Selective alerts for runtime protection of distributed systems

M. Colajanni, D. Gozzi & M. Marchetti
Department of Information Engineering,
University of Modena and Reggio Emilia, Italy

Abstract

Network Intrusion Detection Systems (NIDS) are popular components for a fast detection of network attacks and intrusions, but their efficacy is limited by overwhelming amounts of false alarms that have to be manually managed by system administrators. In order to improve the efficacy of attack detection and reduce the amount of false positives, we propose a novel scheme for runtime alert management. It filters innocuous attacks by taking advantage of the correlation between the NIDS alerts and detailed information concerning the protected information systems, that is retrieved from heterogeneous and unstructured data sources. Thanks to the proposed scheme, an alert is sent to the system administrator only if an attack threatens some real vulnerability of the protected hosts. Otherwise, as it occurs in the large majority of the cases, the alert is stored for a subsequent offline analysis. The viability and efficacy of the proposed solution are demonstrated through an operative prototype that has been tested in networks subject to realistic attacks.
Keywords: intrusion detection system, network protection, false positive reduction, alert correlation, alert filtering.

1 Introduction

All modern networked information systems must be protected by some hardware or software appliance. Beyond the first line of defense represented by firewalls, the *Network Intrusion Detection Systems* (NIDS) are the most valuable technology for increasing the network security level through a continuous monitoring and analysis of the network traffic. A generic NIDS processes a copy of the traffic flowing through the protected networks, with the aim of finding illicit activities, attacks and intrusions. Hence, a properly configured NIDS is able to spot activities that have not been blocked by a firewall, either because these attacks rely on traffic that is considered licit or because of a misconfiguration.

Unlike the previous attacks that were mainly carried out manually, the vast majority of modern attacks are performed through self-replicating malware that is sent blindly against randomly chosen hosts. As a consequence, often a NIDS signals many attacks which cannot be effective because the targeted service is not available or because that vulnerability cannot affect any host of the protected network. These alarms are false positives for the system administrator, but a NIDS alone is not able to assess whether they represent a real threat for the protected machines. Nowadays, it is the network administrator responsibility to carry out for each alarm further examinations of the incidents reported by the NIDS, as well as to devise and deploy the proper countermeasures. Hence, administrators are forced to waste precious time for a manual analysis of irrelevant security alerts. If the false positive rate is too high, the number of false alerts can easily overwhelm the number of alerts related to real network attacks by several orders of magnitude, thus rendering the NIDS completely useless, although fully functional. Moreover, several attacker tools exist that are able to automatically trigger so called *alert storms*. They generate a large number of irrelevant alerts that overwhelm the investigation time and skills and avoid or delay attack detection.

Our goal is to allow the system administrators to focus on few relevant, high priority alerts. To this purpose, we describe a runtime alert filtering scheme that is able to discard most of the irrelevant alerts without the need of human intervention. This proposal effectively mitigates the issues related to false positives and allows a fast and focused reaction of the security team. While a typical NIDS assesses the possible risks by relying only on information conveyed by the network traffic, the proposed scheme correlates at runtime the NIDS alerts with other information coming from multiple, heterogeneous and unstructured data sources.

Once the NIDS detects an attack, the related alert is automatically examined in order to extract information related to the vulnerability that the attack tries to exploit. A list of vulnerable applications is then compared with the configuration of the host targeted by the attack. If a match is found, this means that the victim host is really vulnerable to the detected attack. As a consequence, the probability of a successful network intrusion is high, and the alert priority is raised to signal a request of immediate intervention. On the other hand, if the targeted host is not vulnerable to the attack, the alert priority can be safely lowered and stored for offline analysis.

Our approach is innovative in the way we correlate intrusion alerts with different sources of data, not just with alerts coming from other NIDS sensors. In such a way, we can filter NIDS alerts on the basis of information that cannot be determined by the analysis of network traffic alone. In designing and realizing the described prototype we faced several challenges, such as information extraction from unstructured data sources, constant update of vulnerability-related information, and alert processing speed. Both viability and effectiveness of the proposed solution are demonstrated through a fully operative prototype that is based on open source software.

