism for adapting the security policy of an information system according to the threat it receives, and hence its behavior and the services it offers. The proposed mechanism bridges the gap between preventive security technologies and intrusion detection, and builds upon existing technologies to facilitate formalization on one hand, and deployment on the other hand.

Lastly, Chapter 7 identifies the fundamental requirements that must be satisfied to protect hosts and routers from any form of Distributed DoS (DDoS). Then, a framework that satisfies most of the identified requirements is proposed. It appropriately combines Intrusion Detection and Reaction techniques, and comprises a number of components that actively co-operate to effectively react to a wide range of attacks.

Roma, January 2008

Roberto Di Pietro
Dipartimento di Matematica
Università di Roma Tre
L.go S. Leonardo Murialdo, 1
00146 Roma, Italy
ricerca.mat.uniroma3.it/users/dipietro
dipietro@mat.uniroma3.it

Luigi V. Mancini
Dipartimento di Informatica
Università di Roma "La Sapienza"
Via Salaria, 113
00198 Roma, Italy
www.di.uniroma1.it/mancini
lv.mancini@di.uniroma1.it

Contents

Approaches in Anomaly-based Network Intrusion Detection Systems

Damiano Bolzoni and Sandro Etalle

Abstract Anomaly-based network intrusion detection systems (NIDSs) can take into consideration packet headers, the payload, or a combination of both. We argue that payload-based approaches are becoming the most effective methods to detect attacks. Nowadays, attacks aim mainly to exploit vulnerabilities at application level: thus, the payload contains the most important information to differentiate normal traffic from anomalous activity. To support our thesis, we present a comparison between different anomaly-based NIDSs, focusing in particular on the data analyzed by the detection engine to discover possible malicious activities. Furthermore, we present a comparison of two payload and anomaly-based NIDSs: PAYL and POSEIDON.

1 Introduction

Network intrusion detection systems (NIDSs) are considered an effective second line of defense against network-based attacks directed at computer systems [1, 2], and – due to the increasing severity and likelihood of such attacks – are employed in almost all large-scale IT infrastructures [3].

There exist two main types of intrusion detection systems: signature-based (SBS) and anomaly-based (ABS). SBSs (e.g. Snort [4, 37]) rely on pattern-matching techniques: they contain a database of *signatures* of known attacks and try to match these signatures against the analyzed data. When a match is found, an alarm is raised. On the other hand, ABSs (e.g. PAYL [6]) first build a *statistical model* describing the normal network traffic, then flag any behaviour that significantly deviates from the model as an attack.

University of Twente,
P.O. Box 2100, 7500 AE Enschede, The Netherlands
e-mail: {damiano.bolzoni,sandro.etalle}@utwente.nl

Intuitively speaking, anomaly-based systems have the advantage that (unlike signature-based systems) they can detect zero-day attacks, since novel attacks can be detected as soon as they take place. On the other hand, ABSs (unlike SBSs) require a training phase and a careful setting of the detection threshold (more about this later), which makes their deployment more complex.

Contribution

In this paper, we discuss anomaly-based systems, focusing in particular on a specific kind of them: the ABSs *payload-based*. We argue that payload-based systems are particularly suitable to detect advanced attacks, and we describe in detail the most prominent and the most recent of them: respectively Wang and Stolfo's PAYL [6] and our POSEIDON [7].

2 Anomaly-Based Intrusion Detection Systems

In this section, we present the basic working of anomaly-based systems, and we explain the different kinds of ABSs existing. The thesis we try to substantiate here is that, because of the kind of attacks that are carried out nowadays, packed-based and payload-based systems are becoming the most interesting kind of ABS.

Anomaly-based NIDSs can be classified according to:

1. the underlying algorithm they use,
2. whether they analyze the features of each packet singularly or of the whole connection, and
3. the kind of data they analyze. In particular, whether they focus on the packet headers or on the payload.

