A novel architecture and language concepts for web attacks detection

Abdelhamid Makiou, Samih Souissi, Ahmed Serhrouchni

A novel architecture and language concepts for web attacks detection

Samih Souissi and Abdelhamid Makiou
INFRES Department
Telecom ParisTech
Paris, France
{souissi, makiou}@telecom-paristech.fr

Ahmed Serhrouchni
INFRES Department
Telecom ParisTech
Paris, France
serhrouchni@telecom-paristech.fr

***Abstract*—In recent years, the web has emerged as an important gate to access resources and information systems. Consequently, web applications have become a privileged target for attackers. This paper presents a new formalism for web attacks detection. The objective is to simplify complex rules' expression, thanks to a modular architecture and intuitive syntax that give a high power of expression. The originality of our approach is that the syntax can be deduced from a certain behavior or automatically generated from valid behavioral scenarios. The paper presents the main concepts behind the proposed approach that allows dealing with the growing complexity of web applications and web attacks.**

Keywords- Web application; HTTP; Injection attacks; SQL injection; XSS; Web Application Firewall

I. INTRODUCTION

Web applications are actually widespread and HTTP protocol is becoming the new transport layer of these applications. In fact, nowadays, most of the applications are web-based. On the one hand, the need for web applications is becoming more significant and application developers have to respond to customer needs quickly, sometimes neglecting the security aspects. Thus, applications are more likely to be vulnerable to security attacks. On the other hand, attacks are becoming more complex, diverse and specific, targeting a particular type of application servers.

Many solutions to counter web attacks have been proposed. There are solutions that are specifically conceived for a particular type of attack or server while others are generic or based on modifying the source code of the application. These different existing solutions are not efficient for three main reasons. First, the scope of security of a solution specific to a type of attack is not appropriate for complex attacks that combine different methods to launch an attack on a web server. Second, generic solutions are complex to handle when defining effective detection rules. Third, a solution based on modifying the code needs a high expertise, is fastidious and requires a thorough knowledge of each web application to secure.

Therefore, the objective of this paper is to propose a modular and generic Web Application Firewall architecture in addition to a combined language to simplify rules' writing while improving detection performances. As SQL Injection attacks have been considered as one of the most important threats for web application [1], we have considered this attack as a first topic in our work.

This paper is organized as follows. Section II details the related work. Our proposal and ongoing work are described in section III. Finally, we deal with Future work in section IV.

II. RELATED WORK

In [2], Kruegel and Vigna propose an anomaly-based intrusion detection system for web applications. It characterizes HTTP requests using a number of statistical characteristics derived from the parameter's length, character distribution, structure, presence and order. This method focuses only on the incoming query parameters whereas it ignores the respective HTTP response. These results are either causing unnecessary false positives or missing certain attacks. AMNESIA [3] is an SQL injection detection and prevention system which combines static analysis and runtime monitoring. It uses a model based approach to detect illegal queries. Nonetheless, it requires source code changes in the web applications. Johns et al. [4] proposed an anti XSS solution that operates by matching incoming data and outgoing JavaScript. The model uses similarity metrics but is not suitable for dynamic JavaScript response and requires a learning process before deployment on the server side.

ModSecurity proposed by Ivan Ristic [5] is an open source solution based on signature attack detection. ModSecurity is widely used and has 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 formalism is very complex, needing a high expertise in HTTP protocol and in regular expressions. Another recent open source is NAXSI [6]. It 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.

The systems mentioned above do not offer a compromise between acceptable performance and formalism simplicity. Thus, defining good security rules that guaranties a suitable overall security level is not obvious. For a matter of formalism simplicity and better performances, some approaches offer adapted solutions for a kind or two of attacks. However, unlike attacks that evolve very quickly, they are neither open nor scalable. Other radical approaches focus on modifying the application code source which is a tedious task that should be customized according to each application.

III. PROPOSAL AND ON GOING WORK

In this section, we describe both the proposed language and the architecture of our solution.

A. *Modular architecture*

Our proposed architecture is independent of the server's type. It is a modular system that can fit to any kind of web platform. In fact, the detection engine includes different modules that allow a comprehensive analysis of the HTTP transaction.

This analysis begins with a loading phase of all orchestration rules. Secondly, the HTTP request is dissected into several significant fields and hooks are at the end of each field dissected. Orchestration rules, which are already loaded in memory, are assigned to each hook. Once a hook is reached, the engine applies the rules related to this hook. Fig. 1 shows this mechanism. The system get access to the disk to load the basic rules related to every orchestration rule to perform. This allows caching all detection logic expressed within orchestration rules and load only necessary basic rules every time a hook is reached. Fig. 2 shows the different phases of HTTP request handling.

