

# Modeling Alerts for IDS Correlation

Sebastian Roschke, Feng Cheng, and Christoph Meinel

Hasso-Plattner-Institute (HPI), University of Potsdam

P.O.Box 900460, 14440, Potsdam, Germany

{sebastian.roschke, feng.cheng, meinel}@hpi.uni-potsdam.de

**Abstract:** Intrusion Detection Systems (IDS) have been used widely to detect malicious behavior in network communication and hosts. IDS management is an important capability for distributed IDS solutions, which makes it possible to integrate and handle different types of sensors or collect and synthesize alerts generated from multiple hosts located in the distributed environment. Sophisticated attacks are difficult to detect and it is necessary to integrate multiple data sources for detection and correlation. Attack graph (AG) is used as an effective method to model, analyze, and evaluate the security of complicated computer systems or networks. The attack graph workflow consists of three parts: information gathering, attack graph construction, and visualization. This paper proposes the integration of the AG workflow with an IDS management system to improve the quality of alert correlation. The vulnerability and system information is used to remodel the incoming IDS alerts. The AG is used during the correlation process to filter and optimize correlation results. A prototype is implemented using automatic vulnerability extraction and AG construction based on unified data models.

## I. Introduction

Intrusion Detection Systems (IDS) have been proposed for years as an efficient security measure and are nowadays widely deployed for securing critical IT infrastructures. Based on the protected objective, IDS can be classified into host-based intrusion detection systems (HIDS), network-based intrusion detection systems (NIDS), or distributed intrusion detection systems (DIDS), which contain both types of sensors [5]. Due to different deployment mechanisms, IDS can be categorized as software-based IDS, hardware-based IDS, and Virtual Machine (VM) based IDS [1]. Nowadays, lots of commercial and open source IDS implementations have emerged and been used in practice for identifying malicious behaviors against protected hosts or network environments. Some known examples of existing IDS solutions are F-Secure Linux Security [2], Samhain [3], and Snort [4]. The Intrusion Detection Message Exchange Format (IDMEF) [6] has been proposed as a standard to enable interoperability among different IDS approaches to simultaneously provide multiple benefits from various IDS sensors. Correlation of IDS alerts has been proposed for addressing the problem of false-positive alerts. However, detection of sophisticated attacks remains to be difficult and challenging. In this paper, the utilization of context data sources for IDS correlation is proposed. We focused on the integration of the

Attack Graph (AG) workflow with an IDS management system.

Attack Graphs have been proposed for years as a formal way to simplify the modeling of complex attacking scenarios. Based on the interconnection of single attack steps, they describe multi-step attacks [25]. Attack Graphs not only describe one possible attack, but many potential ways for an attacker to reach a goal. In an attack graph, each node represents a single attack step in a sequence of steps. Each step may require a number of previous attack steps before it can be executed, denoted by incoming edges, and on the other hand may lead to several possible next steps, denoted by outgoing edges. With the help of attack graphs, most of possible ways for an attacker to reach a goal can be computed. This takes the burden from security experts to evaluate hundreds and thousands of possible options. At the same time, representing attack graphs visually allows security personal a faster understanding of the problematic pieces of a network [26, 27].

To perform high quality correlation, the original alerts need to be preprocessed. The information of the target system as well as possible relevant vulnerabilities are usually useful for later correlation. Therefore, we propose a new model of alert, which integrates the system information and vulnerability information which is related to the original alert. The AG workflow is integrated into the correlation process, by which we can easily get: vulnerability information, system information, and the calculated graph. The vulnerability and system information is used to prioritize and tag the incoming IDS alerts. The graph is used during the correlation process to filter incorrect correlation results. The implemented prototype integrates the AG workflow and environment information. An alert modeling for correlation is used, to provide access to environment information and the attack graph in the correlation process.

This paper is organized as follows. Section II describes alert correlation approaches and the attack graph workflow. In Section III, the proposed architecture of the IDS management system is described. Section IV presents the data models for the existing data sources of the attack graph workflow. The way and design to extract information from vulnerability databases is described in Section V. The correlation approaches of the management system using the AG data sources are described in Section VI. Finally, we conclude the paper in Section VIII.

## II. Related Work

### A. Alert Correlation

The alert correlation framework usually consists of several components [9]: *Normalization*, *Aggregation (Clustering)*, *Correlation*, *False Alert Reduction*, *Attack Strategy Analysis*, and *Prioritization*. IDMEF [6] and CVE [10] are important efforts in the field of *Normalization*. Approaches of aggregation are mostly based on similarity of alerts [12, 13] or generalization hierarchies [11]. The correlation [9] can be classified as: *Scenario-based correlation* [14], *Rule-based correlation* [15], *Statistical correlation* [16], and *Temporal correlation* [17]. False alert reduction can be done by using such techniques as data mining [18] or fuzzy techniques [19]. Attack strategy analysis often depends on reasoning and prediction of attacks missed by the IDS [20]. In terms of Prioritization, the alerts are categorized based on their severity, e.g., using attack ranks [21]. Over the last years, many new techniques and methods are applied to improve the quality and performance of alert correlation, e.g., machine learning, data mining [18], or fuzzy techniques [19]. The work described in [22] considers the performance of alert correlation by using memory-based table indexes for hyper alerts. A hyper alert is a cluster of alerts with the same properties, e.g., the same source address and target address. The approach using index tables to enable fast correlation is introduced in [23].

