ATLANTIDES: An Architecture for Alert Verification in Network Intrusion Detection Systems

Damiano Bolzoni – University of Twente, The Netherlands
Bruno Crispo – Vrije Universiteit, The Netherlands & University of Trento, Italy
Sandro Etalle – University of Twente, The Netherlands

ABSTRACT

We present an architecture[1] 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%.

Introduction

Network intrusion-detection systems (NIDSs) are considered an effective second line of defense against network-based attacks directed to computer systems [4, 11], and – due to the increasing severity and likelihood of such attacks – are employed in almost all large-scale IT infrastructures [1].

The Achille's heel of NIDSs lies in the large number of *false positives* (i.e., notifications of attacks that turn out to be false) that occur [26]: practitioners [24, 31] as well as researchers [3, 8, 15] observe that it is common for a NIDS to raise thousands of alerts per day, most of which are false alerts. Julisch [16] states that up to 99% of total alerts may not be 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 – according to Axelsson [3] – the limiting factor for the performance of an intrusion-detection system. False alerts also cause an overload for IT personnel [24], who must verify every single alert, a task that is not only labor intensive but also error prone [9]. Indeed, a high false positive rate can even be *exploited* by attackers to overload IT personnel, thereby lowering the defenses of the IT infrastructure.

The main reason why NIDSs raise false positives is that – quoting Kruegel and Robertson [18] – they are often run without any (or very limited) information about the network resources that they protect (i.e., the context). Chaboya, et al. [6] state that the context knowledge (e.g., network and system configurations) can improve significantly alert verification. On the other hand, building and updating a database of the configurations or running vulnerability assessment tools (e.g., Nessus [35]) 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 are tailored for specific attacks [14, 41] (e.g., worm-like) or support only signature-based NIDSs [33, 36] (e.g., Snort's team has developed a specific plug-in, *flowbits*, to cope with this, but it has limited functionality).

Our thesis is 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* (Architecture for Alert verification in Network Intrusion Detection Systems) an innovative architecture for easing the 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 [44], thus modifying the normal output outcome. Detecting this anomaly can help in reducing false alerts. For instance, a successful SQL Injection attack [43] against a web application often causes the output of SQL table content (e.g., user/admin credentials) rather than the expected web content.

ATLANTIDES, which is completely network-based,[2] works by analyzing (using n-Gram analysis

[1] This research is supported by the research program Sentinels (http://www.sentinels.nl). The work of the second author was partially funded by the IST FP6 GridTrust project, contract No. 033827. 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.

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

[13]) 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 cannot work properly with encrypted data unless the cryptographic keys are provided.

In the past, simple correlations between input and output traffic have already been used to identify possible worm attacks [14, 41]. To the best of our knowledge, ATLANTIDES is the first proposed solution for alert verification that:

- 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 management

We benchmarked ATLANTIDES in combination with the signature-based NIDS Snort [34, 37], as well as in combination with the anomaly-based NIDS POSEIDON [5]. We carried out benchmarks both on a private data set as well as on the common DARPA 1999 data set [22] (for the sake of completeness and to allow duplication of our results, despite criticism [23, 25]). In seven out of eight cases, our benchmarks show a reduction of false positives between 50% and 100%.

Preliminaries

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 [34, 37], are based on pattern-matching techniques: the NIDS contains a known-attack *signature* database 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 [10, 11]). 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.

False Positives in Signature-Based Systems

SBSs raise an alert every time that traffic matches one of the signatures loaded into the system. Consider for example the *path traversal attack*, which allows to access files, directories, and commands residing outside the (given) web document root directory. The most elementary path traversal attack uses the "../" character sequence to alter the resource location requested in the URL. Variations include valid and invalid Unicode-encoding ("..%u2216" or "..%c0%af"), URL encoded characters ("%2e%2e%2f"), and double URL encoding ("..%255c") of the backslash character (excerpted from the *WASC Threat Classification* [43]).

