ATLANTIDES: Automatic Configuration for Alert Verification in Network Intrusion Detection Systems *

Damiano Bolzoni[1], Bruno Crispo[2] and Sandro Etalle[1,3]

[1]University of Twente, Enschede, The Netherlands
[2]University of Trento, Italy
[3]Eindhoven Technical University, The Netherlands

Abstract

We present an architecture designed for alert verification (i.e., to reduce false positives) in network intrusion-detection systems. Our technique is based on a systematic (and automatic) anomaly-based analysis of the system output, which provides useful context information regarding the network services. The false positives raised by the NIDS analyzing the incoming traffic (which can be either signature- or anomaly-based) are reduced by correlating them with the output anomalies. We designed our architecture for TCP-based network services which have a client/server architecture (such as HTTP). Benchmarks show a substantial reduction of false positives between 50% and 100%.

Keywords: Intrusion Detection, Alert Verification, Security Management

1 Introduction

Nowadays, network intrusion-detection systems (NIDSs) are employed in almost all large-scale IT infrastructures [1] as an effective second line of defense against network-based attacks directed to computer systems. Unfortunately, in many cases, the effectiveness of NIDSs can be hindered by the difficulty of their configuration and management. This difficulty depends on the nature of NIDSs that need to detect, react and adapt to attacks. For instance, the detection of a new attack is typically followed by a configuration change (i.e., a new attack signature needs to be added). Furthermore, the variability of these settings, due to reasons such as a change on the security policy of the organization, or an increasing level of desired protection due to new attacks, makes the configuration management even harder. If we consider that such changes always require the direct involvement of IT personnel [16], who must verify the seriosness every single alert, it's easy to understand why this task is labor intensive and also error prone [6].

To make things worse, many of the alarms a NIDS raises are *false positives* (i.e., notifications of attacks that turn out to be false) that however require human intervention before being classified as such. The problem of false positives is a major one since practitioners [16] as well as researchers [3, 10] observe that it is common for a NIDS to raise thousands of alerts per day, most of which are false alerts. Julisch [11] states that up to 99% of total alerts may not be

*This research is supported by the research program Sentinels (http://www.sentinels.nl). Part of this work was carried out during the third author's stay at the University of Trento, supported by the GU-IST project Serenity.

related to real security issues. Notably, false positives affect both *signature* and *anomaly-based* intrusion-detection systems [2]. A high rate of false alerts is – also according to Axelsson [3] – the limiting factor for the performance of an intrusion-detection system. Indeed, a high false positive rate can even be *exploited* by attackers to overload IT personnel, thereby lowering the defenses of the IT infrastructure.

In this article we propose a system that ease NIDS configuration management by sensibly reducing the number of false positives they raise. The main reason why NIDSs raise false positives is that – quoting Kruegel and Robertson [13] – they are often run without any (or very limited) information about the network resources that they protect (i.e., the context). Chaboya et al. [5] agree that the context knowledge (e.g., network and system configurations) can improve significantly alert verification. Unfortunately, building and updating a database of possible configurations or running vulnerability assessment tools (e.g., Nessus [21]) to provide context knowledge is expensive and often not feasible when dealing with complex systems (indeed these activities require additional labor of IT personnel, since the process of using them cannot be completely automated). Most current techniques to improve alert verification either are tailored to specific attacks [9, 24] (e.g., worm-like) or they support only signature-based NIDSs [19].

Here we propose a system to reduce the number of false positive by using correlations between the input and the output traffic. We claim that in many relevant situations, the context information can be obtained by a systematic (and automatic) *anomaly-based* analysis of the output traffic of the monitored network services; we believe this is possible when the output traffic presents some regularities.

To demonstrate our claims, we have developed *ATLANTIDES* (Automatic Configuration for Alert Verification in Network Intrusion Detection Systems) an innovative architecture for easing the configuration management of *any* NIDS (be it signature or anomaly-based) by reducing, in an automatic way, the number of false alarms that the NIDS raises. The main idea behind ATLANTIDES is simple: a successful attack often causes an *anomaly* in the *output* of the service [26], thus modifying the normal output outcome. Detecting this anomaly can help in reducing false alerts. For instance, a successful SQL Injection attackagainst a web application often causes the output of SQL table content (e.g., user/admin credentials) rather than the expected web content.

Our system ATLANTIDES relieves IT personnel from complex deployments by avoiding most of the common side effects of standard configurations (for signature-based NIDSs in particular), when these are left unchanged.

ATLANTIDES, which is completely network-based[1], works by analyzing (using n-Gram analysis [8]) and modeling the normal output payload of the monitored network services that is expected to be sent in response to a client request. This normal output is specific to the site; therefore the derived models reflect – in a way – the network/system context. By correlating the anomalies detected on the output with the alerts raised by the NIDS monitoring the input traffic, we can discard a number of the latter as being false alerts. This way we obtain a system that raises considerably less false positives that the original NIDS, without this correlation system.