### B. The Attack Graph Workflow

The attack graph workflow consists of three independent phases: Information Gathering, Attack Graph Construction, Visualization, and Analysis. In the information gathering phase, all necessary information to construct attack graphs is collected and unified, such as information on network structure, connected hosts, and running services. In the attack graph construction phase, a graph is computed based on the gathered system information and existing vulnerability descriptions. Finally, the attack graph is created in the visualization and analysis phase. Attack graphs always require a certain set of input information. For one, a database of existing vulnerabilities has to be available, as without it, it would not be possible to identify or evaluate the effects of host-specific weaknesses. Also, the network structure must be known beforehand. It is necessary to identify which hosts can be reached by the attacker. Often, an host-based vulnerability analysis is performed before the attack graph is constructed.

Vulnerability information is stored in so called vulnerability databases (VDB), which collect known software vulnerabilities. Such databases comprehend large compilations of software weaknesses in a non-uniform manner. Well known databases are the VDB from SecurityFocus [32], advisories from Secunia [31], and the Open Source Vulnerability Database (OSVDB) [29], operated by the Open Security Foundation. Besides these known VDB from different providers, there is another important effort called the *Common Vulnerabilities and Exposures* (CVE) [10], which is a meta vulnerability database. Its goal is to provide a common identifier for known weaknesses which can be used across various VDBs. Before 1999, each vulnerability database has

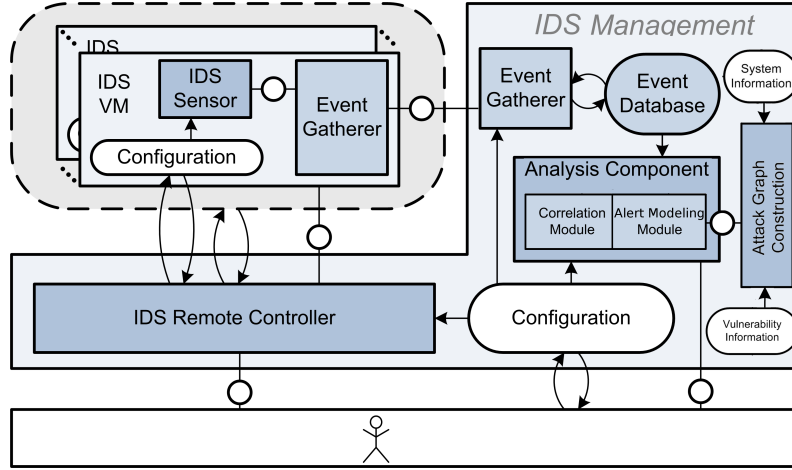
its own name and it was difficult to detect when entries referred to the same weakness. With the help of CVE entries, vulnerabilities at least have a unique identifier. For attack graph construction, up-to-date vulnerability information is crucial to provide high quality results. As vulnerability descriptions are stored in semi-structured textual descriptions, automatic extraction is possible with 70-90 percent of correctness from textual descriptions stored in various VDBs [7].

[28] presents the Multihost, Multistage Vulnerability Analysis (MulVAL), a “logic-programming approach to network security analysis”. In his words, an attack graph is a direct acyclic graph which represents the dependency of actions that lead to the violation of the security property of a network. Attack graphs can be described either based on the network state or on exploit dependencies, whereas the latter is a much more efficient representation. MulVAL is based on exploit dependency graphs and uses Datalog to describe system properties. Datalog is a subset of the Prolog programming language. All relevant input data such as software vulnerability advisories, configuration and network topology information, and security policies are described in the same language, i.e., Datalog. This input is then evaluated with the help of XSB<sup>1</sup> as the logic engine to evaluate predicates. Based on the predicates, XSB will compute all possible paths to complete a defined goal, for example execution rights for an attacker on a specific host. [28] presents about 20 rules which specify exploits such as code execution, file access, and privilege escalation, all of them hand-crafted. Each rule include an exploit range, that is whether an exploit can be used only locally from a remote host. Also, exploit consequences are described, e.g., a denial of service. The rules are specific to the Unix operating system. [28] argues that a “model as needed” approach should be followed which means that aspects of system should only be modeled if they are relevant to determine pre- and postconditions of an attack. To extract the semantics of an attack (e.g., pre- and postconditions and the exploit range), the National Vulnerability Database (NVD)[33] is used, existing vulnerabilities are identified with the help of OVAL[30]. In this paper, the data model described in [7] is used to store vulnerability and system information for IDS correlation. The third party tool MulVAL [28] is integrated and the created AG is used during the correlation process.

