2010 10th IEEE International Conference on Computer and Information Technology (CIT 2010)

Selective and early threat detection in large networked systems

Michele Colajanni, Mirco Marchetti, Michele Messori
Department of Information Engineering
University of Modena and Reggio Emilia
Modena, Italy
{michele.colajanni, mirco.marchetti, michele.messori}@unimore.it

***Abstract*—The complexity of modern networked information systems, as well as all the defense-in-depth best practices, require distributed intrusion detection architectures relying on the cooperation of multiple components. Similar solutions cause a multiplication of alerts, thus increasing the time needed for alert management and hiding the few critical alerts as needles in a hay stack. We propose an innovative distributed architecture for intrusion detection that is able to provide system administrators with selective and early security warnings. This architecture is suitable to large networks composed by several departments because it leverages hierarchical and peer-to-peer cooperation schemes among distributed NIDSes. Moreover, it embeds a distributed alert ranking system that makes it possible to evaluate the real level of risk represented by a security alert generated by a NIDS, and it allows independent network departments to exchange early warnings about critical threats. Thanks to these features, a system administrator can focus on the few alerts that represent a real threat for the controlled infrastructure and can be notified about the most dangerous intrusions before his department is attacked.**

***Keywords*-Alert ranking; early warning; intrusion detection systems; distributed IDS;**

I. Introduction

The vast majority of attacks targeting networked systems are carried out by self-replicating malware and automatic tools for vulnerability discovery and exploitation. These tools randomly scan the Internet address space looking for reachable targets, and attack them by trying to exploit known vulnerabilities affecting common operating systems and applications. As a consequence, NIDSes detect many attacks which cannot be effective because the targeted service is unavailable or because the exploited vulnerability does not affect any host of the controlled network. From the system administrator's perspective, these alarms are considered as false positives, but a NIDS alone is unable to assess whether they represent a real threat for the protected machines. It is the network administrator responsibility to examine each 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. The number of false alerts produced by a NIDS can easily overwhelm the number of alerts related to real network attacks by several orders of magnitude [1], thus making the NIDS useless in practice, although fully functional. Moreover, several attacker tools exist that are able to trigger *alert storms* [2], [3]. They generate a large number of irrelevant alerts among which the few relevant alerts related to real security threats are hidden, thus hindering the alert management and attack recovery activities.

We should also consider that large network-based information systems cannot be effectively monitored by one centralized NIDS. Complete coverage of a realistic network environment can only be achieved by implementing a distributed architecture for intrusion detection, comprising multiple cooperative NIDSes deployed throughout the protected network. However, the multiplication of NIDSes implies an increase in the number of alerts that are generated within the protected network, and that have to be manually validated and analyzed by the system administrator.

In this paper we address these issues by proposing a novel architecture that is suitable for monitoring large organizations' networks, composed by several interconnected departments, each comprising a multitude of smaller network segments. The proposed architecture relies on distributed NIDSes deployed in several network departments. It is characterized by the integration of three novel features, that are:

- a hybrid communication scheme that integrates hierarchical cooperation at the intra-department level and peer-to-peer cooperation at the inter-department level;
- a novel distributed alert ranking scheme, that is able to assess whether a NIDS alert related to a detected intrusion attempt represents a real threat for the attacked machine;
- a selective alert sharing service that propagates early warnings related to the most dangerous security threats towards departments that have not been attacked yet.

The hybrid cooperation scheme of the proposed architecture, based on both hierarchical and peer-to-peer cooperation strategies, facilitates its deployment in complex network environments. With respect to purely hierarchical architectures [4]–[11] it guarantees higher scalability and fault tolerance, thanks to the absence of a single point of failure. Moreover, it allows the deployment of the proposed architecture in large network environments, in which it is impossible to identify a single central authority. On the other hand, unlike purely peer-to-peer architectures [12]–[20], the proposed cooperation strategy does not require the exchange of sensitive information among networks, and allows a network administrator to carefully

978-0-7695-4108-2/10 $26.00 © 2010 IEEE
DOI 10.1109/CIT.2010.124

design the path followed by alerts detected in his network. To the best of our knowledge, this is the first proposal of a hybrid architecture for monitoring complex networks belonging to a large organization.

With respect to other approaches for NIDS alert fusion and ranking, the proposed scheme is able to leverage multiple external data sources, to rank alerts at run-time, and to share the computational load among several distributed components. Moreover, the proposed architecture is able to automatically generate and distribute early warnings about malicious activities that could jeopardize the integrity of critical systems throughout the monitored network. If an attack is detected within a department, it is likely that other network departments belonging to the same organization will be attacked as well. Even though the detected attack poses no real threat to any of the machines in the attacked department, critical and vulnerable machines may be deployed in other networks. If this situation is detected, the proposed architecture is able to issue early and selective warnings to the administrators of these critical and vulnerable systems, allowing them to deploy proactive security countermeasures rather than to recover compromised systems.