Because it is based on output payload analysis, our architecture is designed for TCP-based client/server network services (such as HTTP). Like all (external) payload-based analysis, ATLANTIDES works properly with encrypted data only if the cryptographic keys are provided.

In the past, simple correlations between input and output traffic have already been used to identify possible worm attacks [9, 24]. To the best of our knowledge, ATLANTIDES is the first proposed solution for alert verification that:

[1]It relies only on information gathered over the network, without involving any host-based component.

- works in combination with both signature-based and anomaly-based NIDSs
- operates in a completely automatic way after a quick setup, without any further human involvement (i.e., reducing the IT personnel overload), thus easing NIDS configuration management

We benchmarked ATLANTIDES in combination with the signature-based NIDS Snort [20], as well as in combination with the anomaly-based NIDS POSEIDON [4]. We carried out benchmarks both on a private data set as well as on the common DARPA 1999 data set [14] (for the sake of completeness and to allow duplication of our results, despite criticism [15, 17]). In seven out of eight cases, our benchmarks show a reduction of false positives between 50% and 100%.

2 Background

In this section, we introduce the concepts used in the rest of the paper and explain how false positives arise in signature and anomaly-based systems.

Signature-based systems Signature-based systems (SBSs), e.g., Snort [20], are based on pattern-matching techniques: the NIDS contains a database of *signatures* of known attacks and tries to match these signatures with the analyzed data. When a match is found, an alert is raised. A specific signature must be developed off-line, and then loaded into the database before the system can begin to detect a particular intrusion. One of the disadvantages of SBSs is that they can detect only known attacks: new attacks will be unnoticed till the system is updated, creating a window of opportunity for attackers (and affecting NIDS completeness and accuracy [7]). Although this is considered acceptable for detecting attacks to e.g. the OS, it makes them less suitable for protecting web-based services, because of their *ad hoc* and dynamic nature.

Deployment and management SBSs raise an alert every time that traffic matches one of the signatures loaded into the system: however, legal traffic could be mismatched as malicious. The main reasons why alerts produced by SBSs can turn out to be either false or irrelevant are: (a) SBSs are mostly deployed with *out-of-the-box* configurations, mainly because of lack of specific knowledge, without a signature-selection process. (b) Writing signatures for NIDSs is a thorny task [18], in which it is difficult to find the right balance between an overly specific signature (which is not able to detect a simple attack variation, leading to false negatives) and an overly general one (which will classify legitimate traffic as an attack attempt, raising false positives). (c) The monitored environment is not susceptible to a certain vulnerability. (d) Misconfigured network devices or services producing atypical output (usually, in this case, it is possible to observe recurrent and periodic phenomena).

The first task to accomplish when deploying a SBS is a *tuning* activity: this activity, based on deactivation of unneeded signatures, requires a thorough analysis of the environment by qualified IT personnel. To remain effective, SBSs require regular configuration updates to reflect changes in the environment: new vulnerabilities are discovered daily, new signatures are released regularly, and systems may be patched, thereby (possibly adding or) removing vulnerabilities.

Anomaly-based systems As opposed to SBSs, anomaly-based systems (ABSs) use statistical methods to monitor network traffic. Intuitively, an ABS works by training itself to recognize normal behavior and then raising an alert each time it detects something outside the boundaries of its training. In the training phase, the ABS builds a *model* of the normal network traffic. Later, in the operational phase, the ABS flags as an attack any input that significantly deviates from the model. To determine when an input significantly deviates from the model the ABS uses a *distance* function and a *threshold* set by the user: when the distance between the input and the model exceeds the threshold, an alarm is raised. When compared to SBSs, the ABSs' main advantage is that they can detect zero-day attacks: novel attacks can be detected as soon as they take place. The high false positive rate is generally cited as one of the main disadvantages of anomaly-based systems.

Deployment and management The only task the IT personnel has to accomplish when deploying an ABS is to set the threshold value. The threshold has a direct influence on both false negative and false positive rates [23]: a low threshold (too close to the model) yields a high number of alerts, 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 alerts in general (therefore a high number of false negatives, but a low number of false positives).

The most commonly used tuning procedure for ABSs is finding an optimal threshold value, i.e., the best compromise between a low number of false negatives and a low (or acceptable) number of false positives. This is typically carried out manually by trained IT personnel: different improving steps may be necessary to obtain a good balance between detection and false positive rates.

3 Architecture

ATLANTIDES's architecture (see Figure 1) consists of one external and two internal components. The external component is the NIDS monitoring the incoming traffic. We do not make any assumption about it except that it is capable of raising an alert: ATLANTIDES can work together with any kind of NIDS (signature or anomaly-based).