## III. IDS Management Architecture Integrating AG Workflow

The proposed architecture of the IDS management system is shown in Figure 1. It includes several *IDS VMs* and a *IDS VM Management Unit*. The *IDS VM Management Unit* consists of the following active components: the *Event Gatherer*, the *Event Database*, the *Analysis Component*, and the *IDS Remote Controller*, the *Attack Graph Construction*, the *System Information*, and the *Vulnerability Information*. The *Event Database* is a passive storage that holds information on all received events. It can be accessed through the *Analysis Component*. User controls the IDS management through direct interaction and configuration of the core components. The

<sup>1</sup><http://xsb.sourceforge.net/>



**Figure. 1:** An IDS Management Architecture integrating AG workflow

*IDS Sensors* on the VMs are responsible for detecting and reporting malicious behavior. Each sensor is connected to the *Event Gatherer* component to transmit triggered events. A sensor, which could be a running IDS sensor with all its signatures and configurations, can be configured through the *IDS Remote Controller*.

The IDS sensor identifies malicious behavior and generates alerts through a reporting component, which will be processed by the *Event Gatherer*. The sensor is an independent process, which can be any NIDS or HIDS, e.g., Snort or Samhain. The *Event Gatherer* is responsible for collecting all events from *IDS Sensors*. As shown in Figure 1, the *Event Gatherer* component is introduced on the *IDS Sensor* side as well. This gatherer is used to standardize the outputs from different sensors as well as realize the logical communication, such as file-based or network-based, between the sensor and the management unit. The gatherer consists of several Plugins: *Senders*, *Receivers*, and *Handlers*. *Receivers* are used to read alerts and convert them to IDMEF. *Senders* are used to write alerts to a destination, e.g., a network, a database, or a folder. *Handlers* can be used to modify alerts in processing, e.g., to log each alert from a specific sensor. Each event is made persistent in the *Event Database* storage. The gatherer can be configured by the user and is connected to each sensor it receives events from. A gatherer can be the running instance of an IDS management component that accepts connections and writes events to the database.

The *Attack Graph Construction* module integrates the AG workflow with the system. It provides an interface to the AG construction engine for constructing and re-creating an attack graph at runtime. Furthermore, it provides access to the environment information, e.g., network structure, hosts and its services, as well as existing vulnerabilities. The interface to the correlation process is the *Attack Graph Module*, which caches correlation results and triggers needed changes to the attack graph. The *Analysis Component* consists of the *Correlation Module* and the *Alert Modeling Module*. The *Correlation Module* is responsible for running different correlation algorithms on the *Event Database* using the available data sources: the *Vulnerability Information* and the *System Information*. The *Attack Graph Module* is responsible for connecting the attack graph workflow to the *Correlation Mod-*

*ule*. It handles the integration of the two main data sources and the *Attack Graph Construction* component. In this way, it is possible to integrate the data sources as well as the created attack graph itself by triggering the component.

#### IV. System Information and Vulnerability Information

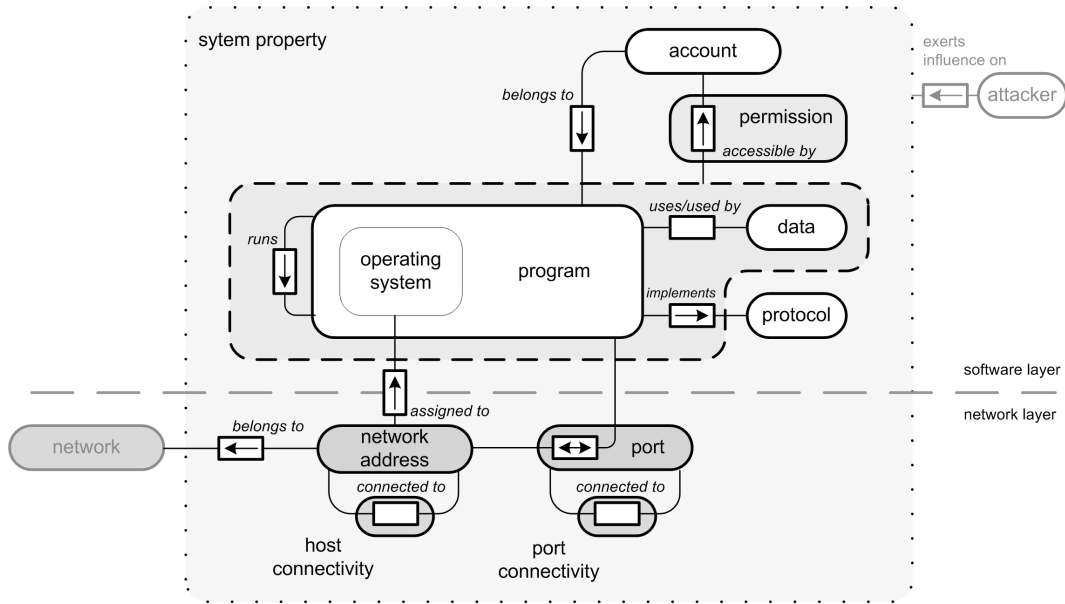
##### A. A Data Model for System Descriptions

Figure 2 shows the so-called *System properties* used to describe systems and networks.

