Toward a novel classification-based attack detection and response architecture

Samih Souissi

Toward a Novel Classification-based Attack Detection and Response Architecture

Samih Souissi
INFRES Department
Telecom ParisTech
Paris, France
samih.souissi@telecom-paristech.fr

Abstract— **Attacks on information systems have increased tremendously and have become more diverse and complex. Evolving in an unpredictable manner and having devastating outcomes, the detection and the selection of appropriate countermeasures has become a priority for security analysts. This paper introduces a classification-based Attack Detection system which provides a framework to evaluate, identify, classify and defend against sophisticated attacks. Our approach helps simplify complex rules' expression and alert handling, thanks to a modular architecture and an intuitive rules defining with a high power of expression language. The proposed system is flexible and takes into account several attack properties in order to simplify attack handling and aggregate defense mechanisms.**

Keywords- Attack detection and response, Attack classification, Modular Architecture, Intrusion Detection and Prevention Systems, Firewall, Web Application Firewall, Detection Rules

I. INTRODUCTION

Attacks against computer systems, networks and distributed applications have become more sophisticated and diverse. Security has emerged as a common problem and has gained more interest. Thus, detection and mitigation of attacks are a priority in order to avoid disastrous outcomes. To deal with the growing complexity of new attacks, several solutions such as intrusion detection and prevention systems (IDS/IPS) and web application firewalls (WAF) have been proposed. They play an important role in countering security threats.

However, these systems are not always able to detect attacks, can lead to false positives or false negatives and can consume a high computing power in order to have high performances. Indeed, in signature-based case, these solutions tend to use static rules and to detect only specific attacks or anomalous behaviors that are already known. In anomaly-based case, they need learning process and detection is more complex. In addition, attack detection techniques are far from satisfactory [1]. In fact, solutions like IDSs provide unmanageable amount of alarms to security administrators and these thousands of alarms by day are hard to inspect especially if the majority of them are false positives. Besides, many detection systems do not offer an appropriate compromise between acceptable performance and detection language simplicity. How can a detection system cope with challenges while providing an easy to use and interactive platform?

In our context, attack modeling is crucial to the detection process, as it is closely related to the choice of implemented rules and attack detection parameters. This was the topic of our previous work [2]. The current paper focuses only on the architectural aspects showing how to bring a level of abstraction to make the detection of complex attacks more feasible and the detection rules and security policy defining process simpler. Moreover, our system helps improve the expressiveness of alerts as our system provides easy to understand and not numerous Meta-alerts. Besides, our proposition offers a better understanding of attacks and a decision-making tool to enhance security and increase systems' robustness. This solution is based on a generic model covering the technical and the organizational aspect of attacks. Even though, our proposal system has several applications, we focus in this paper on its added value in a heterogeneous environment correlation and aggregation purposes.

This paper is organized as follows. Section II details the related work of attack detection solutions. In section III, We present our proposition describing the architecture and briefly the language. We expose, in section IV, an example of implementation is given as proof of concept. Finally, section V presents the conclusion and perspectives.

II. RELATED WORK

Over the last decade, many solutions and mechanisms have been proposed to detect computer and network attacks. Snort [3], one of the most widespread IDS, uses a signatures ruleset. Packets are captured, decoded and diagnosed within a preprocessor. Then detection occurs according to the predefined rules to generate events and report by various means. Snort deployment is easy and it has already existing rich rules database. However, it may not be adapted to detect complex attack or to allow mitigation scenarios defining. Unlike Snort, Bro [4] implements a scripting environment. This IDS is highly customizable, with a powerful scripting language simple configuration. However, it does not provide a well-documented ruleset. Besides, these solutions are better in detecting attack on a packet level.