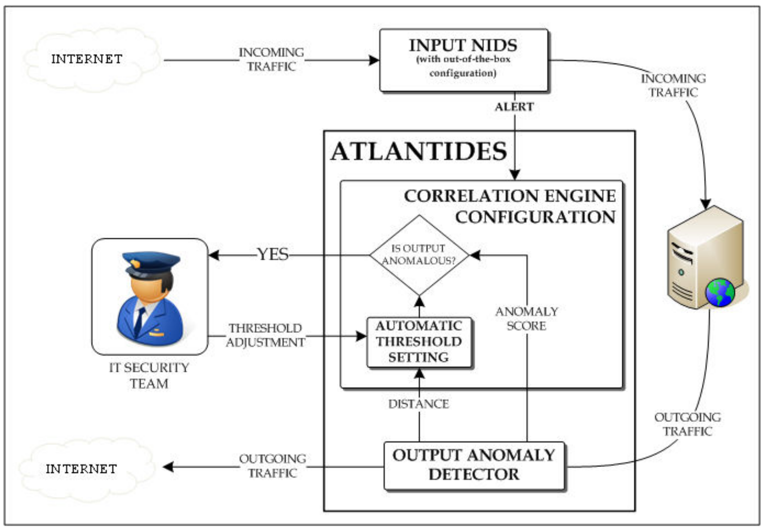

Figure 1: ATLANTIDES's architecture

The first internal component is the output anomaly detector (OAD), which is actually an anomaly detection system monitoring the outgoing traffic: the OAD refers to a statistical model describing the normal output of the system, and flags any behaviour that significantly deviates from the norm as the result of a possible attack.

The second internal component is the correlation engine (CE), which tracks (using stateful-inspectionand correlates alerts related to incoming traffic and raised by the input NIDS with the output produced by the OAD.

ATLANTIDES works as follows (see Figure 1). The input NIDS monitors the incoming traffic while, simultaneously, the OAD (after a training phase) analyses the output of network services. When the input NIDS raises an alert, this is forwarded to the CE, together with the information regarding the communication endpoints (i.e., source and destination IP addresses, source and destination TCP ports as well as sequence numbers and communication status) of the packet that raised the alert. This information is stored in an internal table. At this time, the alert is not considered an incident yet (it is a *pre-alert*) and is not forwarded yet to IT specialists. Next, the CE marks the communication relative to the given endpoints as suspicious and waits for the output of the OAD: if the OAD detects an anomaly in the outgoing traffic related to the tracked communication, then the system considers the alert as an *incident* (i.e., a positive) and the alert is forwarded to the IT specialists for further handling and countermeasure reactions, otherwise it is considered a *false positive* and discarded. The IT personnel can manually set (or adjust) a time value t that the CE waits before dropping an entry from its hash-table, because no output has been produced: during our experiments we set this value at 60 seconds. This time could be critical if an attack results in a large data transfer (but in this case the OAD should detect the anomaly in the transferred data) or in the case where attacker is able to delay server response (although this seems quite difficult to realize and the literature does not provide any example of such an attack). The delay, introduced to allow the OAD to process the data sent back to the client, does not affect the detection itself: in fact, the delay, in the worst case of no output sent at all, is equal to the time value t.

It should be clear from the architecture that ATLANTIDES will never raise more false positives that its input NIDS. In fact, the output of the OAD is evaluated *only* when an alert has already been raised by the input NIDS. Thus, the worst case is that a false positive is not suppressed, but no new false alerts can be generated.

On the other hand, we have to discuss the possibility that ATLANTIDES will introduce additional false negatives (w.r.t. the input NIDS). This happens every time the OAD classifies an alert corresponding to a true attack as a false alert. False negatives are a common problem for alert verification systems (and for ABSs in general). Because of our solution bases its verification on an anomaly-based engine, the threshold used to discern outgoing traffic can be adjusted manually by IT specialists to avoid false negatives (previous proposed solutions cannot be tuned in the same way, e.g. [13]). Although it is theoretically possible that the attacker crafts a particular payload to send a normal response on the current connection after the exploitation, there exist several difficult technical problems which limit the success of this kind of attack. The attacker must inject an attack payload containing the routines to generate the normal output too (or to jump to the original code where this is done): since exploitable buffers are normally small in size, it could be difficult to include the necessary payload.

Missing output response What we just described is the most common behavior; nevertheless we have to take into account that there exist attacks which, e.g., aim to disrupt completely the service or that, exploiting a buffer overflow, radically modify the normal execution. In this case, if the OAD does not detect *any* output related to the pre-alert raised by the NIDS, during the time window t, then the pre-alert is considered an incident and is forwarded to an IT security specialist. Although this could be considered rough, because the missing response could occur for different reasons than a successful attack (e.g., an internal error), this strategy does not introduce any additional false negatives/positives, since with a single NIDS (monitoring the incoming traffic) the alert would be forwarded anyway. Furthermore, Chaboya et al. [5] experimentally verified that most of the buffer overflow attacks against an HTTP server do not produce any output from the attacking requests.