*System properties* are characteristics and resources of a computer system. Each system property describes one specific attribute of such a system, whereas properties are related to one another as depict in Figure 2. For example, the installed version of an application can be a system property. An application's version is meaningless if it cannot be linked to a certain application. Properties and their relations may change over time due to modifications, such that an application may be upgraded to a newer version. System properties can be found in two layers, the network layer and the software layer. The network layer describes properties of interconnected computers, such as network addresses and port numbers. The software layer describes properties of software systems, such as programs, data, and account information.

A *network* is a group of directly connected network addresses. A *network address* is an identifier of a host in a network. Directly connected means it is possible to reach from one host of network another host of the same network. Network addresses may have a number of open ports per address which are used by programs to communicate with other programs.

Also covered are host as well as port connectivity, both are essential to capture which hosts and programs can be reached. *Host connectivity* is a boolean value to describe whether one host can be reached from another host. This may be influenced by the network the corresponding hosts are in or by firewall rules, preventing certain hosts to connect to others. *Port connectivity* is a boolean value to describe whether one port of a network address can be accessed from another port of a network address. Similar to host connectivity, this can be influenced by firewall rules or comparable



**Figure. 2:** System Properties

system configuration tools.

### B. Using available Vulnerability Databases

In [7], existing vulnerability databases are analyzed concerning their usability in attack graph construction. The 10 most popular VDB providers were selected as the base for this evaluation. Most valuable attributes of vulnerability entries in this process include CVE identifiers, the impact of a vulnerability, the range from which an attack can be conducted, and the required or affected programs. The *Open Vulnerability and Assessment Language* (OVAL) [30] provides a framework to describe exploitable software configurations affected by a vulnerability. Similar to the *Common Vulnerability Scoring System* (CVSS) [34], OVAL is standardized and used by several organizations. In [7], only vulnerability definitions are considered. Based on XML, such definitions consist of meta-data and criteria elements, whereas criteria elements are recursive and therefore allow configuration specifications at an arbitrary level of detail. Because important attributes, such as the attack range and the impact, are often described with a selection of English words, the interpretation of textual descriptions cannot be neglected. Not all information is available in CVSS format and OVAL definitions also rely on the use of English phrases. Nevertheless, it has been demonstrated that verbalization is often semi-formal and therefore easy to parse. The approach is analyzed in term of correctness using the attributes of range from which an attack can take place as well as which of the three security goals confidentiality, integrity, and availability can be violated by exploiting a vulnerability. The range information can be identified correctly in more than 90 percent of the cases, confidentiality violations in almost 82 percent of the cases, integrity violations in more than 85 percent, and availability violations in almost 75 percent of the analyzed descriptions.

### C. A Data Model for Vulnerability Descriptions

To use vulnerability descriptions from different databases in attack graph construction, these descriptions need to be unified. We used a flexible and extensible data model to unify vulnerability descriptions of multiple vulnerability databases. As described in [8], the data model is capable to express vulnerability descriptions provided by vulnerability databases. The logical data model describes *system*, *influence*, and *range properties*. *System properties* describe states a system can be in, e.g., running programs, existing accounts, and existing databases. *Influence properties* describe the influence an attacker has on system properties by successful exploitation. *Range properties* describe the location from which an attacker can perform successful exploitation, e.g., local or remote. A vulnerability requires a precondition and a postcondition, which can be represented by system properties. Two basic types are used for descriptions: *properties* and *sets*. *Properties* represent predicates and *sets* allow a grouping of properties based on boolean logic. Both types facilitate a simple evaluation based on matching of True or False values. Finally, descriptions link different system states together, one as the requirement and the other as the result of an attack. Based on this properties and sets, we can flexibly describe many different system states.

*System properties* are characteristics and resources of a computer system which are considered relevant vulnerability information. Each system property describes one specific attribute of such a system, whereas properties are related to one another. For example, the installed version of an application can be a system property. An application's version is meaningless if it cannot be linked to a certain application. Properties and their relations may change over time due to modifications, such that an application may be upgraded to a newer version. *System properties* can be found in two layers, the network layer and the software layer. The network layer describes properties of interconnected computers, such as network addresses and port numbers. The software layer describes properties of software systems, such as programs,

data, and account information. We defined several different *system properties* which are useful to create attack graphs, such as network properties, host connectivity, programs, protocols, data, accounts, and others. To describe actions performed on systems, *influence properties* will be used. *Influence properties* describe the relationship between a potential attacker and system properties which represent computer resources.