To detect these attacks many SBSs (using an out-of-the-box configuration) raise an alert each time they identify the pattern "../" in the incoming traffic. Unfortunately, this pattern could be present in legal traffic too; some Content Management Systems (CMSs) insert relative paths in parameters to load files, which causes SBSs to raise a high number of false alerts. These false alerts can be avoided by deactivating the specific rule. On the other hand, this prevents the NIDS from detecting this sort of attacks.

Tuning Signature-Based Systems

The main reasons why alerts produced by SBSs turn out to be either false or irrelevant include the following:

- Writing signatures for NIDS is a thorny task [32], 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) and an overly general one (which will classify legitimate traffic as an attack attempt).
- The monitored environment is not susceptible to a certain vulnerability.
- Misconfigured network devices or services producing atypical output (usually, in this case, it is possible to observe recurrent and periodic phenomena).

A good deal of the false positives raised by a SBS can be suppressed by a *tuning* activity: this activity, based on deactivation of unneeded signatures, requires a thorough analysis of the environment by qualified IT personnel. Finally, to remain effective, SBSs require configuration updating 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

Anomaly-based systems (ABSs) use statistical methods to monitor network traffic. Intuitively, an ABS works by training itself to recognize acceptable behavior and then raising an alert for any behavior 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 user: when the distance between the input and the model exceeds the threshold, an alarm is raised.

The ABSs' main advantage is that they can detect zero-day attacks: novel attacks can be detected as soon as they take place. Clearly, because of their statistical nature, ABSs are bound to raise a number of false positives, and the value of the threshold actually determines a compromise between the number of false positives and the number of false negatives the IT security personnel is willing to accept.

False Positives in Anomaly-Based Systems

The high false positive rate is generally cited as one of the main disadvantages of anomaly-based systems. The value of the threshold has a direct influence on both false negative and false positive rates [40]: 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.

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).

The first internal component is the output anomaly detector (OAD), which is actually an anomaly-based NIDS 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-inspection [7]) and 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. The CE uses a hash-table to store this information, using less than 20 bytes per each entry: thus, the CE does not requires much memory to store the information, and ATLANTIDES can handle even a rate of 1000 alerts per second with a total memory space of 1 MB (in case the connections are kept in memory, e.g., for a maximum time of 60 seconds before being dropped). At this time, the alert is not considered an incident yet (it is a *pre-alert*) and is not forwarded immediately to IT specialists.

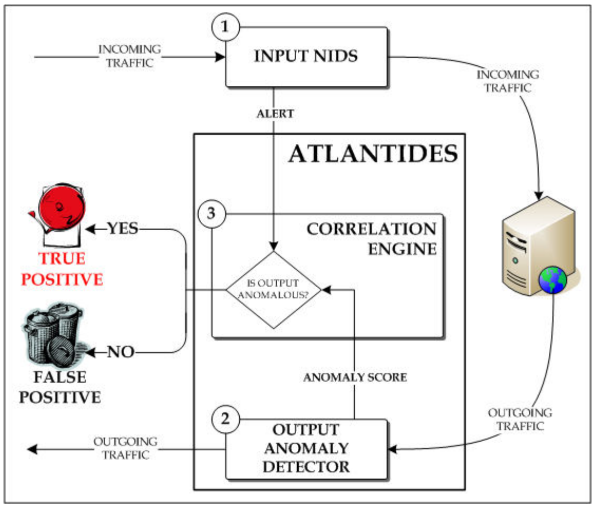

Figure 1: ATLANTIDES's architecture.

Next, the CE marks 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) the 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 fixed this value to 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).

Although a delay is introduced to allow the OAD to process the data sent back to the client, this 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*. In Appendix A we provide the pseudo-code of our architecture.

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 generates false positives or false negatives. The former situation cannot take place because the output of the OAD is evaluated *only* when an alert has already been raised by the input NIDS: the OAD could mistake the alert-related outgoing traffic as anomalous and then forward the alert as a true positive, but this would have happened in any case, if considering the output of the input NIDS only. Thus, the worst case is that a false positive is not suppressed, but any new false alert cannot 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., [18]). an effective threshold automatically.