Regarding the underlying algorithm, Debar et al. [8, 2] define four different possible approaches, but only two of them have successfully employed in the last decade: algorithms based on statistical models and those based on neural networks. The former is the most widely used: according to Debar et al. [8] more than 50% of existing ABSs is statistic-based. In these systems, the algorithm (during the so-called *training phase*) first builds a statistical model of the – legitimate, attack-free – network behaviour; later (in the *detection phase*), the input data is compared to the model using a distance function, and when the distance measured exceeds a given *threshold*, the input is considered *anomalous*, i.e., it is considered an attack. ABSs based on neural networks work in a similar way (they also have a training phase, a detection phase and a threshold), but instead of building a statistical model, they train a neural network which is then in charge of recognizing regular traffic from anomalous one. Good training is of crucial importance for the effectiveness of the system: in particular, the data used in the training phase should be (at least in principle) as *attack-free* as possible: training data must reflect as much as possible the

normal system data flow, not including malicious activity. Furthermore, the training phase should be long enough to allow the system to build a faithful model: a too short training phase could lead to a (too) coarse data classification, which – in the detection phase – translates into flagging legitimate traffic too often as anomalous (false positives).

Concerning feature (b) the distinction one has to make is between *packet-oriented* and *connection-oriented* ABSs. A packet-oriented system uses a single packet as minimal information source, while a connection-oriented system considers features of the whole communication before establishing whether it is anomalous or not. Theoretically, a connection-oriented system could use as input the content (payload) of a whole communication (allowing – at least in principle – a more precise analysis), but this would require a long computational time, which would seriously limit the throughput of the system. In practice, connection-oriented systems typically take into account the number of sent/received bytes, the duration of the connection and layer-4 protocol used. According to Wang and Stolfo's benchmarks [6], payload-based ABSs do not show a sensible increase in performance when they also reconstruct the connection, instead of just considering the packets in isolation. In practice, most ABSs are packet-oriented (see also Table 1).

The last, more practically relevant distinction we can make is between *header-based* and *payload-based* system. Header-based systems consider only packet headers (layer-3 and, if present, layer 4 headers) to detect malicious activities; payload-based systems analyze the payload data carried by the layer-4 protocol; there are also hybrid systems which mix information gathered observing packet headers and (if present) layer-4 payload data. We are going to elaborate on this distinction in the rest of this section. Before we do so, we want to present a table reporting some of the most important ABSs: we select the systems which have been benchmarked with public data sets (either DARPA 1998 [9] or DARPA 1999 [10] data sets, which contain a full dump of the packets, or the KDD 99 [11] data set, which contains only connection meta data).

iSOM [12] uses a one-tier architecture, consisting of a Self-organizing Map [13], to detect two attacks in the 1999 DARPA data set: the first attack against the SMTP service and the other attack against the FTP service. Intel information from the connection meta data once it has been reassembled. PHAD [14] combines 34 different values extracted from the packet headers. MADAM ID [15] extracts information from audit traffic and builds classification models (specifically designed for certain types of intrusion) using data mining techniques. The system indicated by SSAD [16] (Service Specific Anomaly Detection) combines different information such as type, length and payload distribution (computing character frequencies and aggregating then them into six groups) of the request. PAYL [6] and POSEIDON [7] detect anomalies only looking at the full payload. Table 1 summarizes the properties of some ABSs.

System	Detection Engine	Semantic Level	Analyzed Data
iSOM	NN	PO + CO	Meta data
IntelligentIDS	NN	CO	Meta data
PHAD	S	PO	H
MADAM ID	S	CO	Meta data
SSAD	S	PO	H + P
PAYL	S	PO	P
POSEIDON	NN + S	PO	P

Table 1 Anomaly-based systems: NN stands for Neural Networks, S for Statistical model, PO is Packet-Oriented while CO is Connection-Oriented, H and P stand for Headers and Payload respectively

2.1 Payload-based vs header-based approaches

We now elaborate the differences in effectiveness between payload-based and header-based systems. We begin by showing some examples of attacks that can be detected by the systems of one kind, but not by the system of the other kind.

Attacks detectable by header-based systems

Example 0.1. The teardrop exploit [17] is a remote Denial of Service attack that exploits a flaw in the implementation of older TCP/IP stacks: some implementations of the IP fragmentation re-assembly code on these platforms do not properly handle overlapping IP fragments. Figure 1 shows how the attacks takes place: the attacker sends fragmented packets forged so that they overlap each other when the receiving host tries to reassemble them. If the host does not check the boundaries properly, it will try to allocate a memory block with a negative size, causing a kernel panic and crashing the OS.