Since nowadays attacks against connection-less protocols are less common (see the Common Vulnerabilities and Exposures [22] (CVE) database for detailed statistics), we have designed ATLANTIDES with the explicit goal of reducing false positives when monitoring network services based on the TCP protocol (e.g., HTTP, SMTP and FTP) where a response is typically sent by the server to the client

Although we do not aim to handle all kinds of possible attacks (e.g., worms or DDoS attacks perpetrated generating a high quantity of legal connections), we believe our solution can improve the accuracy of a NIDS without any additional component installed directly on the monitored hosts (an additional component could affect under certain circumstances host performance, i.e., a high amount of connections).

3.1 The OAD

The OAD is basically an anomaly payload-based NIDS, monitoring the output of a network service rather than the input of it. In our embodiment we use the architecture of POSEIDON as the OAD, because we are familiar with it and it gives better results than its leading competitor [4]. POSEIDON is a 2-tier anomaly-based network intrusion detection system that combines a neural network with n-gram analysis to detect anomalies. POSEIDON performs a packet-based analysis: every packet is classified by the neural network; then, using the classification information given by the neural network, the real detection phase takes place based on statistical functions considering the byte-frequency distributions (n-gram analysis).

The fact that the OAD is anomaly-based (rather than signature-based) has several advantages: the OAD can adapt to the specific network environment/service, and it does not require the definition of new signatures to detect anomalous output, working in an unsupervised way (after the initial setup). Creating and maintaining a set of signatures for outgoing traffic is a thorny and labor-intensive task, as these signatures heavily depend on the local application, and must be updated each time that modifications of the application change its output content. On the other hand, our OAD can simply include these modification in its model, without re-doing the training. The disadvantage of being anomaly-based is that our OAD needs an extensive (though unsupervised) *training* phase: a significant amount of (normal) traffic data is needed to build an accurate model of the service we monitor.

Setting the threshold As we mentioned in Section 2, the completeness and accuracy of anomaly detection systems are intrinsically related to and heavily influenced by the *threshold value*. Here, we call *completeness* the ratio $TP/(TP + FN)$ and *accuracy* the ratio $TP/(TP + FP)$, where TP is the number of true positives, FN is the number of false negatives and FP is the number of false positives raised during the benchmarks. Consequentely, we have to set a threshold value also for our OAD.

Our experiments show that we can empirically set the threshold to a reasonable value without having to try various values by following a simple heuristic referring to functions from the probability theory. The Chebyshev's inequality theorem characterizes the dispersion of data away from its mean and puts an upper bound on the probability that the difference between the value of a random variable x and its mean σ exceeds a certain threshold t, for an arbitrary distribution with mean σ and variance σ^2

$p(|x - \mu| > t) < \frac{\sigma^2}{t^2}$.

Therefore, by computing first the mean and the standard deviation of the distance value observed between the training samples and the model we can then set the threshold at $k\sigma$. Our experiments show that setting the threshold at 2σ, usually yields reasonably good results. Thus, ATLANTIDES set this parameter following this procedure and IT personnel can later adjust it upon request.

4 Experiments and results

To validate our architecture, we benchmark ATLANTIDES in combination with the signature-based NIDS Snort [20] as well as ATLANTIDES in combination with the anomaly-based NIDS POSEIDON [4]. To carry out the experiments, we employ two different data sets. First, we benchmark the system using a private data set. Secondly, we use the DARPA 1999 data set [14]: despite criticism [15, 17] this is a standard data for benchmarking NIDSs (see, e.g., [19, 25]) and it has the advantage that it allows one to compare experiments. No other data set, containing sufficient data to perform verifiable benchmarks, is publicly available.

We consider an attack to be successfully detected when at least one packet carrying the attack payload is correctly flagged as malicious; all the other non-detected packets carrying the attack payload are not considered to be false negatives. On the other hand, each packet incorrectly flagged as malicious is considered to be a false positive. Thus, the detection rate is attacked-based, while the false positive rate is packet-based.

Tests with a private data set To carry out our validation the first set of experiments, we use a private data set we collected at the University of Twente: we call this *data set A*. Data were collected on a public network for 5 consecutive working days (24 hours per day), logging only TCP traffic directed to (and originating from) a heavy-loaded web server (about 10 Gigabytes of total traffic per day). This web server hosts the department official web sites as well as student and research staff personal web pages: thus, the traffic contains different types of data such as static and dynamically generated HTML pages and, especially in the outgoing traffic, common format documents (e.g., PDF) as well as raw binary data (e.g., software executables). We did not inject any artificial attack. We focus on HTTP traffic because nowadays Internet attacks are mainly directed to web servers and web-based applications [12].