Related work on NIDS alert management and distributed intrusion detection is discussed in Section II. An overview of the proposed architecture is presented in Section III. Detailed descriptions of the alert ranking system and of the cooperation strategies among distributed NIDSes are in Section IV and Section V, respectively. A prototype implementation of the proposed architecture is described in Section VI. Some concluding remarks are reported in Section VII.

II. Related work

This section presents related works in the context of NIDS alert management and distributed architectures for intrusion detection.

The process of automatically ranking alerts on the basis of threat level has been defined *alert verification* in [21] and [22]. According to the taxonomy proposed in these papers, our proposal can be classified as a passive alert verification scheme, meaning that alert verification does not require active interaction with the system(s) targeted by an attack. This characteristic is desirable, and allows our system to rank alerts at run-time, without imposing load on the monitored systems and without relying on results generated by a possibly compromised machine. In [21] and [22], authors propose an alert verification scheme based on one vulnerability database. The dependency on a single information source is identified as a drawback in [23], because of possible incompleteness, lack of timely updates, and lack of trust in the single information provider. The main strength and original contribution of our alert ranking approach is the ability to leverage multiple, heterogeneous and unstructured information sources. This critical task is carried out by parsing these information sources through techniques borrowed from data mining.

The typical approach for the design of distributed intrusion detection architecture is based on a centralized scheme [4]–[11]. All these architectures consist of several sensors getting events produced by different sources, and of a centralized aggregator analyzing the security alerts.

With the increase in size and complexity of the monitored environment, the computational capacity of one centralized alert aggregator becomes a bottleneck. Hierarchical processing schemes [12]–[20] aim to improve scalability by building a hierarchy of intermediate aggregators, each processing a subset of alerts and relaying their partial results to the aggregators at the higher tier. While a hierarchical architecture can scale much better than a centralized architecture, both systems suffer of a single point of failure.

Several distributed NIDS architectures based on a peer-to-peer scheme have been proposed to address this issue [24]–[32]. High scalability and, in some cases, good load balancing are other benefits but the lack of a centralized alert aggregator poses new challenges. Since there is not an entity with a complete view of the environment, it is necessary to devise new intrusion detection algorithms based on partial knowledge, and to define alert routing techniques that enable distributed alert aggregators to receive all the needed information. Moreover, cooperation by the exchange of IDS alerts may reveal sensitive information to cooperative nodes, such as the structure of the internal network, or the fact that some machines have been compromised.

The hybrid architecture proposed in this paper is designed to be deployed within one organization, and the peer-to-peer layer is used to distribute early warnings for network intrusions to which critical systems of the protected organization are vulnerable. Moreover, our architecture performs distributed alert ranking, thanks to the integration of a novel alert ranking system.

To the best of our knowledge the only other distributed architecture that combines peer-to-peer and hierarchical cooperation approaches is represented by DOMINO [33]. However, unlike our proposal the DOMINO architecture does not represent a complete solution for NIDS alert management, and only focuses on the generation of IP blacklists. Moreover, it requires Internet-wide sensors deployment, thus not being applicable to the network infrastructure of one organization.

III. Architecture overview

The architecture proposed in this paper suits the network infrastructure of large organizations. These networks are divided into several independent and possibly geographically distributed departments. Moreover, each department is composed by multiple smaller and interconnected networks.

Complete coverage of a similar IT infrastructure can be achieved by deploying a distributed intrusion detection architecture, comprising several distributed NIDSes. Architectures relying on one centralized alert correlator are not suitable, since it would represent a bottleneck and a single point of failure. Moreover, a centralized correlator represents an issue from an organizational viewpoint: while it is realistic to assume that each department has an administrator, that is responsible for all the networks belonging to his department,

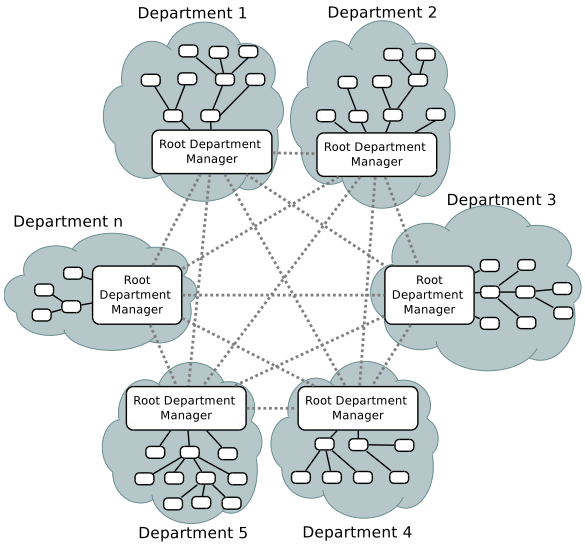

Fig. 1. High-level view of the proposed architecture

a centralized authority responsible for all the heterogeneous and loosely coupled department's networks may not exist. Sometimes security warnings gathered within each department may be considered as classified, thus preventing a department administrator to directly access security warnings related to other departments. As a result, useful security information are often withheld.