IDSs can find this attack only by looking for two specially fragmented IP datagrams, analyzing the headers. This attack exploits a vulnerability at the network layer.

Example 0.2. The Land attack [17] is a remote Denial of Service attack that is effective against some older TCP/IP implementations: the attack involves sending a spoofed TCP SYN packet (connection initiation) with the same source and destination IP address (the target address) and the same (open) TCP port as source and destination.

Some implementations cannot handle this theoretically impossible condition, causing the operating system to go into a loop as it tries to resolve a repeated connections to itself. IDSs detect this attack by looking at packet headers, since TCP SYN segments do not carry any payload. This attack exploits a vulnerability at the transport layer.

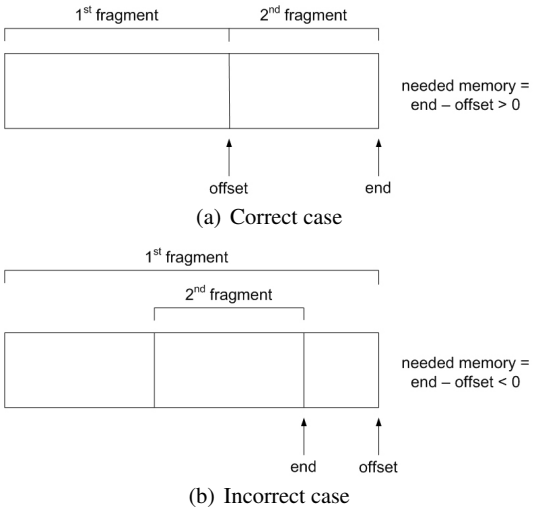

Fig. 1 A correct and incorrect case of fragment management by the network layer

Attacks detectable by payload-based systems

Example 0.3. SQL injection is a technique that exploits vulnerabilities of (web-based) applications which are interfaced to an SQL database: if the application does not sanitize potentially harmful characters first [18], an intruder can *inject* an SQL query in the database, and force the database to output sensitive data (e.g. user passwords and personal details) from database tables, without being authorized. SQL Injections are considered a serious threat and are constantly listed in the "Top Ten Most Critical Web Application Security Vulnerabilities" [19] by "The Open Web Application Security Project".

For instance, the following HTTP request is actually a well-known attack [20] against the Content Management System (CMS) PostNuke [21] that can be used to get hold of the user passwords:

```
http://[target]/[postnuke_dir]/modules.php?op=modload&
name=Messages&file=readpmsg&start=0%20UNION%20SELECT%20
pn_uname,null,pn_uname,pn_pass,pn_p
```

When such an attack is carried out successfully, the output (a database table) is significantly different from the HTML page usually rendered. This attack exploits a vulnerability at the application layer.

Example 0.4. The PHF attack [22] exploits a badly written CGI script to execute commands with the privilege level of the HTTP server user. Any CGI program which relies on the function *escape_shell_cmd()* may be vulnerable: this vulnerability is manifested by the *phf* program that is distributed with the example code for the Apache web server.

```
http://[target]/cgi-bin/phf?Qalias=x\%0A/bin/cat\%20/etc/passwd
```

The issued request will provide the attacker with the list of system users.

To detect a PHF attack, an intrusion detection system can monitor HTTP requests for invocations of the *phf* command with arguments that specify commands to be run. This attack exploits a vulnerability at the application layer.

Some Conclusions

The above examples reflect the unsurprising fact that header-based systems are more suitable to detect attacks directed at vulnerabilities of the network and transport layers; we can also include in this category all the probing techniques used before a real attack takes place (port/host scanning). On the other hand, payload-based systems are more suitable to identify attacks trying to exploit vulnerabilities at the application level, where sensitive data are stored and most of the systems can be subverted.