To train the anomaly-detection engines of both POSEIDON and the OAD on data set A, we used a snapshot of the data collected during *working hours* (approximately 3 hours, 1.8 Gigabytes of data, randomly chosen). The chosen training data set has not been pre-processed and made attack-free: it is thus possible that the model includes some malicious activity (that could negatively affect accuracy). Similarly, for the test we randomly chose another snapshot (approximately 1.8 Gigabytes of data) to benchmark POSEIDON stand-alone against POSEIDON in combination with ATLANTIDES: we set the threshold of POSEIDON experimetally to achieve the best detection rate at the lowest false positive rate possible.

To distinguish true positives from false positives, the authors have classified alerts manually: we found evidences of XSS and SQL Injection attacks, plus some probes checking for well-known paths (33 attack detections in total). Table 1 summarizes the results we obtained. Figure 2 shows detailed results of ATLANTIDES on data set A. Here, left is better than right and above is better than below. A point left-top indicates a configuration in which (almost) every attack

Protocol		POSEIDON	POSEIDON + ATLANTIDES
HTTP	DR	100%	100%
	FP	1683 (2,83%)	774 (1,30%)

Table 1: Comparison between POSEIDON stand-alone and POSEIDON in combination with ATLANTIDES using data set A; DR stands for detection rate (attack instance percentage), while FP is the false positive rate (packets and corresponding percentage); ATLANTIDES reduces false positives by more than 50% without affecting the detection rate (i.e., without introducing false negatives).

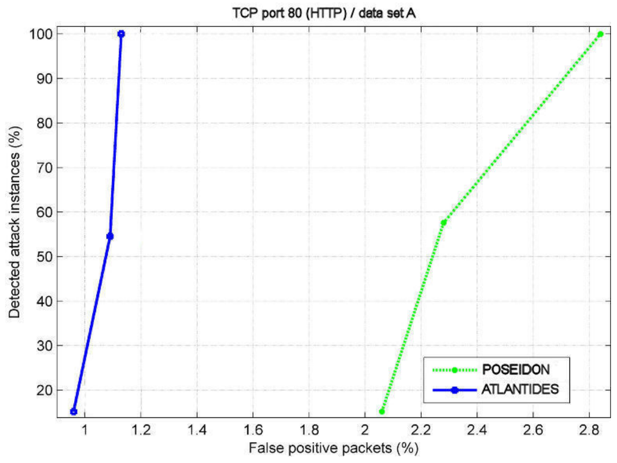

Figure 2: Detection rates for POSEIDON in combination with ATLANTIDES using data set A (HTTP protocol): the x-axis and y-axis present false positive rate (packets) and detection rate (attacks instances) respectively. It is possible to observe that ATLANTIDES presents a lower false positive rate than POSEIDON, considering the same detection rate. It is possible to notice how different ATLANTIDES' threshold settings affect detection and false positive rates.

has been correctly forwarded, with very few false positives left. On the other hand, a point on the low-right side indicates a configuration in which some real attacks have been incorrectly suppressed and a good deal of licit traffic was marked anomalous.

We cannot compare ATLANTIDES in combination with Snort on data set A for the reason that Snort does not find any true attack to the system (Snort raised only false alerts): this is not surprising, since Snort has only few signatures devoted to SQL Injections and XSS attacks. By setting a high threshold value in ATLANTIDES we could remove *all* the false positives, but this would give no indication of the completeness and accuracy of ATLANTIDES.

Tests with the DARPA 1999 data set The testing environment of the DARPA 1999 data set contains several internal hosts that are attacked by both external and internal attackers: in our tests, we consider only inbound and outbound TCP packets that belong to attack connections against hosts inside the network 172.16.0.0/16. We focus on FTP, Telnet, SMTP and HTTP protocols. This is due to the fact that only these protocols, among the ones contained in this data set, provide us with a sufficient number of samples to train the OAD and, at the same time, allow us to compare our architecture with POSEIDON stand-alone, that has been benchmarked following the same procedures.

Protocol		Snort	Snort + ATLANTIDES	POSEIDON	POSEIDON + ATLANTIDES
HTTP	DR	59,9%	59,9%	100%	100%
	FP	599 (0,069%)	5 (0,00057%)	15 (0,0018%)	0 (0,0%)
FTP	DR	31,75%	31,75%	100%	100%
	FP	875 (3,17%)	317 (1,14%)	3303 (11,31%)	373 (1,35%)
Telnet	DR	26,83%	26,83%	95,12%	95,12%
	FP	391 (0,041%)	6 (0,00063%)	63776 (6,72%)	56885 (5,99%)
SMTP	DR	13,3%	-	100%	100%
	FP	0 (0,0%)	-	6476 (3,69%)	2797 (1,59%)

Table 2: Comparison between Snort stand-alone, Snort in combination with ATLANTIDES, POSEIDON stand-alone and POSEIDON in combination with ATLANTIDES using the DARPA 1999 data set: DR stands for detection rate (attack instance percentage), while FP is the false positive rate (packets and corresponding percentage); ATLANTIDES reduces false positives by more than 50% most of the times, being close to zero in 3 tests, without affecting the detection rate (i.e., without introducing false negatives).