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. [6] experimentally verified that most of the buffer overflow attacks against an HTTP server do not produce any output from the attacking requests. 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.

Since nowadays attacks against connection-less protocols are less common (see the Common Vulnerabilities and Exposures [39] (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).

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 choose to use the NIDS POSEIDON as the OAD, because we are familiar with it and it gives better results than its leading competitor [5]. POSEIDON is a 2-tier payload-based ABS 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 various 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 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 local applications, and must be updated each time that modifications of the application change its output content. On the other hand, the OAD can simply include these modification in its model, without starting training over. 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 Anomaly-Based Systems, in ABSs completeness and accuracy are intrinsically related 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. Our experiments show that setting the threshold at $3\ \frac{t\_max}{4}$, usually yields reasonably good results, where *t_max* is the maximum distance between the analyzed data and the model observed during the training phase. Thus, we can automatically set this parameter and IT personnel can later adjust it as necessary.

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).

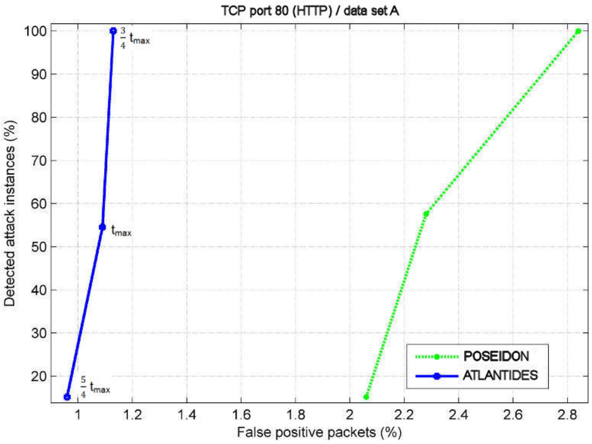

Figure 2: Detection rates for POSEIDON in combination with ATLANTIDES using data set A (HTTP protocol): the x-axis and y-axis present false positive rate (packets) and detection rate (attacks instances) respectively. It is possible to observe that ATLANTIDES presents a lower false positive rate than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

Experiments and Results

To validate our architecture, we benchmark ATLANTIDES in combination with the signature-based NIDS Snort [34, 37] as well as ATLANTIDES in combination with the anomaly-based NIDS POSEIDON [5]. 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 [22]: despite criticism [23, 25] this is a standard data for benchmarking NIDSs (see, e.g., [33, 42]) 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, and to see how the system behaves when trained with a data set that was not made attack-free,[3] we consider a private data set we collected at the University of Twente: this is *data set A*. Data were collected on a public network for five 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 [17]: Kruegel, et al. [19] state that web-based attacks account for 20%-30% from 1999 to 2004 in CVE entries [39]; Symantec Corporation [38] reports that, in the first-half of year 2006, 69% of total discovered vulnerabilities were related to web services and, during the same period, more than 60% of *easily exploitable vulnerabilities* (whenever the exploitation code is not needed or well-known) affected web applications. Symantec states that typical examples of easily exploitable vulnerabilities are SQL Injection and Cross-Site Scripting (XSS) attacks.

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 three hours, 1.8 Gigabytes of data, randomly chosen). The chosen training data set had not been pre-processed and made attack-free: thus it is possible that the model includes some malicious activity (that could negatively affect accuracy). For the same reason, we randomly chose another snapshot (approximately 1.8 Gigabytes of data) to benchmark POSEIDON stand-alone against POSEIDON in combination with ATLANTIDES.

[3] This is useful to see how the system performs in the sub-optimal situation in which the IT security specialist does not have the time to clean up the training data set, a situation that is likely to occur often in practice.