For deeper applicative level detection WAF are often used. ModSecurity [5] is a signature-based attack detection solution and has relatively good performances. Though, this system is strongly related to some types of web servers and it only analyses POST queries to avoid performance deterioration. In addition, the rules' defining is very complex, needing a high expertise in HTTP protocol and regular expressions. Naxsi [6] uses a heuristic approach for the detection of XSS and SQL injection attacks. Its performances are acceptable but require a learning process to define whitelists. Defined rules are static and limited to the context of injection attacks using a cumulative scoring system. These systems do not offer a compromise between acceptable performance and simplicity.

Simmons et al. [7] present a cyber-attack taxonomy called AVOIDIT used to identify and characterize attack. Using attack components, a set of metrics are defined and used by an attack defense performance taxonomy (ADAPT system [8]). This system is game model-based. ADAPT allows classifying

and detecting blended attacks. It helps make an intelligent decision when defending against attacks. However, the taxonomy lacks defense strategies, it is not applicative attacks oriented and it relies on a game decision system that the user is not necessarily able to modify or to define. In [9], Wu et al. propose an attack classification for automatic response systems. Based on this 3 dimensions response-oriented classification (Source: attack origin, Technique: method used by the attacker, Result: outcome of the attack), a correspondence matrix for every attack technique is defined taking into account different sources and results as matrix parameters to define automatic defense techniques. This approach is interesting as the classification helps describe the attack and allows defense mechanisms aggregation. However, types of target is not taken into account. Besides, blended and complex attacks are difficult to classify and thus to counter.

Researchers have done promising works. Nevertheless, no model is widely accepted. As mentioned above, many challenges need to be faced to have a complete, expressive, easy-to-use and manage complex attacks detection system.

III. PROPOSAL

The challenge is to guarantee a good detection of attacks while providing modularity and rule writing simplicity to detect complex attacks and respond automatically according to a user defined security policy. In this section, we present AIDD system (Attack Identification detection and description). This solution should satisfy a set of criteria that will be mentioned at first. Then, we describe our proposal that is composed of: a functional part and a communication part.

A. *AIDD criteria*

In our architecture, a module is an element of the system that performs a predefined function and is able to communicate with other modules. These modules are reusable and interconnected to create a system global function. Our modules and solution should satisfy different criteria:

- Flexibility: Our system is independent of the runtime environment, topology and security devices used. It can be reused though a period of adaptation is needed.
- Expressiveness: The used language guarantees a high power of expression for describing attacks, writing commands or rules to help non-security experts.
- Availability: Working as security monitor, AIDD still sees the whole picture even when facing DOS Attacks.
- Extensibility: User can define its own module to upgrade the system services. He can also update detection rules, attack scenarios and security policy.
- Multi-criteria: AIDD can adapt to security tools from different constructors (open source or not).

B. *AIDD Architecture*

The attack detection and response system, shown in Fig.1, is responsible of flow analysis, attack detection and response. It is composed of the following modules:

- **Dissection Module:** Input (logs/session/event/alert) is transformed, normalized and dissected according to a user defined configuration. A hook system (a hook is an event that will trigger a rule) is closely related to the dissection mechanism. Indeed, hooks are placed and appropriate rules (rule schemes) are associated to evaluate security rules for each dissected field.
- **Analysis Module:** Input can be a dissected network traffic, system/applicative logs or alert. The attack signature or the malicious behavior is described within the detection rules. Seen from another angle, these rules can be considered as a signature database. The detection engine that is used is IDS/IPS/WAF-like system. The analysis can be based on one or many events coming from one or many probes. The analysis can be either offline (log file) or continuous (events, traffic, etc.). This analysis raises an alert or reacts to eventual attack detection. This module is associated to a cache to store appropriate queries for further analysis and a scoring system to indicate suspicious queries.