The proposed architecture combines hierarchical and peer-to-peer NIDS cooperation schemes, together with distributed alert ranking servers (see Section IV), to achieve high scalability, resiliency to failures and the ability to adapt to complex network environments. The alert ranking mechanism allows department administrators to selectively focus on relevant security alerts, and makes it possible to exchange selective and early warnings without requiring tight coupling and strong trust relationships among different departments.

A high-level representation of the proposed architecture is given in Figure 1.

The network shown in this figure has 6 departments, and each department consists of several interconnected subnetworks. To provide complete network coverage, it is necessary to deploy multiple NIDSes in each department. All the NIDSes that belong to the same department are part of an intra-department hierarchical architecture, whose structure reflects the topology of the network of the monitored department. The root of this hierarchy is a single logical component, named *Root Department Manager*. All the alerts generated within a department are collected, evaluated and relayed to the root department manager, that provides a comprehensive network security report to the the department administrator. The intra-department hierarchical architecture is described in Section V-A.

The proposed architecture provides an inter-department alert sharing service by allowing root department managers belonging to different departments to communicate through a decentralized and unstructured peer-to-peer overlay, as shown by the dotted lines in Figure 1. The inter-department alert sharing is described in Section V-B.

Both the intra-department hierarchical architecture and the inter-department alert sharing service rely on the alert ranking mechanism presented in Section IV.

IV. NIDS ALERT RANKING

This section describes the main functions of the NIDS alert ranking system. The proposed strategy for alert ranking and prioritization relies on the correlation of data coming from three heterogeneous domains:

- vulnerability information gathered from external, open and unstructured sources;
- software configuration of the machines deployed in the protected network;
- network security alerts generated by NIDSes.

The aim is to integrate at runtime the existing knowledge on vulnerabilities and local software configurations with the intrusion alerts raised by the NIDSes. By knowing both the software configuration of the machines connected to the monitored network and the software that are vulnerable to network attacks, the system can check whether a machine contains one or more software packages that are vulnerable to a specific attack. If a NIDS raises an alert related to an intrusion attempt to which the targeted machine is vulnerable, then the targeted machine has been compromised with high probability. Hence, the related alert should be analyzed by the network administrator as soon as possible, thus hastening the incident response procedures.

On the other hand, if the alert reports an intrusion attempt to which the targeted machine is not vulnerable, than the attack has not been successful. Since the alert does not pose an immediate threat to the integrity of the protected systems, its priority can be lowered, and its analysis can be safely delayed.

The design of the alert ranking system is shown in Figure 2, that evidences three main components: the *vulnerable software database*, the *Configuration Management DataBase* (CMDB) and the *alert ranking server*.

The vulnerable software database stores associations between known vulnerabilities and the list of vulnerable software. Vulnerability information are provided by several heterogeneous and unstructured open sources, that can be freely consulted on the Web[1]. We use a custom crawler to download unstructured vulnerability information and parse them through custom wrappers. Each wrapper is designed to parse all the vulnerability advisories produced by a specific external source, and to extract the list of vulnerable software. Parsed information is then materialized in the vulnerable software database, ready to be used for alert ranking. While each

[1]notable examples are: 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™ vulnerabilities (http://www.securityfocus.com/vulnerabilities/)