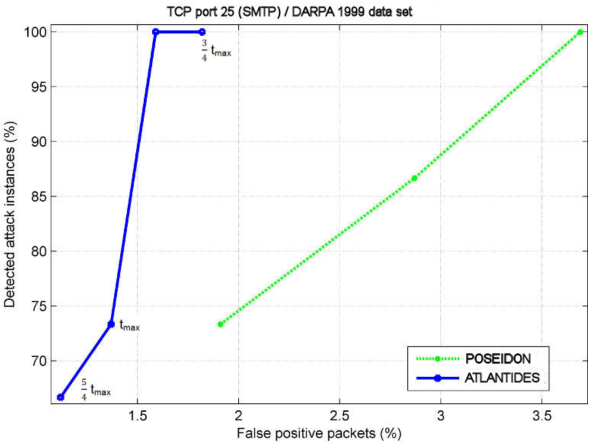

Figure 3: Detection rates for POSEIDON in combination with ATLANTIDES using DARPA 1999 data set (SMTP protocol): the x-axis and y-axis present false positive rate (packets) and detection rate (attacks instances) respectively. Is it possible to observe that ATLANTIDES presents a lower false positive rate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

ABSs can, obviously, achieve a 100% detection rate using a very low threshold value, but this negatively affects the false positive rate too (as we mentioned in Section Anomaly-Based Systems): we set the threshold of POSEIDON experimetally to achieve the best detection rate at the lowest false positive rate possible.

The alerts have been classified by the authors: we found evidences of XSS and SQL Injection attacks [43] (and this is not surprising, accordingly to Symantec's report), plus some probes checking for well-known paths (33 attack detections in total). Table 1 summarizes the results we obtained. 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. 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 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.

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.

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).

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).

In both cases, ATLANTIDES achieves a substantial improvement on the stand-alone system, neither 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. ATLANTIDES is not applied to SMTP traffic in combination with Snort because in this case Snort raises no false positives.

Related Work

The problem of alert verification has been addressed using two different kinds of approaches: we have techniques for *identifying true positives*, and techniques for *identifying false positives*. The main difference between our work and the papers described below is that we take into account the outgoing traffic of the system.

Identifying True Positives

Kruegel and Robertson [18] 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 fired, 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 [39]), 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 ATLANTIDES can work with both types and in a complete automatic way (i.e., no manual updates needed).

Ning, et al. developed a model [30] and an intrusion-alert correlator [27] to help human analysts during the alert verification phase. This work is based on the observation that most attacks consist of several related stages, with the early stages preparing for the later ones. Hyper-alert correlation graphs are used to represent correlated alerts in an intuitive way. However, this correlation technique is ineffective when attackers use a different (yet not spoofed) IP source address at each attack step. Ning and Cui [27] demonstrate the effectiveness of this approach when applied on a small data set (due to the exponential complexity of hyper-alert graphs): in [28, 29] the same authors present other utilities they developed to facilitate the analysis of large sets of correlated alerts, and report some benchmarks employing network traffic used during the DEFCON 8 Capture the Flag (CTF) event [12]. ATLANTIDES does not present the same limitations on data set size.

Lee and Stolfo [20] develop a hybrid network and host-based framework based on data mining techniques, such as sequential patterns mining and episodes rules, to address the problem of improving attack detection while maintaining a low false positive rate. The system detects attacks combining different models and comparing them with actual traffic features. Benchmarks have been conducted using the DARPA 1998 data set [21]: detection score for different attack typologies has a minimum value of 65% with a false positive rate always below 0.05%. Since the authors use a different data set, we cannot compare directly the two approaches: however, we can notice that our approach does not use information collected from the operating system hosting the monitored network service(s), thus ATLANTIDES can work on-line without affecting the host performance.

Identifying False Positives

Pietraszek [33] tackles the problem of reducing false positives 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. To ensure the stability of the system over time, a sub-sampling technique is applied: regularly, the system randomly selects *n* alerts to be forwarded to the analyst instead of processing them autonomously. 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. 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).

Julisch [15] 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.