## V. Extraction of Vulnerability Information

A prototype for automatic extraction of vulnerability descriptions from vulnerability databases is used as described in [7]. The prototype will use a designed data structure as an exchange format between components which extract information from various VDBs as well as components which output information for attack graph tools and related applications. The prototype is based on plugins: so called readers and writers. In the following, the extracting components will be referred to as readers, because they read information from a vulnerability database or some other source. Every reader is able to extract information from a specific data source. For example, an NVD reader is able to filter relevant attack information from the National Vulnerability Database (NVD) [33]. The counterpart of readers are writers, which output vulnerability information in different formats. Gathered data can be read by various source, e.g., attack graph tools or vulnerability analysis programs. Thus, it is reasonable to provide a writer for each target application.

Readers such as the NVD Reader or the OVAL Reader transform information from one XML representation into another XML representation, but the transformed information remains the same. The major benefit of this type of readers is the increased amount of available vulnerability information provided by a common vulnerability database which is based on the data structure used in the implementation. The CVE Reader on the other hand extracts information from textual descriptions of vulnerabilities. To be able to evaluate how much of the encoded information can be retrieved, it is useful to have a closer look at the extracted information. For this, the retrieved data will be compared to the data which is available in the form of CVSS entries. Those CVSS entries provide range and impact information of vulnerabilities in a standardized format. The NVD contains both, textual descriptions as well as CVSS values for all entries. Both information sets should contain the same data, therefore the comparison is based on these two sets. Note that this evaluation aims not at the evaluation of vulnerabilities itself, but rather at an analysis of how much of the information encoded in textual descriptions can be extracted correctly.

## VI. Remodeling IDS Alerts

The AG workflow involves three data sources that can be utilized for IDS correlation: the system information, the vulnerability information, and the generated attack graph. There are several useful parts of the system information that can be used in the correlation process. First, we are using host connectivity information to find attacks that are based on spoofed packets. If an alert shows a *SrcIP-DstIP* pair and the hosts have no connectivity, the alert is caused by a spoofed packet.

That prevents the suspicion of the wrong person or host. Information on running OS and programs of a target host are used to filter out alerts for less dangerous attacks and to set high priority for very dangerous attacks. This can be useful in case of many alerts for an attack to a Linux OS based host when we know that a Windows OS is running on that host, i.e., the attack is less dangerous as it is unlikely that it leads to critical damage. Contrary, an alert for an attack on a Windows host that runs Windows OS is critical. Account data is used to identify accounts and persons for target and source hosts of an attack. The target account is identified to inform the responsible persons that their system is under attack. The source account is identified to either track the attacker or inform the responsible persons of the attacking host that their system is used to attack hosts in the network and might have been compromised in the past.

We are using vulnerability information for remodeling the alerts by enriching each alert with tagging information and priorities. CVSS information is used to define the priority of an IDS alert which is created for an attack exploiting the specific vulnerability. We are using the *Base Score* of CVSS and tag each alert that can be assigned to a CVE with the specific value. During the correlation, the system can be configured to ignore scores below 5.0. On the frontend the system can do ranking and filtering according to CVSS scores to help the user with manual analysis. Additionally, the system shows possible vulnerability information for generic alerts. If an alert announces shellcode detection in a communication between host *A* and host *B* on port 445 or 139, the system lists vulnerabilities for all SMB vulnerabilities (matching the host OS and running programs if required). The system can also order alerts due to the publication date of the related vulnerability, e.g., showing alerts for more recent vulnerabilities first before others.

The utilization of attack graph involves another type of graph related to IDS: the *Scenario Graph* ([15], [36]). A scenario graph represents a way of a recognized attack path through the network. The system uses the attack graph to match the scenario graph and identify subpaths in the attack graph. By specifying important hosts, the system can generate new correlation alerts if the attacker covered 70 – 80% of a known attack path. It is also possible that the scenario graph reveals a new way an attacker walked through the network. In this case the AG is updated. If a host is part of the scenario graph, an actual attack is going on. IDS alerts that have such a host as source IP are ranked with highest priority. In this way, the network administrator can observe ongoing attacks and take precautions using the attack graph showing possible next steps of the attacker.