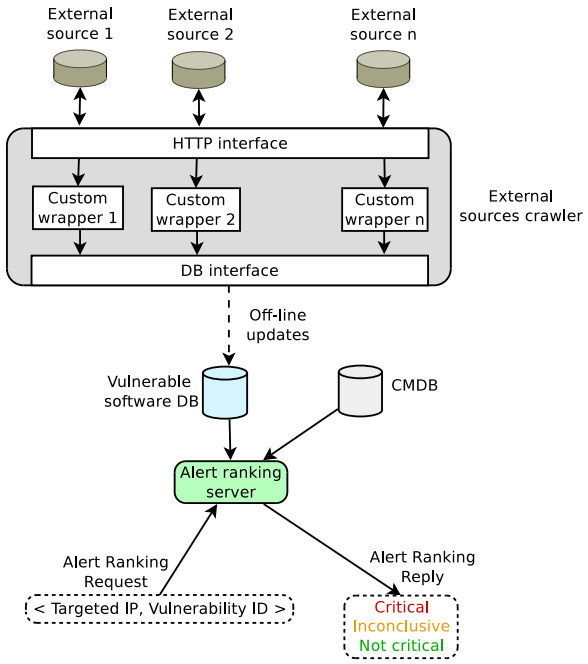

Fig. 2. Architecture of the alert ranking system

external source uses a different system to classify and label the known vulnerability, the large majority of vulnerability advisories are cross referenced. Hence it is possible to identify the advisories published by different sources but related to the same vulnerability. Depending on the number of vulnerabilities known by each source, on the computational complexity of the operations that the wrapper has to perform, and on the speed of the Internet connection used to access to the source, the complete parsing of all he vulnerabilities may require several hours. However, once a complete scan of an open source has been performed for the first time, it is easy to keep the structured vulnerability information database constantly updated by parsing only the advisories that has been recently introduced or updated. It is also important to observe that these operations can be performed off-line at regular time intervals, without introducing any delay in the alert ranking process.

The *Configuration Management Database* (CMDB) is a structured information source (typically, a relational database) that contains authoritative and complete information on all the devices, software and services belonging to the IT infrastructure. For the scope of this paper, we only require that the CMDB includes updated information on the software configuration of all the machines connected to the network that we want to monitor. Moreover, as explained in more detail in Section V-A, we do not require the deployment of a single, centralized CMDB containing configuration information for all the machines connected to the network of a large and complex organization, or even to a single department. Configuration information can be partitioned in several local databases, managed directly by administrators that are only responsible for small networks.

The *alert ranking server* is the the central component of the proposed alert ranking system, whose purpose is to fuse information coming from the CMDB, the vulnerable software database and the NIDS alert. The alert ranking process is initiated when the alert ranking server receives an alert ranking request, including two mandatory fields: the IP address of the machine targeted by the attack, and an identifier of the vulnerability that the attack tried to exploit. These information can be easily extracted from the alerts produced by signature-based NIDSes. As an example, all the alerts produced by Snort [34] (a widely deployed open source NIDS) include a signature-id that ties the alert to the related signature definition. When applicable, the signature definition contains a reference to vulnerability advisories of several open sources. Hence it is possible to infer the vulnerability ID starting from the signature ID included in an alert. This approach is applicable to all the signature-based NIDSes.

After having received the request, the alert ranking server performs a query to the CMDB using the IP address as a key in order to retrieve the list of software deployed on the attacked machine. A second query is executed against the vulnerable software database using the identifier of the vulnerability as a key, to retrieve the list of vulnerable software. The two queries are independent, and can be executed concurrently. When the two lists of software have been retrieved, they are compared by the alert ranking server. If the software configuration of the targeted machine includes at least one of the vulnerable software packages, then the attacked machine has been compromised with high probability, and the alert is ranked as *Critical* by the alert ranking server. On the other hand, if there is no match between the two lists of software, the attack does not represent a real threat for the targeted machine, and the alert is ranked as *Not critical*. It is also possible that the alert ranking server is not able to compute a rank for an alert. This may be caused by failures of systems and networks (e.g., a database that becomes unreachable) or by lack of data in the CMDB or in the vulnerable software database. I this situation the result of the alert ranking process is *Inconclusive*, meaning that the alert ranking server has not been able to generate a reliable result.

The distributed architecture proposed in this paper leverages this alert ranking strategy by integrating an alert ranking server into several distributed components, as described in Section V.

Alert ranking can be performed at run-time with respect to the NIDS activities. Indeed, the computational cost related to the comparison among the two software lists (which usually contain only a few elements each) is very low, and comparable, if not lower, to the computational complexity of the in-depth packet analysis performed by a signature-based NIDS, in which the payload of a packet has to be matched against hundreds of rules.

V. Cooperation among distributed NIDSes

The hybrid architecture proposed in this paper combines two cooperation schemes: a hierarchical architecture for en-

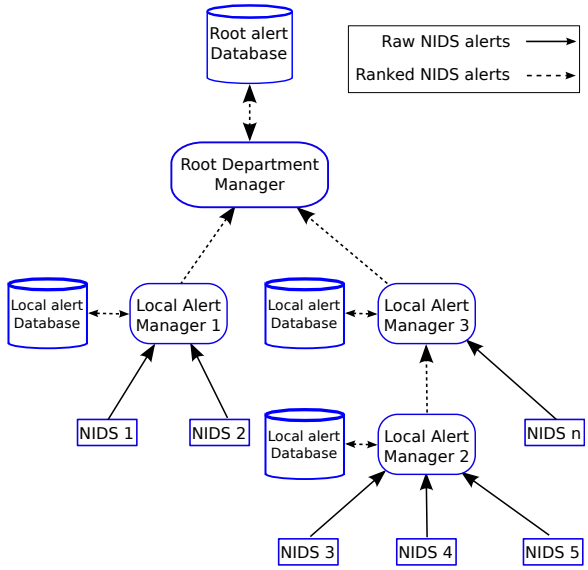

Fig. 3. Hierarchical architecture for intra-department network monitoring

abling cooperation among NIDSes deployed within the same department, and a peer-to-peer overlay to share early critical alerts among different departments.