B. *Composed language*

The complexity of the formalisms used in current WAFs has led us to propose a solution to simplify the writing of security rules. These security rules can include security checks on http protocol and attack signatures expression.

The formalism combines two languages: a *basic language* and an *orchestration language*. The basic language offers simple and specific rules used to describe either a pattern or a control. It is used to hide the complexity of security rules' expression to a non-expert user (instead of writing long and complex regular expressions). These rules serve as components to build another language called *Orchestration language*. This language is used to define the context of a particular attack, such as SQL injection for example, by composing the basic rules through simple operators. In addition, those basic rules can be generated automatically from a behavioral analysis. The format and an example of both languages are given in Table I.

TABLE I. PROPOSED LANGUAGE

	Basic Language	***Orchestration Language***
Format	* Rule_type | matching expression | {options} | Rule_ID	* Hook | {pattern class} | Rules composition |{options}
Example	*SQLRule 'password' SQLi_character score 3 R1 * ActionRule Log R2	* H_URI SQLi_Evasion R1 AND R2

IV. FUTURE WORK

We are currently dealing with the specification of the scoring system for correlation between the request and the response, in addition to the syntax and the scope of both basic and orchestration languages.

After defining the different elements of our architecture and the SQL injection detection formalism, we are focusing on the optimal placement of the hooks, trying to find the appropriate dissection for HTTP requests. We are also directing our work towards transaction content classes' definition using statistical methods. Those methods will help evaluate the transaction to determine different classes related to attacks' type.

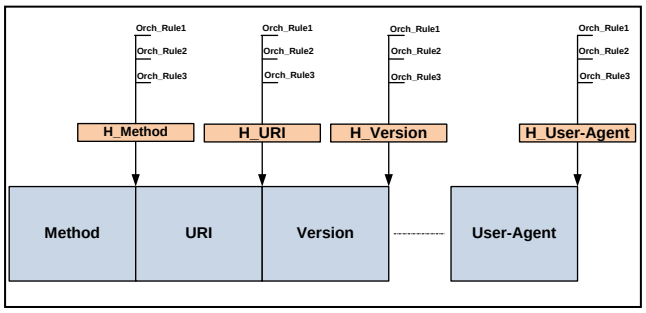

Figure 1. Hook mecanism

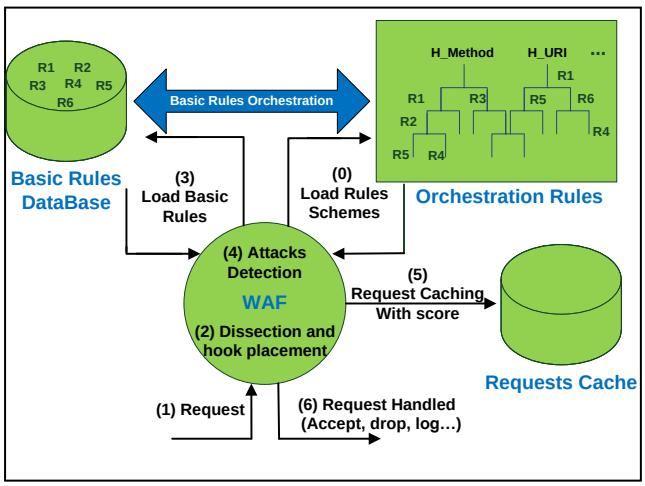

Figure 2. Request handling mecanism

REFERENCES

[1] "The Top 10 2013 classification of attacks on Web applications" https://www.owasp.org/index.php/Top_10_2013-Top_10

[2] C. Kruegel and G. Vigna. "Anomaly detection of web-based attacks". 10th ACM Conference on Computer and Communication Security (CCS '03), pages 251–261. ACM Press, October 2003.

[3] W.G. Halfond and A. Orso, "AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks," Proc. 20th IEEE and ACM Int'l Conf. Automated Software Eng., pp. 174-183, Nov. 2005.

[4] M.Johns, B.Engelmann, J.Poegga, "XSSDS:Server-side Detection of Cross-site Scripting Attacks on Computer Security Applications " in International Conf. ACSAC 2008.

[5] Ivan Ristic : ModSecurity Handbook: The Complete Guide to the Popular Open Source Web Application Firewall, 2010 Feisty Duck Ltd Edition ISBN: 1907117024 .

[6] Naxsi (Nginx Anti Xss & Sql Injection) October 2013 https://www.owasp.org/index.php/OWASP_NAXSI_Project