Analyzing output traffic The idea of analyzing (and correlating) the output of a (possible) compromised system as been used before in the context of worm detection.

Gu, et al. [14] scan the output traffic for specific port numbers. When an anomaly has been detected in the incoming traffic directed to a certain destination service port, their system start monitoring the output traffic to check whether the host tries to contact other systems using the same destination service port: if this is the case then the system is probably infected by a worm. Wang, et al. [41] proceed in a similar way, comparing outgoing to incoming traffic, looking for similarities: when an anomaly has been detected in the incoming traffic, the anomalous traffic is cached and compared to subsequent outgoing traffic (to detect polymorphic worms). A successful match indicates that the host has been infected and that the worm is trying to replicate itself, infecting other hosts. Any other kind of attack will not be handled by the system. In contrast, our solution presents a general architecture to carry out a complete anomaly detection on the output to reduce false positives of any NIDS placed on the input channel. Indeed we have shown that our architecture works well in combination with both a signature and an anomaly-based input NIDS.

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 for reducing false positives both in signature and anomaly-based NIDSs. 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 ATLANTIDES 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.

Author Information

Damiano Bolzoni is currently a Ph.D. student at the University of Twente, Netherlands. His research interests are focused on intrusion detection systems and information risk management. He received a MSc in Computer Science from the University of Venice, Italy, with a thesis about anomaly-based network intrusion detection systems. He can be reached at damiano.bolzoni@utwente.nl .

Bruno Crispo is a faculty member at the University of Trento and at the Vrije Universiteit Amsterdam. His research interests are security protocols, authentication, authorization and accountability in large distributed systems, RFID and sensors security. He has a Ph.D. in Computer Science from the University of Cambridge, UK. Contact him at crispo@dit.unitn.it .

Sandro Etalle received a Ph.D. from the University of Amsterdam, and has worked for the universities of Genova, Amsterdam, Maastricht, Trento. At the moment he is associate professor in the Distributed and Embedded Systems Group at the University of Twente, the Netherlands. His research covers trust management, intrusion detection systems and information risk management. He can be reached at sandro.etalle@utwente.nl .

Bibliography

[1] Allen, J., 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] Axelsson, S., "Intrusion Detection Systems: A Survey and Taxonomy," *Technical Report 99-15*, Chalmers University, Mar, 2000.

[3] Axelsson, S., "The Base-Rate Fallacy and the Difficulty of Intrusion Detection," *ACM Transactions on Information and System Security (TISSEC)*, Vol. 3, Num. 3, pp. 186-205, 2000.

[4] Bace, R., *Intrusion detection*, Macmillan Publishing Co., Inc., 2000.

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

[6] Chaboya, D. J., 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*, Vol. 4, Num. 6, pp. 36-43, 2006.

[7] Check Point Software Technologies, *Stateful Inspection Technology*, 2005, http://www.checkpoint.com/products/down-loads/Stateful_Inspection.pdf .

[8] Clifton, C. and G. Gengo, "Developing Custom Intrusion Detection Filters Using Data Mining," *Proceedings of the 21st Century Military Communications Conference (MILCOM)*, Vol 1, pp. 440-443, IEEE Computer Society Press, 2000.

[9] Dain, O., and R. Cunningham, "Fusing Heterogeneous Alert Streams into Scenarios," *Proceedings of the Workshop on Data Mining for Security Applications, 8th ACM Conference on Computer Security (CCS)*, pp. 1-13, ACM Press, 2002.

[10] Debar, H., M. Dacier, and A. Wespi, "Towards a Taxonomy of Intrusion-Detection Systems," *Computer Networks*, Vol. 31, Num. 8, pp. 805-822, 1999.

[11] Debar, H., M. Dacier, and A. Wespi, "A Revised Taxonomy of Intrusion-Detection Systems," *Annales des Télécommunications*, Vol. 55, Num. 7-8, pp. 361-378, 2000.

[12] DEFCON8, *Defcon Capture the Flag (CTF) Contest*, 2000, http://www.defcon.org/html/defcon8/defcon-8-post.html .