A. Intra-department hierarchical architecture

The main components of the hierarchical architecture, together with a representative example of their interconnections, are presented in Figure 3.

The lowest layer of the hierarchical architecture is composed by NIDSes that independently monitor different networks of the same department. All the alerts generated by these NIDSes are relayed to a *local alert manager*. The purpose of the local alert manager is to collect all the received alerts, to apply the alert ranking mechanism presented in Section IV and to relay the ranked alerts to a local alert manager belonging to the higher tier of the hierarchical architecture. In the proposed design, a network is represented by its local alert aggregator, and changes in its topology or in the software configuration of its machines do not affect the higher levels of the architecture. It is also worth noticing that it is not necessary for the aggregation hierarchy to be shaped as an n-ary tree, or to be balanced. The in-degree of each element in the hierarchy, modeled as a directed graph, can be adapted to the characteristics of each deployment scenario. Similarly, the height of the sub-trees connecting each NIDS to the root department manager can also vary.

In order to perform alert ranking, each local alert manager integrates an alert ranking server. This server has to access the software vulnerability information database and a CMDB including the software configuration of all the machines that are monitored by the NIDSes to which the local alert manager is directly connected. As an example, the *local alert manager 2* in Figure 3 has to be able to access a CMDB containing the software configuration of all the machines monitored by *NIDS 3*, *NIDS 4* and *NIDS 5*, while the *local alert manager 3* only needs software configuration information for the machines monitored by *NIDS n*. Hence there is no need of one centralized CMDB containing the complete configuration of all the machines belonging to the department network. By relying on several local alert aggregators it is possible to share the load related to alert ranking among multiple distributed components. Each local alert aggregator has to rank only the alerts that are generated by a NIDS, while alerts received by lower-level local alert aggregators have already been ranked, and have to be relayed to the upper tiers of the hierarchy.

Each local alert manager also stores the ranked alerts in a *local alert database* that can be accessed by the administrator of the corresponding network to monitor the activities carried out within his domain. This chance eliminates the need to ask for this information to the department administrator, or to access department-wide security information that may be classified.

The *root department manager* represents the root of the hierarchical architecture used to monitor a whole department. By design, a root department manager is only connected to lower-layer local alert manager, and has no direct connections with any NIDS. Hence, all the alerts received by the root department manager have already been ranked. Each root department manager stores the received alerts in the corresponding *root alert database*, and integrates a graphical dashboard that provides the department network administrator with a graphical representation of all the alerts that have been generated within the department network. Since alerts are sorted with respect to their rank, the network administrator can easily focus on the critical alerts (i.e., alerts related to attacks that targeted a vulnerable machine) while delaying the analysis of alerts related to non critical attacks.

B. Inter-department alert sharing service

We identify two scenarios in which information sharing among departments is beneficial for the whole organization:

- successful attacks targeting machines that provide services to several departments;
- attacks that could compromise the integrity of critical systems.

As an example of the first scenario, let us consider an intrusion attempt targeting a host that provide *Single Sign On* (SSO) services for several departments of the same organization. While this machine is deployed in a single department, a successful attack could compromise the security of all the departments that rely on the SSO to authenticate users. In this situation, it is important to share the knowledge about attacks targeting the machines providing the SSO service with all the interested root department managers, so that all the network administrators can be notified of possible ongoing security breaches.

In the second scenario an attack is detected in a department, to which a critical machine in another department (such as a database containing critical information, or a Web server hosting an e-commerce Web site) is vulnerable. Without

selective sharing of security threats, the network administrator of the department that hosts the critical machines would not receive any alert, because the attack has been detected in a different department. However, the same alert is useful for the administrator of the department hosting the critical machine, since it represents an early warning of a dangerous attack. This early warning may allow the network administrator to deploy the proper countermeasures proactively, before the vulnerable machine is targeted by the attack.

To allow peer departments to exchange relevant security information while maintaining complete independence, all the root department managers representing different departments are interconnected through a inter-department peer-to-peer network. This higher level network allows root department managers to share alerts related to network intrusion that could jeopardize the integrity of critical systems within a department, as well as systems that provide critical services to several departments (such as single sign on). Information sharing at this level allows the proposed architecture to distribute early warnings to administrators of different departments about critical security threats before their department is directly involved. This feature makes it possible to deploy proactive countermeasures, thus increasing the security of the most critical systems throughout the organization's IT infrastructure.