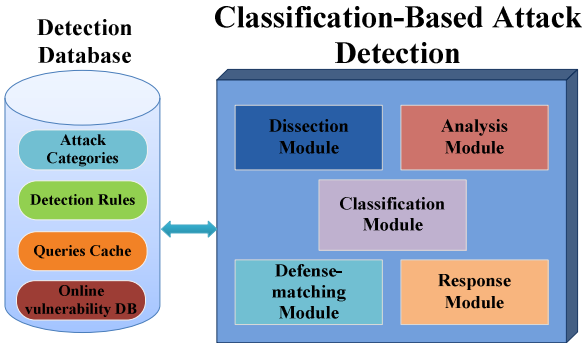

Fig. 1. AIDD Architecture

- **Classification Module:** The originality of our work consists on adding classification to detection. Detection is no longer Attack-centric but based on attack categories having generic patterns or behavior for each class. This classification will help detect attacks whose signatures are not available but whose behavior or related collected data allow classifying it into a certain category of attack. Information needed to classify the attack are: source, target, vector and result of the attack. This approach allows to aggregate defense mechanisms. If given events or alerts from the same or different sources, it will match them with predefined attack scenarios appropriate to the deployment context so that the system is able to respond to complex attacks.
- **Defense matching Module:** This module matches each attack category with appropriate defense mechanisms which are classified into different categories (detection, prevention, response, tolerance). They help the detection system be reactive (define suitable responses for the detected attack class) and proactive (rules update).
- **Response Module:** According to the defense matching different reactions to attacks can be defined. A reaction can be responsive (mitigation/remediation), passive (tolerance) or informative (alert/log/ awareness). After response, data (events/alerts) can be resent to analysis module for further review.

Finally, we define a Detection Database that contains all the information needed: attack classification scheme, detection rules, Attack scenarios and queries. In fact, we propose a generic approach to define Attack categories based on our attack classification [2]. These categories will be the base of our detection process. They can be updated by the user. Orchestration rules are predefined and assigned to specific queries. In addition, there is a cache which is used to store queries/alerts that are assigned scores to have an overview on information exchange and help the correlation process. Our

system is able to get updated information by accessing online vulnerability databases such as OSVDB [10], CVE [11], etc.

C. AIDD Language

In our detection architecture, the communication between the different modules and within each one is handled by a composed language. This language helps modularity to security controls. Given the complexity of the existing languages, our original idea is to use two languages:

- **Basic language (atomic language):** contains single action rules. Different rule types can be found: Action, Alert, Comparison, Detection, Log, Transformation and Normalization rules.
- **Orchestration language (composition language):** composes the basic rules defining the scheme of rules to follow at the detection engine. Different operators can be used to compose these rules: Algebraic, Logic, Correlation and Synchronization operators.

These rules are for attack description, scenario definition and detection rules. Thus, rules' defining is easier as policy creation has become a matter of composing predefined rules.

IV. FEASIBILITY

This architecture focuses on the concept of detecting attacks predefined classes and proposing the appropriate defense mechanisms. Our solution provides security by operating in the following way: (1) evaluation of the queries (events), (2) attack identification, (3) extraction of the scenario and the category that are relevant to the identified attack, (4) assessment of candidate defense mechanisms and (5) relevant ones execution. Our solution accepts different types of input. Data come from logs generated by operating systems and applications, information from the network and even alerts generated by IDS or WAF (traffic analysis systems in general). Input will be in IDMEF [14] or Syslog formats. For other specific formats, like ModSecurity output for example, a normalization of the input is done within dissection module.

The system interacts with sensors and actuators. These sensors can be system, network, application, firewall, IDS or WAF. The actuators can be a firewall or a reverse-proxy based WAF, able to alert, accept, drop or log. The sensors feed the information to the decision system which identifies the attack in question. The knowledge system is composed by the basic rule database and the orchestration rules that describe the policy defined by the user. It also includes attack schemes that need to be detected. When detected, the attack information is sent to AIDD to assess the attack and provide the attack class in order to select the optimal defense mechanism(s).