Here, we must take notice of the trend that shows that this second kind of attack is increasingly gaining importance: this is due both to the large success of web-based services, and to the fact that attacks at network and transport layers are becoming rare. Because of this, we believe that payload-based system will be increasingly useful in the future. We believe that this trend not only favors payload-based IDS wrt header-based ones, but also anomaly-based systems wrt signature-based ones (we are going to elaborate on this in the conclusions).

We should not forget, however, that payload-based NIDSs cannot function properly in combination with applications or application protocols (e.g. SSH and SSL) which apply data encryption, unless the encryption key is provided. A possible solution to this makes use of a host-based component to access data once they have been decrypted, but this causes an overhead on the monitored host. This problem is going to grow in importance when IPv6 will gradually replace IPv4: in fact, one of the main design issues of IPv6 is the authentication and confidentiality of data (through cryptography).

As we mentioned before, header-based approaches alone do not represent a valid solution to detect modern attacks. In this case, we believe that further research on approaches based on the analysis of connection meta data (connection duration, sent/received bytes, etc.), which are not affected by cryptographic content, should be conducted to verify their application in real environment, since they show to be quite effective in detecting malicious activity with standard data set such as DARPA 1999 and KDD 99.

3 Setting up an ABS

As we have seen, to determine whether a certain input is anomalous or not, an ABS compares it to the model it has: if the distance between some function of the input

and the model exceeds a given *threshold*, the input is considered anomalous. This shows that the quality of the model and the value of the threshold have a direct influence on the effectiveness of the ABS. Both the model and the threshold are determined during the system setup (though the threshold could be refined successively). In the rest of this section we elaborate on these two crucial aspects.

Before we do so, we define that the effectiveness of a NIDS is determined by its *completeness* and its *accuracy*.

- $completeness = TP/(TP+FN)$
- $accuracy = TP/(TP+FP)$

Here, *TP* is the number of true positives, *FN* is the number of false negatives and *FP* is the number of false positives raised during a given time frame.

The number of false positives per hour determines the workload of IT personnel: with a hundred thousands input packets per hour (which is a reasonable figure for a web server), a false positive rate of 1% still determines a thousand false positive per hour, which is more than a typical company can afford to handle. When a system raises too many false positives, then system managers tend to ignore alerts raised. As a matter of fact, a high false positive rate is generally cited as one of the main disadvantages of ABSs.

3.1 Building the Model

The model an ABS refer to should reflect the behaviour of the system *in absence of attacks*, otherwise the ABS may fail to recognize an attack as such. Because of this, the ABS should be trained with a *clean* data set. However, obtaining such a data set is difficult in practice: a casual dump of network traffic is likely to be *noisy*, i.e., to contain attacks.

The standard way to deal with this is by cleaning the data set by manual inspection. This relies completely on the expertise of the IT personnel which must analyze a large amount of data. Clearly, this approach it is labour intensive, also because the model of the ABS needs to be updated regularly to adapt to environment changes. The manual inspection can be aided by an automatic inspection using a signature-based IDS, which can pre-process the training data and discover well-known attacks (e.g. web-scanners, old exploits, etc.). A signature-based IDS however will not detect all attacks in the data, leaving the training set with a certain amount of noise.

Nevertheless, we believe that it is possible to clean automatically the data set in such a way that the resulting model is a faithful representation of the legitimate network traffic. Intuitively, we apply an anomaly-based intrusion detection algorithm in which the threshold is set in such a way that we are sure of catching all attacks: this is possible because noise typically forms a small percentage of the total data [23], moreover, its content is typically very different from the content of regular data [24, 25]. The fact that we eliminate also a percentage of the legitimate data with them is – at this stage – not a serious concern.

In our experience clustering techniques from data mining can be quite useful to obtain a good training set. Clustering is the classification of similar objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) shares some common trait, often according to a defined distance measure. Past research applies this concept to detect network attacks [26], but because of the intrinsic limitations of the classification algorithms, it does not achieve a high detection rate. There exists a large number of clustering algorithms which deal with different data types: in our case, we can use clustering algorithms to classify the training data and then disregard the data belonging to the clusters which are not dense enough to be considered significant for the training the model.