[13] Forrest, S. and S. A. Hofmeyr, "A Sense of Self for Unix Processes," *Proceedings of the 17th IEEE Symposium on Security and Privacy (S&P)*, pp. 120-128, IEEE Computer Society Press, 2002.

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

[15] Julisch, K., "Mining Alarm Clusters to Improve Alarm Handling Efficiency," *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)*, pp. 12-21, ACM Press, 2001.

[16] Julisch, K., "Clustering Intrusion Detection Alarms to Support Root Cause Analysis," *ACM Transactions on Information and System Security (TISSEC)*, Vol. 6, Num. 4, pp. 443-471, 2003.

[17] Klein. D. V., "Defending Against the Wily Surfer-Web-based Attacks and Defenses," *Proceedings of the Workshop on Intrusion Detection and Network Monitoring*, pp. 81-92, USENIX Association, 1999.

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

[19] Kruegel, C., G. Vigna, and W. Robertson, "A Multi-model Approach to the Detection of Web-based Attacks," *Computer Networks*, Vol. 48, Num. 5, pp. 717-738, 2005.

[20] Lee, W. and S. J. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems," *ACM Transactions on Information and System Security*, Vol. 3, Num. 4, pp. 227-261, 2000.

[21] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation," *Proceedings of the 1st DARPA Information Survivability Conference and Exposition (DISCEX)*, Vol. 2, pp. 12-26. IEEE Computer Society Press, 2000.

[22] Lippmann, R., 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*, Vol. 34, Num. 4, pp. 579-595, 2000.

[23] Mahoney, M. V. 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)*, Vol. 2820 of *LNCS*, pp. 220-237, Springer-Verlag, 2003.

[24] Manganaris, S., M. Christensen, D. Zerkle, and K. Hermiz, "A Data Mining Analysis of RTID Alarms," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol. 34, Num. 4, pp. 571-577, 2000.

[25] McHugh, J., "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)*, Vol. 3, Num. 4, pp. 262-294, 2000.

[26] Morin, B., L. Mé, H. Debar, and M. Ducassé, "M2D2: A Formal Data Model for IDS Alert Correlation," *Proceedings of the 5th Symposium on Recent Advances in Intrusion Detection (RAID)*, Vol. 2516 of *LNCS*, pp. 115-127, Springer-Verlag, 2002.

[27] Ning, P. and Y. Cui, "An Intrusion Alert Correlator Based on Prerequisites of Intrusions," *Technical Report TR-2002-01*, North Carolina State University, 2002.

[28] Ning, P., Y. Cui, and D. Reeves, "Analyzing Intensive Intrusion Alerts via Correlation," *Proceedings of the 5th Symposium on Recent Advances in Intrusion Detection (RAID)*, Vol. 2516 of *LNCS*, pp. 74-94, Springer-Verlag, 2002.

[29] Ning, P. Y., Cui, D. Reeves, and D. Xu, "Techniques and Tools for Analyzing Intrusion Alerts," *ACM Transactions on Information and System Security (TISSEC)*, Voo. 7, Num. 2, pp. 274-318, 2004.

[30] Ning, P., D. Reeves, and Y. Cui, "Correlating Alerts Using Prerequisites of Intrusions," *Technical Report TR-2001-13*, North Carolina State University, 2001.

[31] Ning, P., and D. Xu, "Learning Attack Strategies From Intrusion Alerts," *Proceedings of the 10th ACM conference on Computer and Communications Security (CCS)*, pp. 200-209, ACM Press, 2003.

[32] Paxson, V., "Bro: a System for Detecting Network Intruders in Real-time," *Computer Networks*, Vol. 31, Num. 23-24, pp. 2435-2463, 1999.

[33] Pietraszek, T., "Using Adaptive Alert Classification to Reduce False Positives in Intrusion Detection," *Proceedings of the 7th Symposium on Recent Advances in Intrusion Detection (RAID)*, Vol. 3224 of *LNCS*, pp. 102-124, Springer-Verlag, 2004.