As described in Section V-A, each root department manager receives all the alerts generated within the monitored department, that have already been ranked by the local alert aggregators. However, these ranks are only relevant within the department, since they have been computed against the software configuration of the attacked machines. Hence they do not indicate the level of threat that the alert may represent for critical systems deployed throughout the organization. To assess whether the alerts represent a threat for critical systems, all the root department managers integrates an alert ranking server, and have access to a CMDB that includes the software configuration of a small subset of critical machines deployed throughout the organization's IT infrastructure, called *Critical CMDB* (CCMDB). Each department administrator contributes to the CCMDB on a voluntary basis, by adding the software configurations of machines that are deployed within his department and that he considers critical (and not adding any classified information). When a root department manager receives an alert, this alert is ranked against all the machines included in the CCMDB. If the alert is ranked as critical, meaning that at least one critical machine is vulnerable to a detected attack, the root department manager broadcasts the related NIDS alert to all the other root department managers. Each root department manager can then validate the ranked alert and add it to its root alert database, thus making it available to the administrator through the alert visualization interface.

VI. Prototype implementation

The feasibility of the proposed architecture is demonstrated through a prototype based on open source software and custom integration modules. The NIDSes used to generate alerts are unmodified instances of the well known Snort [34] open source NIDS. The prototype has been validated experimentally in controlled conditions as a whole and in its individual components through the injection of network traffic containing known attacks.

A. Integration of external sources

Intrusion alerts issued by Snort are marked by a unique identifier of the triggered signature (called *Snort Identifier*, or SID). The SID can be used as research key to access further information related to the signature. When applicable, these information contain reference to online vulnerability description repositories, which in turn contain the list of vulnerable software. So the list of applications affected by a specific vulnerability can be inferred through a custom 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 several sources for rules references. In particular we focus on the two open sources that provide the most complete information related to vulnerable software[2]: CVE and Bugtraq.

Integration of external information sources is carried out by the *external sources crawler* in Figure 2. In the current prototype, this component is implemented as a set of Python scripts, and implements three wrappers. The first wrapper is designed to access to all the information related to the SID of all Snort's signatures, by executing HTTP requests to the URL http://www.snort.org/search/sid/X, where X represents the SID associated to a specific rule. The downloaded web page is then parsed, looking for links of vulnerability advisories provided by CVE and Bugtraq websites. If those links are found, the related vulnerability advisories are downloaded and sent to the CVE and Bugtraq wrappers respectively. These wrappers parse the downloaded web page, and extract the vulnerable software list. For each software, it is possible to retrieve its producer, its name, and the vulnerable versions. This information is normalized by converting all the letters to lowercase, and by replacing white spaces with underscores. Each wrapper sends the SID and the normalized software list to the *DB interface*. This interface is a Python script that queries the *vulnerable software database* by looking for entries related to the SID received from a wrapper. If there is no such SID in the database, then the new SID and the list of vulnerable software, represented as a CSV string, is inserted into the database. If the database already contains a list of vulnerable software for that SID, and if the new list of vulnerable software contains some new elements, the database entry is updated by adding the new software names to the list.

B. Alert ranking server

The alert ranking server in Figure 2 is a stand-alone application implemented in Python. It listens for requests including two fields: the IP address of a machine and a SID. It then queries the vulnerable software database by using the SID as

[2]other open information sources that are referenced by some rules are arachNIDS, nessus, Microsoft, Application Security Inc. and McAfee Inc.

search key, and the CMDB using the IP address to identify a single machine. If at least one of the two queries fails (i.e. does not return a non-null list of software), a reply is immediately generated, marking the alert as *Inconclusive*. If both the queries are successful, the two lists are compared with the goal of looking for at least one common element. The alert ranking server assumes that the entries in the CMDB and in the vulnerable software database have already been normalized, and the comparison among strings is not case sensitives. If a software exists in both lists, a reply is generated in which the alert is ranked as *Critical*. On the other hand, the alert is ranked as *Not critical*.

C. Local alert manager

Local alert managers are implemented through the Prelude [35] hybrid IDS. Prelude can receive alerts generated by Snort, store them in a local alert database, and relay the received alerts to other instances of Prelude. These features are used to implement the hierarchical architecture shown in Figure 3. The local alert manager is implemented by modifying the Prelude correlator module, that now can invoke a local instance of the alert ranking server when a new alert is received from a NIDS. The rank assigned to the alert is then included in the IDMEF [36] messages used by Prelude to store and relay Snort alerts.

D. Root department manager

Root department managers are implemented through Prelude and integrate a local alert ranking server invoked by the Prelude correlator module. They only receive alerts that have already been ranked by a lower-level local alert aggregator, and leverage the local alert ranking server to repeat the ranking process against the CCMDB, that contains the software configuration of the few machines that are considered critical for the whole organization. Alerts ranked as critical by a root department manager are transmitted to all the other root department managers belonging to the same organization through a custom communication module, that implements the Pastry [37] DHT routing overlay. This component is implemented in Java. It leverages the freepastry [38] libraries to implement the DHT overlay and Scribe [39] to broadcast IDMEF messages to the other root department managers connected to the same DHT overlay. Root department managers are also provided with a custom version of the Prewikka graphical user interface, modified to sort NIDS alerts by their rank and to clearly separate the alerts received from one of the local alert managers to the alerts issued by other root department managers.