The remainder of this paper is organized as follows. Section 2 describes the current shortcomings that characterize unranked NIDS alerts and motivates our innovative approach for the reduction of false positives. Section 3 outlines the proposed runtime scheme for filtering false alarms. Section 4 describes the most innovative solutions that are integrated into an operative prototype tested on a real scenario. Section 5 reports some concluding remarks.

2 Intrusion alert filtering and ranking

Several efforts have been devoted to the design and deployment of appliances for computer network security. As an example, network firewalls are included in the large majority of modern operating systems and hardware devices (e.g., routers, wireless access points, modems). Network intrusion detection systems are part of the security infrastructure, but their potential has been only partially exploited. Most organizations still utilize NIDS to log intrusion attempts. Collected data is examined only after an incident has been detected through other signals (e.g., bad server behavior, huge spread of worm infections, large increase of some types of network traffic). This approach to NIDS comes from the limitations of the early appliances, which were too slow to analyze network traffic at runtime. For these reasons, the traffic was recorded and analyzed in batch mode usually at night when the system load was low.

Nowadays, modern NIDS are able to analyze data at runtime and therefore they can be used to detect attacks while they are occurring. Once the NIDS has identified an illicit network activity, an alert is immediately generated and sent to the system administrator, which receives several useful information, such as the host that has been targeted by the attack and the vulnerability that the attack tried to exploit.

The worst problem that administrators have to face while dealing with the alerts generated by a NIDS is represented by the *false positive* rate (the impact of false positives on Intrusion Detection Systems reliability has analyzed in [1]). By false positive we mean all network activities that are licit or harmless to the protected systems, but that have been erroneously signaled by the NIDS as a security threat. We should also consider that several techniques can be utilized by skilled attackers to cover up the attack traces by forcing a NIDS to generate storms of irrelevant alerts [2, 3]. If the NIDS is working properly, the real attack is likely identified and signaled, but important alerts are hidden among several thousands of other irrelevant and misleading alerts. This kind of diversion is very effective because it prevents a timely deployment of proper countermeasures since the administrator is overwhelmed by the large amount of alerts.

Appropriate countermeasures can be deployed on time only if there is a way to distinguish between random attack attempts and possibly successful break-ins. We observe that the distinction between these two types of events depends on the software installed on the targeted machines: attempts at compromising a service which is not installed or attacks exploiting a vulnerability that has been patched do not deserve worrying.

The issue of intrusion detection alert management has been extensively addressed in literature. The first steps aimed to relieve the administrator from the burden of analyzing redundant alerts led to design of *alert fusion* techniques [4]. While alert fusion may help to aggregate several homogeneous alerts, it suffers several drawbacks that are solved by our proposal. First of all, aggregation itself is not useful to assess the real alert dangerousness. Moreover, alert fusion algorithms rely only on intrusion detection alerts, and do not leverage any external information source. Authors of [5] proposed a formal aggregation scheme able to correlate alerts exploiting the same vulnerability. To this purpose, their design utilizes vulnerability information, but the proposal is just a more sophisticated fusion algorithm that does not assess the threat level of an alert. Moreover, their aggregation scheme rely on a single, homogeneous and structured vulnerability database.

The process of automatically ranking vulnerabilities on the basis of a computed threat level has been defined *alert verification* in [6] and [7]. Following the taxonomy proposed in previous works, our proposal can be classified as a passive alert verification scheme. This means that our alert verification is carried out without an active interaction with the system(s) targeted by an attack. This characteristic is indeed desirable, and allows our system to verify alerts in real time, without imposing load on the monitored systems and without relying on results generated by a possibly compromised machine. In [6] and [7], authors propose an alert verification scheme based on a single, structured vulnerability database. The dependency on a single information source is clearly identified as a drawback in [8], due to possible incompleteness, lack of timely updates, and lack of trust in the single information provider.