Apart from the introduced interactions between the AG workflow and the IDS correlation and management, there might be lots of other possibilities for interaction. The introduced methods are implemented in our Advanced IDS Management Architecture [24]. The system uses a plugin concept for many parts and is implemented in Java. It provides connectors for popular IDS sensors (e.g., Snort [4], Samhain [3]) and for other IDS management systems (e.g., Prelude [5]). The system uses multiple alert storages (In-Memory DBS, column-based and row-based DBS) and has a plugin engine for correlation modules. The frontend is implemented using

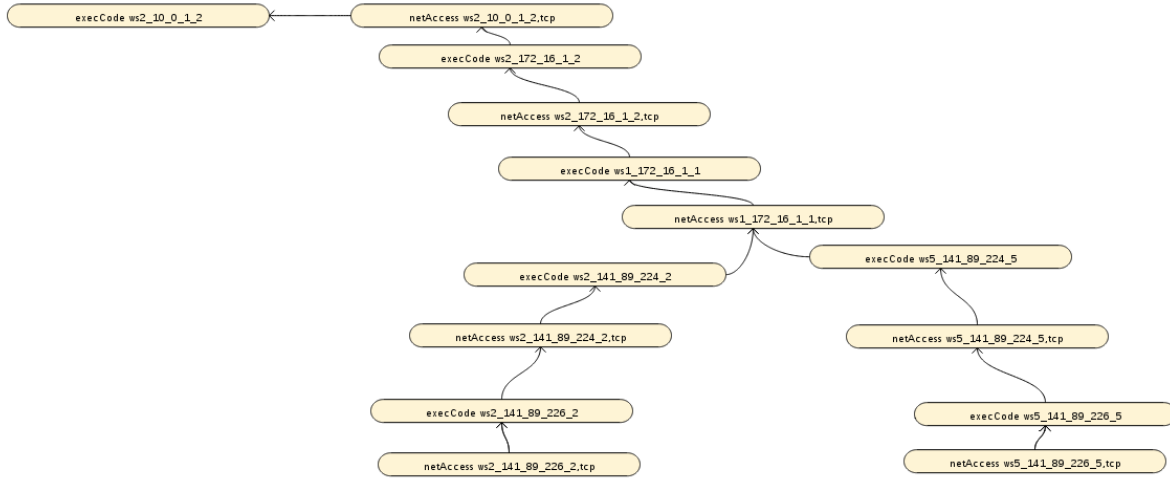


Figure. 3: IDS AG Platform Screenshot

Java servlets.

## VII. Integrating MulVAL with the Platform

The Multi-host, Multi-stage Vulnerability Analysis Language Tool (MulVAL) is a research tool developed in [28]. In contrast to other tools, the MulVAL application is based on a logical programming approach. The required input data includes security advisories, the network configuration, as well as the machine configuration of participating hosts. MulVAL has been tested with a network of up to 1,000 machines with up to 100 vulnerabilities.

To integrate the MulVAL tool, the operating data has to be imported into the platform. MulVAL is using multiple data inputs and one basic data output. The data inputs are describing the network information as well as the integrated vulnerability information. The data inputs are formatted in predicates. There are different predicates for different kinds of information. Vulnerability information is presented by the predicate *vulProperty(val1, val2, val3)*. The predicate takes three values: *val1* provides the CVE identifier, *val2* provides the range of the vulnerability, e.g., local or remote, and *val3* provides the type, e.g., Denial-of-Service, privilege escalation, etc. Network information is stored with the predicates *haci(val1, val2, ...)*, *NetworkServiceInfo(val1, val2, val3, val4, val5)*, and *vulExists(val1, val2, val3)*. The predicate *haci()* defines connectivity between hosts and networks. It can also be used to connect subnetworks and to define hosts. The predicate *NetworkServiceInfo()* is used to describe services running on hosts. It uses *val1* to specify the host, *val2* to specify the software, *val3* to define the transport protocol (either tcp or udp), *val4* to specify the port, and *val5* to define the user executing the software. The predicate *vulExists()* defines if there are vulnerabilities present on the different hosts. It uses *val1* to define the host, *val2* to provide

the CVE identifier, and *val3* to specify the software.

Apart from the visual graph, the MulVAL reasoning engine provides a temporary file with graph data which is used abreast for graph visualization. The *Attack Graph Construction* component uses this file to visualize and represent the created attack graph. For correct representation of the graph, the platform needs to read the MulVAL input files as the graph file does not provide enough information for IDS correlation. The graph file provides information on all existing hosts that can be used to compromise a network, but not on the specific vulnerabilities used to compromise one host. This information is of special interest for IDS correlation. The prototype of the platform and its AG visualization is shown in Figure 3.

## VIII. Conclusion

A promising future task is to find and connect more applicable data sources to the system, e.g., historic user and system data can be used for forensics and correlation of IDS alerts over a long period of time. The system needs extensive performance tests and scalability tests, as the current testing is using a dataset of 1.3 million alerts generated from one Snort sensor. The attack graph is created on a relatively small network of 10 hosts. The system shows sufficient performance with this network configuration, but it needs to be evaluated based on large networks. Usability tests of the network administrators using this platform need to be conducted in the future to prove that the system and its algorithms improve their workflow. The extraction of information from exploit databases is also considered as interesting research topic and valuable source of information for the IDS and correlation process.