We train the OAD of ATLANTIDES with the data of weeks 1 and 3 (attack-free): for each different protocol we use a different OAD instance. Afterwards, we test ATLANTIDES together with both POSEIDON and Snort using week 4 and week 5 traffic. In order to distinguish between true and false positives, we refer to the attack instance table provided by the DARPA data set authors. Table 2 reports a comparison of the detection and false positive rates of Snort stand-alone (first column), Snort in combination with ATLANTIDES (second column), POSEIDON stand-alone (third column) and POSEIDON in combination with ATLANTIDES (fourth column).

In both cases, ATLANTIDES achieves a substantial improvement on the stand-alone system, without affecting the detection rate nor introducing false negatives; ATLANTIDES reduces the false positive amount by at least 50% on every protocol benchmarked, except for the Telnet protocol together with POSEIDON. In our opinion, this discrepancy is due to the fact that Telnet has a great output variability, since an user could issue hundreds of different commands with different output; on the other hand, protocols like HTTP, FTP and SMTP present well-defined protocol schemas to exchange information between client and server. We did not test ATLANTIDES on SMTP traffic in combination with Snort because in this case Snort raises no false positives.

4.1 Supporting NIDS configuration and management

As we mentioned in Section 2, the most important and time-consuming tasks of NIDSs' management are their tuning and the verification of alerts raised. The former could be a thorny task, especially in the case of signature-based systems, since personnel with high technical skills must be involved: performing a poor tuning phase (or even omitting it) usually leads to a high rate of false positives. Next to the reduction in workload due to the reduction of false positives, ATLANTIDES reduces the workload of the IT specialists by allowing them to carry out a less precise tuning phase. In fact, ATLANTIDES is effective in reducing the false positive rate even when a tuning phase of the input NIDS has not taken place. To support this, in our benchmarks

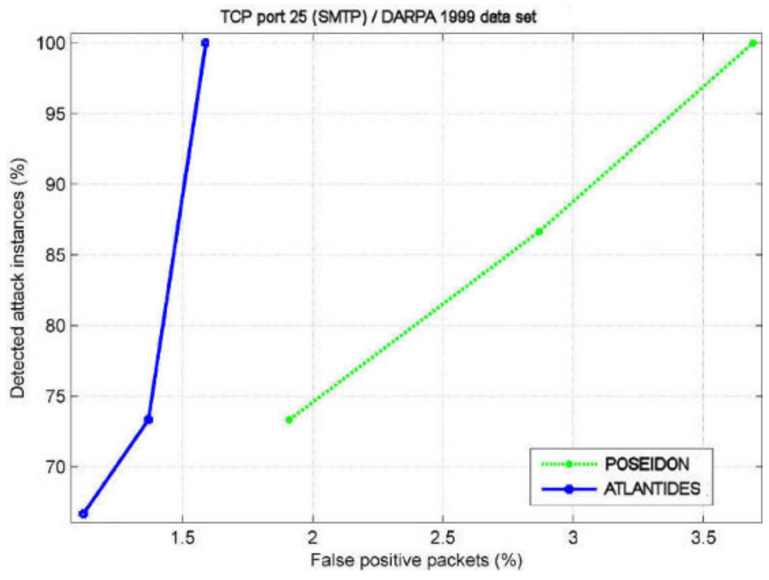

Figure 3: Detection rates for POSEIDON in combination with ATLANTIDES using DARPA 1999 data set (SMTP protocol): the x-axis and y-axis present false positive rate (packets) and detection rate (attacks instances) respectively. Is it possible to observe that ATLANTIDES presents a lower false positive rate than POSEIDON, considering the same detection rate. It is possible to notice how different ATLANTIDES' threshold settings affect detection and false positive rates.

with Snort we did not tune it wrt our testing environment: the latest available set of signatures has been loaded into the signature database without any signature selection process.

Also (and perhaps even more so) in the case of anomaly-based systems, ATLANDITES eases the task of the IT specialist: as we mentioned before, the main difficulties one faces when deploying an anomaly-based IDS are (a) coping with the usually high number of false positives and (b) finding the optimal value of the threshold, i.e. the value reaching the best compromise between false positives and false negatives. ATLANTIDES influences directly (a) by reducing the number of false positives and indirectly (b) by allowing the system to produce good results even when the threshold is not set optimally.

5 Related work

Although no specific solution has been previously developed to ease NIDS management, some previous work regarding alert verification is also relevant to this topic.