3.2 Setting the threshold

The value of the threshold has an obvious and direct impact on the accuracy and completeness: a low threshold yields a high number of alarms, and therefore a low false negative rate, but a high false positive rate. On the other hand, a high threshold yields a low number of alarms in general (therefore a high number of false negatives, but a low number of false positives). Therefore, setting the threshold requires skill: its "optimal" value depends on environment monitored and on the distribution of the training data.

In our ATLANTIDES system [27] we introduce a simple heuristics to set the threshold value when using a *noisy* data set: our experiments show that setting the threshold at $\frac{3t_{max}}{4}$, usually yields reasonably good results; here t_{max} is the maximum distance between the analyzed data and the model observed during the training phase.

4 PAYL and POSEIDON

In this section, we present PAYL and POSEIDON, the first is recognized as the most prominent payload-based anomaly-based NIDS, POSEIDON is the improvement on PAYL we have developed.

4.1 PAYL

PAYL (Wang and Stolfo [6]) is a system based on a 2-step algorithm. First, packets are classified according to the payload length, then an n-gram [28] analysis is applied to the payload. PAYL works as follows (see Appendix 6.1 for the pseudocode).

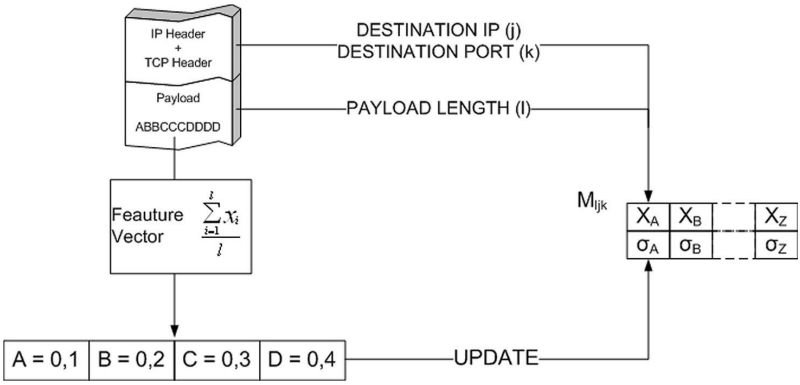

Fig. 2 PAYL architecture

During the *training phase*, the training set T is split into a number of disjoint subsets T_{ljk}, where each T_{ljk} contains the packets of length l, destination IP address j and TCP port k. Then PAYL creates statistical models M_{ljk} of each T_{ljk} by first carrying out n-gram analysis [28] of size 1 on each packet of T_{ljk}, and then incrementally storing in M_{ljk} a feature vector containing the average byte frequency distribution together with the variance value of each frequency.

During the *detection phase*, the same values are computed for incoming packets and then compared to model values: a significant difference from the norm produces an alert. To compare models, PAYL uses a simplified version of the Mahalanobis distance, which has the advantage of taking into account not only the average value but also its variance and the covariance of the variables measured.

The maximum amount of space required by PAYL is: $p * l * k$, where p is the total number of ports monitored (each host may have different ports), l is the length of the longest payload (payload length can vary between 0 and 1460 in a Local Area Network infrastructure based on Enthernet) and k is a constant representing the space required to keep the mean and the variance distribution values for each payload byte (PAYL uses a fixed value of 512). To reduce the otherwise large number of models to be computed, PAYL collapses similar models. After comparing two neighbouring models using the Manhattan distance, if the distance is smaller than a given threshold t, models are merged: the means and variances are updated to produce a new combined distribution. This process is repeated until no more models can be merged. Experiments with PAYL show [6] that a reduction in the number of model of up to a factor of 16 can be achieved.