The main strength of our approach, that differentiates the proposed alert verification scheme from all the previous works, is the ability to leverage a vast amount of vulnerability data, built upon multiple, heterogeneous and unstructured information sources. This critical task is carried out by parsing and semantically tagging unstructured information, with techniques borrowed by the data mining field.

Worth noticing works in which data mining techniques are applied in the network intrusion detection context are [9] and [10]. However, in those papers data mining techniques are proposed for attack detection, while intrusion alert management is not considered.

Other attempts at identifying complex attack scenarios, so as to help human users, have been discussed in [11]. However, that approach is suitable for offline analysis, while the solution proposed in our paper is able to perform runtime threat evaluation, thus allowing for a timely deployment of proper countermeasure(s).

A lower amount of false positive alerts augments the runtime analysis efficacy and allows a broader application of the so called *Network Intrusion Prevention Systems* (NIPS) [12]. They are the most modern versions of NIDS that are able to automatically deploy some proactive countermeasure(s) after an alert, for example by intervening on firewall rules at runtime.

3 System design

The proposed scheme for prioritizing alerts and reducing the false positive ratio relies on the correlation of data coming from three heterogeneous domains:

1. **NIDS alerts**.
2. **detailed description of the software installed on each machine** of the protected network.
3. Several external **online vulnerability reports**.

Our aim is to integrate at runtime the existing knowledge on vulnerabilities and local software configurations with the intrusion alerts raised by the NIDS. By knowing the real vulnerabilities that affect some machines of the protected network, we can highlight the attacks exploiting these vulnerabilities by raising the priority of the related NIDS alerts. On the other hand, the priority of alerts related to attacks trying to exploit vulnerabilities that do not affect the hosts can be safely lowered, and their analysis postponed. Although one might argue that known vulnerabilities should be corrected as soon as they are found, several circumstances exist in which it is impossible or extremely difficult to correct known software vulnerabilities in a short time. Let us consider as examples:

- there is a time between a vulnerability disclosure and the patch released by the software producer;
- within large organizations, comprising a large number of hosts with heterogeneous software and hardware platforms, updating may take a long time;
- a patch may introduce an incompatibility with previous software versions (for example if the organization is using legacy software) and not installed on purpose;
- some vulnerability is protocol specific and no immediate patch exists.

The design of the proposed architecture is shown in Figure 1.

Intrusion alert information is commonly expressed as one or more identifiers in the context of heterogeneous online vulnerability repositories, that can be freely consulted (Open Source Vulnerability Database (http://osvdb.org/), Common vulnerabilities and exposures (http://cve.mitre.org/), National Vulnerability Database (http://nvd.nist.gov/nvd.cfm), Packet Storm security advisories (http://www.packetstormsecurity.org/alladvisories/), SecurityFocus^{TM}vulnerabilities (http://www.securityfocus.com/vulnerabilities/)). From these sources, we extract a complete list of the vulnerable operating systems and applications, including their revision numbers.

To leverage the knowledge contained in the several external data sources that can be used by the proposed systems, it is necessary to convert the heterogeneous and unstructured data in a structured and homogeneous vulnerability database. This task is performed by means of parsers. A parser is in charge for analyzing data belonging to a single data source and producing as output structured information compliant with a standard data format, used by all the parsers. Parsed information is then materialized in the *structured vulnerability information database*, ready to be used for alert filtering and prioritizing. Depending on the source size and on the computational complexity related to its analysis, the parsing of a complete data source may require several hours. However, once an information source has been parsed for the first time, it is easy to keep the structured vulnerability database constantly updated by parsing only new and recently modified elements in the data source.