In this paper, we propose a new model of IDS Alert by integrating the AG workflow with an IDS management system to

improve correlation quality. The approach uses the information sources of the AG workflow: vulnerability information, system information, and the calculated graph. The vulnerability and system information is used to prioritize and tag the incoming IDS alerts. The AG is used during the correlation process to filter incorrect correlation results. An architecture is described consisting of an *Event Gatherer*, a *Correlation*, an *Attack Graph Construction* module, and a *Frontend* for the user. The *Correlation Engine* works based on pluggable *Correlation Modules* and uses the *Alert Storage*, the *Vulnerability Information* and *System Information* as input. The *Frontend* works on alert information which is already tagged and filtered based on the *Vulnerability Information* and *System Information*. A prototype is implemented using unified data models for system information and vulnerability information.

## References

- [1] Laureano, M., Maziero, C., Jamhour, E.: Protecting host-based intrusion detectors through virtual machines. *Computer Networks*, Vol. 51, Issue 5, pp. 1275-1283 (2007).
- [2] F-Secure Linux Security: <http://www.f-secure.com/linux-weblog/> (accessed Dec 2010).
- [3] Samhain IDS: WEBSITE: <http://www.la-samhna.de/samhain/> (accessed Dec 2010).
- [4] Snort IDS: WEBSITE: <http://www.snort.org/> (accessed Dec 2010).
- [5] Prelude IDS: WEBSITE: <http://www.prelude-ids.com/> (accessed Dec 2010).
- [6] Debar, H., Curry, D., Feinstein, B.: *The Intrusion Detection Message Exchange Format, Internet Draft*, Technical Report, IETF Intrusion Detection Exchange Format Working Group (July 2004).
- [7] Roschke, S., Cheng, F., Schuppenies, R., and Meinel, Ch.: "Towards Unifying Vulnerability Information for Attack Graph Construction", In: *Proceedings of the 12th Information Security Conference (ISC'09)*, Springer LNCS, vol. 5735, pp. 218-233, Pisa, Italy (Sep 2009).
- [8] Cheng, F., Roschke, S., Schuppenies, R., and Meinel, Ch.: "Remodeling Vulnerability Information", In: *Proceedings of the 5th Inscrypt Conference (Inscrypt'09)*, Springer LNCS 6151, Beijing, China, pp.324-336 (Dec 2009).
- [9] Sadoddin, R., Ghorbani, A.: *Alert Correlation Survey: Framework and Techniques*, In: *Proceedings of the International Conference on Privacy, Security and Trust (PST'06)*, ACM Press, Markham, Ontario, Canada, pp. 1-10 (Oct 2006).
- [10] Mitre Corporation: *Common Vulnerabilities and Exposures (CVE)*, Website: <http://cve.mitre.org/> (accessed Dec 2010).
- [11] Julisch, K.: *Clustering intrusion detection alarms to support root cause analysis*, In: *ACM Transactions on Information and System Security*, vol. 6, Issue 4, pp. 443-471 (2003).
- [12] Cuppens, F.: *Managing alerts in a multi-intrusion detection environment*, In: *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC'01)*, IEEE Press, New-Orleans, USA, pp. 0-22 (Dec 2001).
- [13] Valdes, A., and Skinner, K.: *Probabilistic alert correlation*, In: *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID'01)*, CA, USA, Springer LNCS 2212, pp.54-68 (Oct 2001).
- [14] Debar, H., and Wespi, A.: *Aggregation and correlation of intrusion-detection alerts*, In: *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID'01)*, CA, USA, Springer LNCS 2212, pp. 85-103 (Oct 2001).
- [15] Ning, P., Cui, Y., and Reeves, D.: *Constructing attack scenarios through correlation of intrusion alerts*, In: *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*, ACM Press, Washington, DC, USA, pp. 245-254 (2002).
- [16] Qin, X.: *A Probabilistic-Based Framework for INFOSEC Alert Correlation*, PhD thesis, Georgia Institute of Technology (2005).
- [17] Qin, X.: *Statistical causality analysis of infosec alert data*, In: *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID'03)*, PA, USA, Springer LNCS 2820, pp. 73-93 (Sep 2003).
- [18] Manganaris, S., Christensen, M., Zerkle, D., and Hermiz, K.: *A data mining analysis of rtid alarms*, In: *Computer Networks*, vol. 34, Issue 4, pp. 571-577 (2000).
- [19] Siraj, A., and Vaughn, R. B.: *A cognitive model for alert correlation in a distributed environment*, In: *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI'05)*, IEEE Press, Atlanta, GA, USA, pp. 218-230 (May 2005).
- [20] Ning, P., Xu, D., Healey, C. G., and Amant, R. S.: *Building attack scenarios through integration of complementary alert correlation method*, In: *Proceedings of the Network and Distributed System Security Symposium (NDSS'04)*, The Internet Society, San Diego, California, USA (2004).
- [21] Porras, P., Fong, M., and Valdes, A.: *A mission-impact-based approach to infosec alarm correlation*, In: *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID'02)*, Zurich, Switzerland, Springer LNCS, pp. 95-114 (2002).