4.2 POSEIDON

The Achille's heel of PAYL lies in the classification phase: a good classification algorithm should produce clusters with high intra-class similarity and high inter-

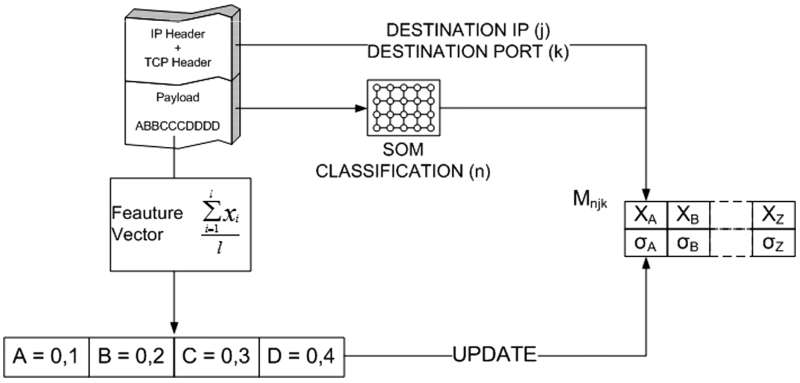

Fig. 3 POSEIDON architecture

class dissimilarity. Using the payload length information as primary method to divide packets into clusters, similar payloads could be classified in different clusters because they present a small difference in their length; on the other hand, dissimilar payloads could be classified in the same cluster because they present the same length. This can affect negatively the subsequent n-gram analysis.

To solve this problem, we designed POSEIDON. The design goal was to obtain a good – unsupervised – classification method for network packets (which are high-dimensional data). This is a typical clustering problem which can be tackled using neural networks in general and Self-Organizing Maps (SOM) [13] in particular. SOMs have been widely used in the past both to classify network data and to find anomalies. in POSEIDON, we use them for pre-processing.

The POSEIDON architecture combines a SOM with a modified PAYL algorithm and works as follows. The SOM is used to pre-process each packet, afterwards PAYL uses the classification value given by the SOM instead of the payload length. Instead of using model M_{ijk}, PAYL uses the model M_{njk} where j and k are the usual destination address and port and n is the classification derived from the neural network. Then, mean and variance values are computed as usual.

Having added a SOM to the system we must allow for both the SOM and PAYL to be trained separately. Regarding resource consumption, we have to revise the required amount of space to: $p * n * k$, where the new parameter n indicates the amount of SOM network nodes. Our experiments show that POSEIDON allows to reduce the number of models needed (wrt PAYL) by a factor of 3 while achieving a better accuracy and completeness.

How the SOM works

Self-organizing maps are topology-preserving single-layer maps. SOMs are suitable to analyze high-dimensional data and belong to the category of competitive learning networks [13]. Nodes are also called *neurons*, to remind us of the artificial intelli-

gence nature of the algorithm. Each neuron n has a *vector of weights* w_n associated to it: the dimension of the weights arrays is equal to the length of longest input data. These arrays (also referred as *reference vectors*) determine the SOM behaviour. Appendix 6.2 reports the SOM pseudo-code.

To accomplish the classification, SOM goes through three phases: initialization, training and classification.

Initialization

First of all, some system parameters (number of nodes, learning rate and radius) are determined by e.g. the IDS technician. The number of nodes directly determines the classification given by the SOM: a small network will classify different data inputs in the same node while a large network will produce a too sparse classification. Afterwards, the array of node weights is initialized, usually with random values (in the same range of input values).

Training

The training phase consists of a number of iterations (also called *epochs*). At each iteration one input vector x is compared to all neuron weight arrays w_n with a distance function: the most similar node (also called *best matching unit*, BMU) is then identified. After the BMU has been found, the neighbouring neurons and the BMU itself are updated. The following update parameters are used: the neighbourhood is governed by the *radius* parameter (r) and the magnitude of the attraction is affected by the *learning rate* (α).

During this phase, the map tends to converge to a stationary distribution, which approximates the probability density function of the high-dimensional input data. As the learning proceeds and new input vectors are given to the map, the learning rate and radius values gradually decrease to zero.

Classification

During the classification phase, the first part of the training phase is repeated for each sample: the input data is compared to all the weight arrays and the most similar neuron determines the classification of the sample (but weights are not updated). The winning neuron is then returned.

Conclusions

Our approach to designing a payload-based ABS involves the combination of two different techniques: a self-organizing map and the PAYL architecture. We modify

the original PAYL to take advantage of the unsupervised classification given by the SOM, which then functions as pre-processing stage.