The *installed software database* is used to store and retrieve all the knowledge related to the software installed on the machines contained in the protected network. It is a main administrator duty to report which software and version are installed on each machine and to keep that list updated and consistent. While it does not exist a single automated tool able to perform this task without human intervention, useful information can be gathered from package management software. However, we remark that a network administrator should already know the exact configuration of the administered machines to be able to apply the correct patches and to keep the software updated.

The *alert filter* is the central element of the proposed architecture. Its purpose is to receive all the alerts generated by the intrusion detection system, to evaluate their level of threat for the protected machines and to rank each alert based on that treat level. Ranked alert are then presented to the administrator for further analysis.

For each alert generated by the intrusion detection system, the alert filter extracts the information related to the vulnerability that the attacker tried to exploit and to the targeted host. After having identified the vulnerability, the alert filter looks up all the vulnerable applications in the structured vulnerability information database. It is worth noticing that, assuming that the database uses a unique vulnerability identifier (as example, the vulnerability ID used by the NIDS) as an index, this

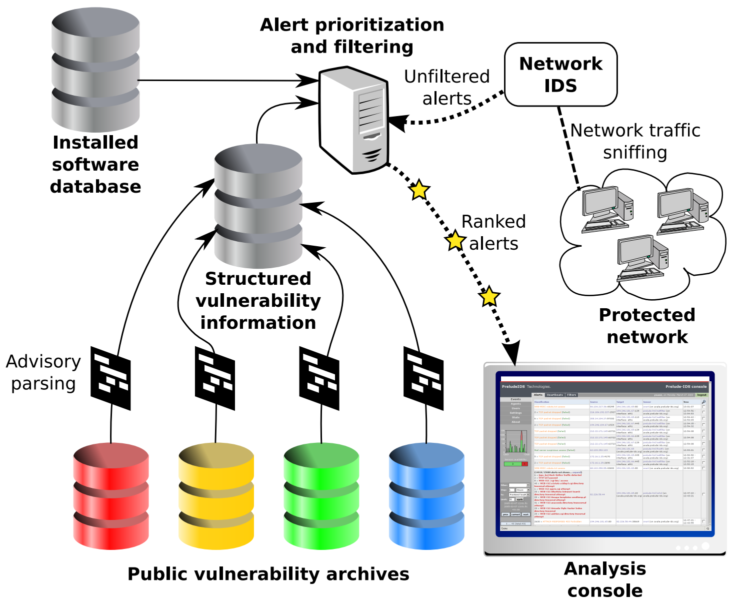

Figure 1: Architecture design.

operation can be performed in a very small time. Concurrently, the identifier of the host targeted by the attack (e.g., its IP address) is used to retrieve a complete list of the installed applications. After that, those two lists are compared in order to identify if the attacked system runs one or more vulnerable applications. If this is the case, then there is an high probability that the attacker succeeded in compromising the targeted machine, and the alert priority is increased. Otherwise, the attack failed, and the alert priority can be safely lowered. We remark that the comparison among the two application lists (which usually contain only a few elements each) is very low, and comparable, if not lower, to the computational complexity of the packet analysis performed by the intrusion detection system. Hence, all this operation can be performed at runtime with respect to the NIDS activities.

Our tool is able to effectively highlight the most significant alerts in real time, thus allowing system administrators to focus their attention just on real intrusions, without losing precious time in the analysis of a multitude of insignificant and misleading alerts.

4 Integrated NIDS prototype

4.1 Architecture

The proposed data aggregation and runtime alert filter is implemented through a combination of open source software that we can modify and integrate for our purposes. The main software components are based on the *Snort* NIDS which is the most popular signature-based network intrusion detection system, and on the hybrid IDS framework named *Prelude*. To facilitate prototype operations, we have implemented a graphical user interface and administration console through the *Prewikka* software.

The Prelude manager is configured in alert-relaying mode. A custom manager module specifically implemented in this project applies its previous knowledge to each incoming alert.