E. System validation

To verify the effectiveness of the proposed solution, we tested our prototype in a controlled network environment. Our experimental testbed comprises 30 machines equipped with one Intel Xeon 2.4 GHz CPU, 1 GByte RAM, a 32 bit and 33 MHz PCI bus and a Gbit Ethernet NICs. We emulate a network configuration consisting of three departments, each composed by four network segments. To test the ability to issue early warnings, we simulated three heterogeneous network departments: one populated with Windows-based servers, one with Linux-based servers, and the last one with a mixed Windows and Linux environment. Three machines are configured as root department managers. Each root department manager is the head of a hierarchy of three local alert aggregators. Moreover, four NIDSes are installed within each department. We deploy the external sources crawler and the vulnerable software database on the same machine. Another machine hosts three CMDBs (one for each department) and one CCMDB. The remaining 4 machines of our testbed are used to generate the network traffic analyzed by the NIDSes. We use Tcpreplay to replay the publicly available IDEVAL [40] traffic traces. On top of this background traffic, we added network packets crafted to match specific rules of the Snort signature database. We carried out several experiments for different workload scenarios. Our prototype has always been able to highlight the most dangerous attacks by flagging the related alerts as critical. The graphical user interfaces of the three root department managers denoted *critical* alerts with red flags, *inconclusive* alerts with yellow flags, and *not critical* alerts with green flags.

VII. Conclusion

This paper proposes a distributed architecture suitable for monitoring the IT infrastructure of large organizations consisting of multiple departments, each with multiple network segments. The proposed architecture has three novel features. It combines hierarchical and peer-to-peer communication schemes, thus being suitable to large and complex networks composed by several independent departments, without requiring a centralized authority. It can selectively highlight the few relevant security alerts thanks to an innovative alert ranking scheme that is distributed among several components of the proposed architecture. Finally, it is able to issue early warnings that allow a system administrator to be notified of security threats before vulnerable and critical machines are attacked, thus enabling the deployment of proactive countermeasures. The feasibility of the proposed architecture is verified through the implementation of a prototype based on open source software.

Future works will include the integration of new external information sources. Moreover, we plan to reduce the response time of the alert ranking server by caching previously computed ranks, and by pre-computing off-line the list of machines that are vulnerable to the most common attacks.

Acknowledgment

This research has been partly funded by the EU project CoMiFin (Communication Middleware for Monitoring Financial Critical Infrastructures) IST-225407.

References

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

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

[3] S. Patton, W. Yurcik, and D. Doss, "An achilles' heel in signature-based ids: Squealing false positives in snort," in *Proc. of the International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2001.

[4] S. Snapp, J. Brentano, G. Dias, T. Goan, T. Grance, L. Heberlein, C.-L. Ho, K. Levitt, B. Mukherjee, D. Mansur, K. Pon, and S. Smaha, "A system for distributed intrusion detection," in *Compcon Spring '91. Digest of Papers from the IEEE Computer Society Thirty-sixth International Conference*, Feb 1991.

[5] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C.-L. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur, *Internet besieged: countering cyberspace scofflaws*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., May 1988, ch. DIDS (distributed intrusion detection system)—motivation, architecture, and an early prototype, pp. 211–227.

[6] R. A. Kemmerer, "Nstat: A model-based real-time network intrusion detection system," University of California at Santa Barbara, Santa Barbara, CA, USA, Tech. Rep., 1998.

[7] T. Bass, "Multisensor data fusion for next generation distributed intrusion detection systems," in *Proc. of the 1999 DoD-IRIS National Symposium on Sensor and Data Fusion (NSSDF)*, May 1999.

[8] ——, "Intrusion detection systems and multisensor data fusion," *Communications of the ACM*, vol. 43, no. 4, pp. 99–105, May 2000.

[9] Y.-S. Wu, B. Foo, Y. Mei, and S. Bagchi, "Collaborative intrusion detection system (cids): A framework for accurate and efficient ids," in *Proc. of the 19th Annual Computer Security Applications Conference*, December 2003.

[10] Y. Wang, H. Yang, X. Wang, and R. Zhang, "Distributed intrusion detection system based on data fusion method," in *Proc. of the Fifth World Congress on Intelligent Control and Automation (WCICA 2004)*, June 2004.

[11] Y.-F. Zhang, Z.-Y. Xiong, and X.-Q. Wang, "Distributed intrusion detection based on clustering," in *Proc. of 2005 International Conference on Machine Learning and Cybernetics*, Aug. 2005.

[12] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "Grids - a graph-based intrusion detection system for large networks," in *Proc. of the 19th National Information Systems Security Conference*, October 1996.