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

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

[36] Sommer, R., and V. Paxson, "Enhancing Byte-level Network Intrusion Detection Signatures With Context," *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, pp. 262-271, ACM Press, 2003.

[37] *Sourcefire*, Snort Network Intrusion Detection System Web Site, 1999, http://www.snort.org .

[38] Symantec Corporation, *Internet Security Threat Report*, 2006, http://www.symantec.com/enterprise/threat-report/index.jsp .

[39] The MITRE Corporation, *Common Vulnerabilities and Exposures Database*, 2004, http://cve.mitre.org .

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

[41] Wang, K., G. Cretu, and S. J. Stolfo, "Anomalous Payload-based Worm Detection and Signature Generation," *Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Vol. 3858 of *LNCS*, pp. 227-246, Springer-Verlag, 2005.

[42] Wang, K. and S. J. Stolfo, "Anomalous Payload-based Network Intrusion Detection," *Proceedings of the 7th Symposium on Recent Advances in Intrusion Detection (RAID)*, Vol. 3224 of *LNCS*, pp. 203-222, Springer-Verlag, 2004.

[43] Web Application Security Consortium, *Web Security Threat Classification*, 2005, http://www.webappsec.org/projects/threat/ .

[44] Zhou, J., A. J. Carlson, and N. Bishop, "Verify Results of Network Intrusion Alerts Using Lightweight Protocol Analysis," *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC)*, pp. 117-126, IEEE Computer Society, 2005.

ATLANTIDES Pseudo-code

In this section we give a semi-formal description of how ATLANTIDES works.

```
DATA TYPE

l = length of the longest packet payload
PAYLOAD = array [1..l] of [0..255]                 /* packet payload */
HOMENET = set of IP addresses                      /* hosts inside the monitored network */

HOST = RECORD [
    address: IP address ∈ ℕ
    port: TCP port ∈ ℕ
]

PACKET = RECORD [
    source: HOST
    destination: HOST
    payload: PAYLOAD
]

alert = RECORD [
    alert:
        −∞ if input NIDS is SBS
        value ∈ Real if input NIDS is ABS
    processed: BOOLEAN                             /* tracks a processed alert by the OAD */
    true_alert: BOOLEAN                            /* alert is marked as an incident */
]

DATA STRUCTURE
τ ∈ ℕ                                              /* number of packets used for OAD training */
oad ∈ NIDS                                         /* ABS analyzing outgoing network traffic */
out_threshold ∈ Real                               /* OAD threshold */
t ∈ ℕ                                              /* time value to wait for output */
pre-alerts = set of alerts                         /* alerts received from the NIDS monitoring incoming traffic */

INIT PHASE                                         /* IT specialists set out_threshold and t values */

TRAINING PHASE

INPUT:
    p: PACKET                                      /* outgoing network packet */

for t := 1 to τ                                    /* first, train the OAD with τ samples */
    oad.train(p.source.address, p.source.port, p.payload)  /* POSEIDON builds a profile for each monitored service */
end for

TESTING PHASE

INPUT:
    p: PACKET                                      /* outgoing network packet */

OUTPUT:
    true_alerts: set of alerts

for each a ∈ pre-alerts do                         /* checks if the packet belongs to a communication
                                                      marked as anomalous by the input NIDS */
    if (match_alert(a, p) = TRUE) then
        anomaly_score := oad.test(p.source.address, p.source.port, p.payload)
                                                   /* tests if the output is anomalous */
        if (anomaly_score > out_threshold) then
            a.true_alert := TRUE
            true_alerts.add(a)
        end if
        a.processed := TRUE
    end if
end for
```

```
for each $a \in pre - alerts$ do                    /* missing-output-response handling */
    if (a.processed = FALSE) and (current_time > t) then
        a.true_alert := TRUE
        a.processed := TRUE
        true_alerts.add(a)
    end if
end for
```