Kruegel and Robertson [13] introduces a plug-in for Snort to verify alerts: the plug-in integrates the Nessus vulnerability scanner into the Snort's core. When an alert is raised, this is not immediately forwarded but is firstly passed to the verification engine. Since every Snort's signature comes with a unique identifier (assigned by CVE [22]), this index is used to check the presence of a corresponding Nessus attack script. If found, the script is executed against the target machine/network: the output is extracted and used to flag the alert as either true or false; an output cache is used to avoid further verification for the same alert/target. Although this approach is effective, there are several drawbacks: one has to maintain the Nessus's attack script database updated, and this approach works only for signature-based NIDSs, while AT-

LANTIDES can work with both types and in a complete automatic way (i.e., no manual updates needed). Besides, Nessus could produce false positives/negatives too.

Pietraszek [19] tackles the problem by introducing an alert classifier system (**ALAC**, **A**daptive **L**earner for **A**lert **C**lassification) based on machine learning techniques. During the training phase, the system classifies alerts into true and false positives, by attaching a label from a fixed set of user-defined labels to the current alert. Then, the system computes an extra parameter (called *classification confidence*) and presents this classification to a human analyst. The analyst's feedback is used to generate training examples, used by the learning algorithm to build and update its classifiers. After the training phase, the classifiers are used to classify new alerts. This approach relies on the analyst's ability to classify alerts properly and on his availability to operate in real-time (otherwise the system will not be updated in time); we believe that these (demanding) requirements can be considered acceptable for a signature-based NIDS (where the analyst can easily inspect both the signature and network packet(s) that triggered the alert), but it could be difficult to perform the same analysis with an anomaly-based NIDS. The reported benchmarks conducted over the 1999 DARPA data set, using Snort to generate alerts, show an overall false positive reduction of over 30% (details on single attack protocols are not given). The the main differences between ALAC and ATLANTIDES include: (a) ALAC does not consider the outgoing traffic, and (b) ALAC relies heavily on the expertise and the presence of an analyst (in ATLANTIDES, all the IT specialist has to do is to set the thresholds).

As we mentioned before, e.g. misconfigured devices can generate recurrent (and benign) alerts. Julisch [10] presents a semi-automatic approach, based on techniques which discover frequently occurring episodes in a given sequence, for identifying false positives based on the idea of *root cause*: an alert root cause is defined as "the reason for which it occurs". The author observes that in most environments, it is possible to identify a small number of highly predominant (and persistent) root causes: thereby removing such root causes drastically reduces the future alert rate. Benchmarks conducted on a log trace from a commercial signature-based NIDS deployed in a real network show a reduction of 87% of false positives. No further details are given about the testing condition, network topology or traffic typology. We cannot compare directly this approach with ATLANTIDES because the data used by the author is private, nevertheless we can notice that this approach is applicable only to signature-based NIDes, while ATLANTIDES is effective with anomaly-based systems too.

6 Conclusion

In this paper we present ATLANTIDES, an architecture for automatic alert verification exploiting in a structural way the detection of anomalies in the output traffic of a system. ATLANTIDES can be used to facilitate the deployment and management of NIDSs by reducing false positives both in signature and anomaly-based systems. The core of ATLANTIDES consists of an output anomaly detector (OAD), which compares output traffic with a model it has created during the training phase. To reduce *false positives* on the input NIDS (be it signature or anomaly-based) monitoring the incoming traffic, ATLANTIDES checks if the communication raising an alert in the input NIDS actually produces an anomaly in the outgoing traffic too. In this case (and in another exceptional situation), the alert is forwarded to the IT specialist, otherwise it is discarded. The fact that the OAD is anomaly-based (rather than signature-based) allows it to adapt to the specific network environment/service, and to work in an unsupervised way (at least, after the setup). Anomaly-based systems typically use a distance function and a threshold to discern anomalous from licit traffic. We introduce a simple heuristic to set AT-

LANTIDES threshold in an automatic, though effective, way, to further ease the management for IT security specialists (which can in case adjust the threshold value).

Benchmarks on a private data set and on the DARPA 1999 data set show that ATLANTIDES determines a reduction of false positives between 50% and 100% in most of the cases, without introducing any extra false negative, easing signifincantly the management of NIDSs.

One possible extension to our architecture is adding additional information to make the detection of anomalies in the output more precise: this information (e.g., the usual amount of bytes sent back from the server and the communication duration) could be included in the model and evaluated as well. Our architecture has been designed to work with TCP-based network services: although it could be easily adapted to work with UDP-based services, there exist some issues related to this protocol. In fact UDP is a connection-less protocol and this add some difficulties to distinguish real connections from the ones using spoofed IP addresses. We will investigate this in future.

References

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

[2] S. Axelsson. “Intrusion Detection Systems: A Survey and Taxonomy”. Technical Report 99-15, Chalmers University, mar 2000.

[3] S. Axelsson. “The Base-rate Fallacy and the Difficulty of Intrusion Detection”. *ACM Trans. Inf. Syst. Secur. (TISSEC)*, 3(3):186–205, 2000.