- [22] Tedesco, G. and Aickelin, U.: *Real-Time Alert Correlation with Type Graphs*, In: *Proceedings of the 4th international Conference on Information Systems Security (ISS'09)*, Springer LNCS 5352, Hyderabad, India, pp. 173-187 (2008).
- [23] Ning, P. and Xu, D.: *Adapting Query Optimization Techniques for Efficient Intrusion Alert Correlation*, Technical Report, North Carolina State University at Raleigh (2002).
- [24] Roschke, S., Cheng, F., and Meinel, Ch.: *An Advanced IDS Management Architecture*, In: *Journal of Information Assurance and Security*, Dynamic Publishers Inc., vol. 51, Atlanta, GA 30362, USA, ISSN 1554-1010, pp. 246-255 (Jan 2010).
- [25] Schneier, B.: *Attack Trees: Modeling Security Threats*, In: *Journal Dr. Dobbs' Journal*, online available from <http://www.ddj.com/architect/184411129> (Dec 1999)
- [26] Sheyner, O., Haines, J., Jha, S., Lippmann, R., and Wing, J. M.: *Automated Generation and Analysis of Attack Graphs*, In: *Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P'2002)*, IEEE Press, Washington DC, USA, pp. 273-284 (May 2002)
- [27] Noel, S., and Jajodia, S.: *Managing attack graph complexity through visual hierarchical aggregation*, In: *Proceedings of the Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC 2004)*, ACM, Washington DC, USA, pp. 109-118 (Oct 2004)
- [28] Ou, X., Govindavajhala, S., and Appel, A.: *MULVAL: A Logic-based Network Security Analyzer*, In: *Proceedings of the 14th USENIX Security Symposium*, USENIX Association, Baltimore, MD, pp. 8 (Aug 2005).
- [29] OSV Database: *Open source vulnerability database*, Website: <http://osvdb.org/> (accessed Mar 2010).
- [30] Mitre Corporation: *Open Vulnerability and Assessment Language (OVAL)*, Website: <http://oval.mitre.org/> (accessed Mar 2010).
- [31] Secunia: *Secunia Advisories*, Website: <http://secunia.com/advisories/> (accessed Mar 2010).
- [32] SecurityFocus: *Security Focus Bugtraq*, Website: <http://www.securityfocus.com/> (accessed Mar 2010).
- [33] NIST: *National Vulnerability Database (NVD)*, NVD Website: <http://nvd.nist.gov/> (accessed Mar 2010).
- [35] Franqueira, V., and van Keulen, M.: *Analysis of the NIST database towards the composition of vulnerabilities in attack scenarios*, Technical Report, TR-CTIT-08-08, University of Twente, Enschede, February 2008.
- [34] Mell, P., Scarfone, K., and Romanosky, S.: *A complete guide to the common vulnerability scoring system version 2.0*, Website: <http://www.first.org/cvss/> (accessed Mar 2010).
- [36] Hughes, T., Sheyner, O.: *Attack scenario graphs for computer network threat analysis and prediction*, In: *Journal of Complexity*, Wiley Periodicals, Inc., Vol. 9, Issue 2, pp. 15-18 (2004).

## Author Biographies

**Sebastian Roschke** studied IT-System-Engineering from 2003 to 2009 at the Hasso-Plattner-Institute (HPI), University of Potsdam. He got his B.Sc. degree in 2007 and his M.Sc. degree in 2009. He is now working as a PhD student at the chair of Prof. Dr. Christoph Meinel. His research interest focuses on Intrusion Detection, Attack Graph, Virtualization, and practical attack techniques.

**Feng Cheng** is a research associate at Hasso-Plattner-Institute (HPI) at University of Potsdam, Germany. In 1998, he got B.Eng. degree from Beijing University of Aeronautics and Astronautics, China. In 2002, he got M.Eng. degree on Computer Science at Beijing University of Technology, China. There he has done many research works on image processing, JPEG2000 and digital watermarking. From October 2002 to April 2005, he worked as a research assistant at University of Trier. Since April 2005, he started to work at Hasso-Plattner-Institute in Potsdam, Germany. His current research interests include network security, firewalls, secure data transmission, and physical separation technology.

**Christoph Meinel** studied from 1974-79 Mathematics and Computer Sciences at the Humboldt-University in Berlin. He received his PhD degree in 1981. From 1981-1991 he worked at the Department of Mathematics at the Humboldt University and at the Institute of Mathematics of the Academy of Sciences in Berlin as a scientific co-worker. 1988 he received his habilitation degree with a thesis about complexity theory that was published in the series of the Springer Lecture Notes (Vol. 370). After a research stay at the University of Saarbruecken and a visiting position at the University of Paderborn from 1992 to 2004 he worked as a full professor (C4) for computer science at the university of Trier. He received various offers to become a full professor at universities in Germany, Austria, and Norway. 2004 he accepted the offer to become the director of the Hasso-Plattner-Institute (HPI) and professor at the University of Potsdam.