Vulnerability descriptions are gathered off-line from existing archives (currently from Bugtraq) and are updated periodically. We built a database of known vulnerabilities through these Bugtraq descriptions and a custom parser. We assume that a suitable list of installed software for each machine of the organization is provided by the administrator. Every intrusion alert triggers a lookup in the vulnerability database, from which we retrieve the list of software targeted by the attack. Thanks to offline data mining operations, a lookup in the stored list of installed software for the targeted machine is sufficient to know if there is a real risk. Intrusion alerts which pass this validation are marked as high priority, while other alerts are kept as a proof for later investigation.

4.2 External data source integration

Intrusion alerts are marked by a unique identifier of the triggered rule (called *Snort Identifier*, or SID in Snort) and each rule has a reference to a Web page describing the vulnerability. Most rules reference online vulnerability description repositories, so the list of applications affected by a specific vulnerability can be obtained through a parser that is able to extract information from vulnerability reports.

By examining the set of Snort rules freely available from Sourcefire, Inc. we noticed that there are a few major sources for rules references: CVE, Nessus, ArachnNIDS (arachNIDS was the *Advanced Reference Archive of Current Heuristics for Network Intrusion Detection Systems*, its development was discontinued in 2001, the project was hosted on http://whitehats.com/), Bugtraq, Microsoft, Application Security Inc. and McAfee Inc. Instead of relying on one information source, our choice is to merge data coming from several sources, thus achieving a complete and vendor-agnostic list of vulnerabilities and vulnerable applications.

We extract data from the vulnerability report Web pages that the advisory service is offering and integrate an external data source through a suitable parser specifically implemented for this purpose. In particular, the parser is able to transform the unstructured or semi-structured advisory data into a structured description of the vulnerabilities, that is directly usable by the alert filtering and ranking element.

The deployed NIDS (or NIDSs, when a network requires more than one intrusion detection sensor) is not directly affected by the alert aggregation framework. Alerts are issued through a remote logging facility, while filtering and ranking occur only on the centralized alert database.

From the alerts issued by the NIDS sensors we are able to extract the IP address of the targeted host and the TCP or UDP destination port, so as to determine the service which is being attacked. The identifier of the triggered Snort rule is utilized to extract the vulnerability description from the archive which contains it (Snort rules references span over many archives), while a lookup in the installed applications database is sufficient to know the exact name and version of the actually running application. Searching the running application in the list of the applications affected by the attack is all it takes to determine if there exists a real threat. If this is the case, the priority of the alert is raised. This technique allows for effective mitigation of *alert storms*, diversion attacks produced by a variety of automated tools.

At any given time, there are some vulnerabilities which are known only to a small group of security researchers, software developers or criminals. The research community refers to these vulnerabilities as *0-day*. Since they are not found in public vulnerability archives, the corresponding alerts (if any are issued) will not be assigned a high priority by our ranking system. There are some generic Snort rules which raise alerts upon intercepting byte sequences that are commonly found in remote exploits such as simple unencoded shellcodes. However, the incident handling operations do not differ from what is traditionally done with an unfiltered/unranked alert system: look up past records to find the attacks, if they triggered some minor generic alert.

4.3 System management

A list of installed software and operative systems may be prepared directly by the administration personnel or may be inferred through a fingerprinting tool such as *nmap* from a remote host. (The identification of remote services and operative systems using fingerprinting is just one of the features offered by the Nmap scanner.) In either case, the proposed system knows if the attack has likely succeeded or the alert was just a false positive by comparing the software sustaining the attack to the list of targeted software.

The Prelude hybrid IDS system provides alert aggregation from any intrusion detection system which supports logging through the Prelude interface library (*libprelude*). The task of examining security alerts is carried out by a human operator through an appropriate graphical interface. Prewikka is a Web interface to the Prelude alert database which helps investigating security incidents and tracking the deployed intrusion detection sensors status. High-priority alerts are emphasized in the alert view presented to the security team, so as to highlight the attacks which were most likely successful.