[4] D. Bolzoni, E. Zambon, S. Etalle, and P. Hartel. “POSEIDON: a 2-tier Anomaly-based Network Intrusion Detection System”. In *Proceedings of the 4th IEEE International Workshop on Information Assurance (IWIA)*, pages 144–156. IEEE Computer Society Press, 2006.

[5] D. J. Chaboya, R. A. Raines, R. O. Baldwin, and B. E. Mullins. “Network Intrusion Detection: Automated and Manual Methods Prone to Attack and Evasion”. *IEEE Security and Privacy*, 4(6):36–43, 2006.

[6] O. Dain and R. Cunningham. “Fusing Heterogeneous Alert Streams into Scenarios”. In *Proceedings of the Workshop on Data Mining for Security Applications, 8th ACM Conference on Computer Security (CCS)*, pages 1–13. ACM Press, 2002.

[7] H. Debar, M. Dacier, and A. Wespi. “A Revised Taxonomy of Intrusion-Detection Systems”. *Annales des Télécommunications*, 55(7–8):361–378, 2000.

[8] S. Forrest and S. A. Hofmeyr. “A Sense of Self for Unix Processes”. In *Proceedings of the 17th IEEE Symposium on Security and Privacy (S&P)*, pages 120–128. IEEE Computer Society Press, 2002.

[9] G. Gu, M. Sharif, X. Qin, D. Dagon, W. Lee, and G. Riley. “Worm Detection, Early Warning and Response Based on Local Victim Information”. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC)*, pages 136–145. IEEE Computer Society, 2004.

[10] K. Julisch. “Mining Alarm Clusters to Improve Alarm Handling Efficiency”. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)*, pages 12–21. ACM Press, 2001.

[11] K. Julisch. “Clustering Intrusion Detection Alarms to Support Root Cause Analysis”. *ACM Transactions on Information and System Security (TISSEC)*, 6(4):443–471, 2003.

[12] D. V. Klein. “Defending Against the Wily Surfer-Web-based Attacks and Defenses”. In *Proc. of the Workshop on Intrusion Detection and Network Monitoring*, pages 81–92. USENIX Association, 1999.

[13] C. Kruegel and W. Robertson. “Alert Verification: Determining the Success of Intrusion Attempts”. In *Proceedings of the 1st Workshop on the Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, 2004.

[14] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. “The 1999 DARPA Off-line Intrusion Detection Evaluation”. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 34(4):579–595, 2000.

[15] M. V. Mahoney and P. K. Chan. “an Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection”. In *Proceedings of the 6th Symposium on Recent Advances in Intrusion Detection (RAID)*, volume 2820 of *LNCS*, pages 220–237. Springer-Verlag, 2003.

[16] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz. “A Data Mining Analysis of RTID Alarms”. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 34(4):571–577, 2000.

[17] J. McHugh. “Testing Intrusion Detection Systems: a Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as performed by Lincoln Laboratory”. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):262–294, 2000.

[18] V. Paxson. “Bro: a System for Detecting Network Intruders in Real-time”. *Computer Networks*, 31(23–24):2435–2463, 1999.

[19] T. Pietraszek. “Using Adaptive Alert Classification to Reduce False Positives in Intrusion Detection”. In *Proceedings of the 7th Symposium on Recent Advances in Intrusion Detection (RAID)*, volume 3224 of *LNCS*, pages 102–124. Springer-Verlag, 2004.

[20] M. Roesch. “Snort - Lightweight Intrusion Detection for Networks”. In *Proceedings of the 13th USENIX Conference on System Administration (LISA)*, pages 229–238. USENIX Association, 1999.

[21] Tenable Network Security. Nessus Vulnerabilty Scanner, 2002. URL http://www.nessus.org/.

[22] The MITRE Corporation. Common Vulnerabilities and Exposures database, 2004. URL http://cve.mitre.org.

[23] H. L. Van Trees. *“Detection, Estimation and Modulation Theory. Part I: Detection, Estimation, and Linear Modulation Theory”*. John Wiley and Sons, Inc., 1968.

[24] K. Wang, G. Cretu, and S. J. Stolfo. “Anomalous Payload-based Worm Detection and Signature Generation”. In *Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection (RAID)*, volume 3858 of *LNCS*, pages 227–246. Springer-Verlag, 2005.

[25] K. Wang and S. J. Stolfo. “Anomalous Payload-based Network Intrusion Detection”. In *Proceedings of the 7th Symposium on Recent Advances in Intrusion Detection (RAID)*, volume 3224 of *LNCS*, pages 203–222. Springer-Verlag, 2004.

[26] J. Zhou, A. J. Carlson, and N. Bishop. “Verify Results of Network Intrusion Alerts Using Lightweight Protocol Analysis”. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC)*, pages 117–126. IEEE Computer Society, 2005.