We benchmark our system using the DARPA 1999 data set [10]: this standard data set is used as a reference by a number of researchers (e.g. [14, 12, 6]), and offers the possibility of comparing the performance of various IDSs. Experiments show that our approach achieves a better completeness and accuracy than the original PAYL algorithm: Table 2 reports the results for the four most representative protocols (FTP, Telnet, SMTP and HTTP) inside the data set. Moreover we observe a reduction of models used by PAYL by a factor of 3 when taking advantage of the SOM classification (payload length can vary between 0 and 1460 in a Local Area Network Ethernet-based, while the SOM neural network used in our experiments has less than one hundred nodes).

		PAYL	**POSEIDON**
Number of models used		4065	1622
HTTP	DR	89,00%	100,00%
	FP	0,17%	0,0016%
FTP	DR	95,50%	100,00%
	FP	1,23%	0,93%
Telnet	DR	54,17%	95,12%
	FP	4,71%	6,72%
SMTP	DR	78,57%	100,00%
	FP	3,08%	3,69%
Overall DR with FP $< 1\%$		58,8% (57/97)	73,2% (71/97)

Table 2 Comparison between PAYL and POSEIDON; DR stands for detection rate (completeness), while FP is the false positive rate (accuracy)

5 Conclusions

This paper makes a reasoned case for anomaly payload-based network intrusion detection systems.

Header-Based vs Payload-Based

We have argued that header-based approaches are useful in detecting principally attacks at network level, and that of most modern remote attacks target vulnerabilities at the application layer: thus, we can no longer rely solely on header-based approaches. Moreover, firewalling systems can easily detect (and discard) well-known malicious packets as well. The fact that header-based approaches manage to achieve high detection rates when benchmarked using the DARPA 1999 data set is – as explained by Mahoney and Chan [29] – often due to the fact that it is possible to tune

an IDS which uses some attributes – specifically: remote client address, TTL, TCP options and TCP window size (these present a small range in the DARPA simulation, but have a large and growing range in real traffic) – in such a way that it scores particularly well on this data set. Because most of the attacks to the application significantly differ in content from legitimate traffic (while they are similar when considering header attributes), a payload-based approach is necessary to detect malicious activities.

Signature-based vs Anomaly-Based

The fact that that modern attacks are usually directed to weaknesses of the application rather than weaknesses of the underlying system has another important consequence: as we mentioned in the introduction, next to anomaly-based intrusion detection systems, there exist also *signature-based* systems. These system rely on a set of pre-defined signatures (typically defined by the NIDS manufacturer and updated regularly via Internet). Because of the ad-hoc nature of attacks directed at applications, in which it is often possible to modify the syntax of an attack without changing its semantics, signature-based system are becoming easier to circumvent than anomaly-based systems. For instance, due to the high level of polymorphisms presented by SQL Injection attacks, it is impossible to produce few generic signatures to detect them, and most of these attacks will go unnoticed by a SBS.

We believe that the next generation IDS architectures will have to combine signature-based and anomaly-based approaches, as well as network-based with host-based systems: these architectures, supported by adequate correlation techniques, will combine the advantages offered by each approach, improving at the same time the overall completeness and accuracy.

6 Appendix

6.1 PAYL algorithm

DATA TYPE

feature vector $=$ *RECORD* [
  mean $=$ *array* $[1..256]$ *of Reals*,
    /* average byte frequency */
  stdDev $=$ *array* $[1..256]$ *of Reals*
    /* standard deviation of each */
    /* byte frequency */
]

/* Defining M_{ljk} model */
model $=$ *RECORD* [
  ip $\in \mathbb{N}$, /* destination host address */
  sp $\in \mathbb{N}$, /* destination service port */
  $l \in \mathbb{N}$, /* payload length */
  fv : *feature vector*
]

PAYLOAD $=$ *array* $[1..l]$ *of* $[0..255]$

DATA STRUCTURE

M $=$ *set of finite models*
threshold $\in \mathbb{R}$
  /* numeric value used for anomaly */
  /* detection given by user */

TRAINING PHASE

INPUT:
  ip : *IP address* $\in \mathbb{N}$
  sp : *service port* $\in \mathbb{N}$
  l : *payload length*
  x : *PAYLOAD*

for each m $\in$ *M do*