4.4 System validation

To verify the effectiveness of the proposed solution, we tested our prototype in different controlled network environments. We carried out several experiments for different system and workload scenarios. In each of them, we injected some network attacks targeting vulnerable applications installed in the protected hosts, as well as several irrelevant attacks. Our system has always been able to evidence the really dangerous attacks by raising the priority of their alerts. High priority alerts were also clearly distinguishable from ordinary alerts in the graphical user interface.

We remark that the delay introduced by the alert filtering at runtime is minimal, and fully compatible with runtime traffic analysis and system protection. The operations performed by the alert ranking framework require only two searches in indexed fields (identifier of the vulnerability and IP address of the targeted machine) and the comparison of two lists of strings (the names of the vulnerable software and the currently installed software). The efficiency of runtime operations is guaranteed by a continuous offline data mining that relies on parsers to integrate data from multiple and heterogeneous data sources.

5 Conclusions

We present a novel scheme to filter false alarms by correlating network intrusion functions with previously acquired knowledge in the domain of security vulnerabilities and local software deployment. This proposal greatly reduces the need for extensive manual analysis. The most immediate benefit is the chance to know at runtime if a serious threat is endangering the protected network, instead of noticing it only after a successful inter alert correlation which may require more than one host to be compromised. This effect is achieved through a substantial and completely automated filtering of false positives, that is carried out by applying prior knowledge concerning the administered domain. This filtering process highlights only the alerts that represent a real security threat for the protected systems, thus relieving the network administrator of the burden of manually analysing a multitude of irrelevant and misleading alerts. Both viability and effectiveness of the proposed approach have been demonstrated through an operative prototype, based on well known and widely deployed open source software. Prototype performance is compatible with runtime filtering of intrusion alerts.

References

[1] Axelsson, S., The base-rate fallacy and its implications for the difficulty of intrusion detection. *ACM Conference on Computer and Communications Security*, pp. 1–7, 1999.

[2] Mutz, D., Vigna, G. & Kemmerer, R., An experience developing an IDS stimulator for the black-box testing of network intrusion detection systems. *Proceedings of the 2003 Annual Computer Security Applications Conference (ACSAC '03)*, Las Vegas, Nevada, pp. 374–383, 2003.

[3] Patton, S., Yurcik, W. & Doss, D., An achilles' heel in signature-based ids: Squealing false positives in snort, 2001.

[4] Li, Z., Chen, Y. & Beach, A., Towards scalable and robust distributed intrusion alert fusion with good load balancing. *Proc. of the SIGCOMM Workshop on Large Scale Attack and Defense (LSAD06)*, 2006.

[5] Morin, B., Mé, L., Debar, H. & Ducassé, M., M2d2: A formal data model for ids alert correlation. *Proc. of the 5th symposium on Recent Advances in Intrusion Detection (RAID 2002)*, 2002.

[6] Valeur, F., Vigna, G., Kruegel, C. & Kemmerer, R.A., A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 2004.

[7] Kruegel, C. & Robertson, W., Alert verification: Determining the success of intrusion attempts, 2004.

[8] R. Gula, Tenable Network Security, Inc., Correlating IDS alerts with vulnerability information, 2002. Available at http://www.nessus.org/whitepapers/va-ids.pdf.

[9] Lee, W. & Stolfo, S.J., Data mining approaches for intrusion detection. *Proc. of the Seventh USENIX Security Symposium (SECURITY '98)*, 1998.

[10] Portnoy, L., Eskin, E. & Stolfo, S.J., Intrusion detection with unlabeled data using clustering. *Proc. of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, 2001.

[11] Ning, P., Cui, Y. & Reeves, D., Constructing attack scenarios through correlation of intrusion alerts, 2002.

[12] Botwicz, J., Buciak, P. & Sapiecha, P., Building dependable intrusion prevention systems. *Proc. of International Conference on Dependability of Computer Systems*, 2006.