For attacks and events formalism, we consider enhanced attack trees [13] the most appropriate to describe scenarios of complex events. Besides, concerning the used language to build detection and description rules Lua [14] will be used as a language framework to build the domain specific language. The two different rule types are defined as following:

- Basic Rules are Functions: Basic_Rule_Type (Param1, Param2, Param3…).
- Orchestration Rules are lists: Orchestration_Rule {Hook_1, Rules_composition1, Hook_2, Rules_composition2 … *Option*}.

The identified attack is resolved into attack components: parameters indicating some aspects of the system, eventual malfunction or failure. They are composed of various anomalies which are observed by sensors. To illustrate how these rules are used, we take the example of SQL injection attack. To describe the policy which is set up we use our combined rule sets. The basic rules used are the following:

- R1 = Spec_Rules (SQL, password, SQLi_chars, 3), where SQLi_chars are regular expressions for SQL injection pattern matching and 3 is the query score.
- R2 = Action_Rule (Log, SQLi attempt)
- R3 = Action_Rule (drop)
- R4 = Control_Rule (Version, ueq, 1)

An orchestration rule can be: Orchestration_Rule {H_URI, R1 AND R2, H_Version, R3 AND R4}, where H_URI is a hook at the URI level and H_Version the Version hook. This orchestration rule can be different as it expresses the policy defined by the user. Besides, this is a simple rule where we combine the different basic rules to detect the attack. It does not show necessarily the added value of our language as the detection is a simple matching but it shows at least the mechanism that simplifies the rule definition process.

V. CONCLUSION

So far, few rule based attack detection systems have taken into account the extensibility of the architecture and the simplicity of rules writing and a deterministic attack category based detection. In this paper, we have proposed a novel attack detection system that is easy to configure. It offers modular architecture and is able to learn from previous detected attacks. Our proposal will be used by system/network administrator to provide information in a rule-based and non-heuristic manner avoiding long and complex analysis.

The originality of our work is that the detection is no longer attack focused but attack category focused. This helps aggregate defense mechanisms and automates response. Creating policies is easier and Alerts seen by the system administrator are simpler allowing to have an overview of security state within a defined perimeter. As a future work, our system may include encrypted information handling and metrics to enhance the attack-defense matching.

REFERENCES

[1] D. Vennila, .R.Nedunchezhian "Correlated Alerts and Non-Intrusive Alerts", International Journal of Soft Computing, 2012

[2] S. Souissi, A. Serhrouchni "AIDD: A novel generic attack modeling approach", High Performance Computing & Simulation Conference 2014

[3] Snort IDS, http://www.snort.org

[4] Vern Paxson, "Bro: A System for Detecting Network Intruders in Real-Time", Lawrence Berkeley National Laboratory, CA,USENIX 1998

[5] Ivan Ristic, "ModSecurity Handbook: The Complete Guide to the Popular Open Source Web Application Firewall", 2010

[6] Naxsi: https://www.owasp.org/index.php/OWASP_NAXSI_Project

[7] C. Simmons, S. Shiva, H. Bedi, D. Dasgupta, "AVOIDIT: A Cyber Attack Taxonomy", Symposium On Information Assurance, 2014

[8] C. Simmons, S. Shiva, H. Bedi, V. Shandilya "ADAPT: A Game Inspired Attack-Defense And Performance Metric Taxonomy", IFIP Advances in Information and Communication Technology, 2013

[9] Z. Wu,Y. Ou, Y. Liu: "A Taxonomy of Network and Computer Attacks Based on Responses", International Conference of Information Technology, Computer Engineering and Management Sciences 2011

[10] Open Source Vulnerability Database OSVBD, http://www.osvdb.org

[11] Common Vulnerabilities and Exposures CVE, http://www.cve.mitre.org

[12] Debar H., Curry, D., Feinstein B "The Intrusion Detection Message Exchange Format (IDMEF)". RFC 4765

[13] S.A. Camtepe, B.Yener, "Modeling and detection of complex attacks", SecureComm, 2007

[14] R.Ierusalimschy, L. de Figueiredo, "Building Domain-Specic Languages over a Language Framework", PUC-Rio, 1997