[13] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," in *Proc. of the 4th International Symposium on Recent Advances in Intrusion Detection RAID 2001*, October 2001.

[14] Z. Zhang, J. Li, C. N. Manikopulos, J. Jorgenson, and J. Ucles, "A hierarchical anomaly network intrusion detection system using neural network classification," in *Proc. of 2001 WSES Conferemce on Neural Networks and Applications (NNA '01)*, Feb. 2001.

[15] ——, "HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification," in *Proc. of the 2001 IEEE Workshop on Information Assurance and Security*, June 2001.

[16] Y. Chu, J. Li, and Y. Yang, "The architecture of the large-scale distributed intrusion detection system," in *Proc. of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT '05)*, December 2005.

[17] Q. Xue, J. Sun, and Z. Wei, "Tjids: an intrusion detection architecture for distributed network," in *Proc. ot the 2003 IEEE Canadian Conference on Electrical and Computer Engineering CCECE 2003*, May 2003.

[18] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. H. Spafford, and D. Zamboni, "An architecture for intrusion detection using autonomous agents," in *Proc. of the 14th Annual Computer Security Applications Conference (ACSAC 1998)*, December 1998.

[19] M. Colajanni, D. Gozzi, and M. Marchetti, "Collaborative architecture for malware detection and analysis," in *Proc. of the IFIP 23rd International Information Security Conference (SEC '08)*, September 2008.

[20] D. Ragsdale, C. Carver, J. Humphries, and U. Pooch, "Adaptation techniques for intrusion detection and intrusion response systems," in *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics (SMC 2000)*, October 2000.

[21] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "A comprehensive approach to intrusion detection alert correlation," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 3, pp. 146–169, 2004.

[22] C. Kruegel and W. Robertson, "Alert verification: Determining the success of intrusion attempts," in *Proc. of the First Workshop the Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA 2004)*, 2004.

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

[24] G. White, E. Fisch, and U. Pooch, "Cooperating security managers: a peer-based intrusion detection system," *Network, IEEE*, vol. 10, no. 1, pp. 20–23, Jan/Feb 1996.

[25] D. J. Ingram, H. S. Kremer, and N. C. Rowe, "Distributed intrusion detection for computer systems using communicating agents," in *Proc. of 2000 Command and Control Research and Technology Symposium*, June 2000.

[26] R. Janakiraman, M. Waldvogel, and Q. Zhang, "Indra: A peer-to-peer approach to network intrusion detection and prevention," in *Proc. of the 12th IEEE International Workshops on Enabling Technologies (WETICE'03)*, June 2003.

[27] M. Locasto, J. Parekh, A. Keromytis, and S. Stolfo, "Towards collaborative security and p2p intrusion detection," in *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop (IAW '05)*, June 2005.

[28] M. Marchetti, M. Messori, and M. Colajanni, "Peer-to-peer architecture for collaborative intrusion and malware detection on a large scale," in *Proc. of the 12th Information Security Conference (ISC 2009)*, September 2009.

[29] D. J. Malan and M. D. Smith, "Host-based detection of worms through peer-to-peer cooperation," in *Proc. of the 3rd workshop on rapid malcode (WORM'05)*, November 2005.

[30] V. Vlachos, S. Androutsellis-Theotokis, and D. Spinellis, "Security applications of peer-to-peer networks," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 45, no. 2, pp. 195–205, 2004.

[31] Z. Li, Y. Chen, and A. Beach, "Towards scalable and robust distributed intrusion alert fusion with good load balancing," in *Proc. of the 2006 SIGCOMM workshop on Large-scale attack defense (LSAD '06)*, September 2006.

[32] C. Duma, M. Karresand, N. Shahmehri, and G. Caronni, "A trust-aware, p2p-based overlay for intrusion detection," *Proc. of the 17th International Conference on Database and Expert Systems Applications (DEXA'06)*, September 2006.

[33] V. Yegneswaran, P. Barford, and S. Jha, "Global intrusion detection in the domino overlay system," in *In Proc. of the 11th Annual Network and Distributed System Security Symposium (NDSS'04)*, February 2004.

[34] "Snort home page." [Online]. Available: www.snort.org

[35] Prelude IDS technologies, "Prelude ids homepage." [Online]. Available: http://www.prelude-ids.org/

[36] H. Debar, D. Curry, and F. B., "The Intrusion Detection Message Exchange Format," RFC 4765, March 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4765.txt

[37] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware '01)*, November 2001.

[38] "Freepastry home page," available at http://www.freepastry.org/FreePastry/.

[39] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1489–1499, Oct 2002.

[40] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "Analysis and results of the 1999 darpa off-line intrusion detection evaluation," in *Proc. of the Third International Workshop on Recent Advances in Intrusion Detection*, Toulouse, France, October 2000.