ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 2012. No. 3(119)
ELEKTRONIKA IR ELEKTROTECHNIKA

SYSTEM ENGINEERING, COMPUTER TECHNOLOGY
T 120
SISTEMŲ INŽINERIJA, KOMPIUTERINĖS TECHNOLOGIJOS

Securing Web Application by Embedded Firewall

E. Kazanavicius, V. Kazanavicius, A. Venckauskas
Computer Department, Kaunas University of Technology,
Studentų str. 50, LT-51368 Kaunas, Lithuania, phone: +370 37 300309, e-mails: ekaza@ifko.ktu.lt, vkaza@ifko.ktu.lt, algimantas.venckauskas@ktu.lt
R. Paskevicius
JSC "Piligrimas",
Vilniaus str. 74a, LT-44291 Kaunas, Lithuania, e-mail: rokas@debian.lt

crossref http://dx.doi.org/10.5755/j01.eee.119.3.1366

Introduction

In today's society, Internet technology is widely used. According Internet service provider Netcraft [1], in December 2011 there was 555,482,744 websites, which is an increase of 5.6%, 175 million of which were active. The main indicator of the quality of Internet service is safety and reliability. Based on the evaluation of vulnerability assessment analyzes performed by management service provider WhiteHat Security [2] only 16% of websites were not vulnerable during the year 2010. WebCohort, Inc. studies [3] indicate that at least 92% of web applications are vulnerable to an attack of some kind.

Web portals are being attacked on various levels of system architecture: Operating System, Application Server, Web Server, Database Server, Application or Network. Up to 75% of attacks are aimed at the Application Layer [4]. The situation in Lithuania is similar [5]: 79% of business companies and even 100% of Internet services providers are facing with Internet security problems.

Web application vulnerability

U.S. NIST logs National Vulnerability Database (NVD), has registered 48,790 vulnerabilities by 12/10/2011 [6]. In order to understand the causes of vulnerabilities better, to analyze their impact on web application security vulnerabilities and to develop better management tools, they were classified in different ways, emphasizing certain properties. Vulnerability classifications are a few, some of them: NVD Common Vulnerability and Exposure (CVE) [6], Web Application Security Consortium (WASC) Threat Classification [7], Open Web Application Security Project (OWASP) Critical Web Application Security Risks [8], Common Weakness Enumeration (CWE ™) [9]. Research a variety of the most common vulnerabilities inherent in web application summarizes the top five in Table 1.

Table 1. The most widespread vulnerabilities

OWASP [8]	CWE/SANS [10]	NVD [6]	WASC [11]	WhiteHat [2]	iMPERVA [12]
Injection	Injection	Injection	Cross-Site Scripting (XSS)	Cross-Site Scripting (XSS)	Directory Traversal
Cross-Site Scripting (XSS)	Cross-Site Scripting (XSS)	Cross-Site Scripting (XSS)	Information leakage	Cross-Site Request Forgery (CSRF)	Cross-Site Scripting (XSS)
Broken Authentication and Session Management	Missing Authentication and Authorization	Permissions, Privileges, and Access Control	SQL Injection	Insufficient Authorization	SQL injection
Insecure Direct Object References	Use of Hard-coded Credentials	Input Validation	Insufficient Transport Layer Protection	Predictable Resource Location	Remote File Inclusion
Cross-Site Request Forgery (CSRF)	Missing Encryption of Sensitive Data	Path Traversal	Fingerprinting	SQL Injection	

As showed in the table above, constructed top5 are similar. There are differences of terminology and position of vulnerability. Since OWASP classification are wide spread and has no big differences among other classifications, in further work will be used OWASP terminology.

Techniques to improve Web application security

To keep web application secure besides standard firewalls, various types of solutions are used in application layer: external tools – web application scanners and firewalls (WAS, WAF) and internal – the application itself must be self-defending [8].

A web application scanner is an automated program that examines web applications for specific security vulnerabilities [13]. A WAS uses the negative logic *(blacklist)* based filtering algorithms to detect vulnerabilities in web applications. Negative logic filtering built on signatures of known attacks and allows security systems to prevent any requests that appear to match the attacks' signatures from reaching protected servers.

A web application firewall (WAF, Fig. 1) is a firewall that operates in the Application layer of a web application server. It is kind of an intrusion detection system (IDS) but does not operate on the Network level of a web application; instead, it is exclusively applied on the Application layer [14, 15]. A WAF can work using any web traffic filtering mechanism (positive, negative, or session) and it either works as an embedded firewall with the web server or as a separated layer of security in a reverse proxy form [16].

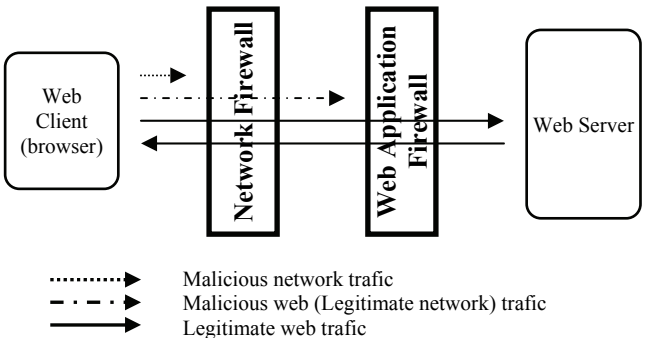


Fig. 1. Web Application Firewall infrastructure

Positive logic filtering allows valid requests based on a signature set *(whitelist)* detailing what types of communications protected server's know-how to handle; it prevents any requests not known to be valid from reaching secured servers.

Session based filtering (also called event-driven dynamic rules) utilizes positive logic based rules but allows the inclusion of variables in the rule set. The values of the variables are set dynamically during user sessions.

The disadvantages of positive logic filtering: is requirement of large vulnerabilities database based of regex rules. It causes poor bandwidth, requires more resources, hardly adaptable to large web systems. By reducing the number of rules, in order to improve bandwidth, decreases vulnerability detection quality.

In order to increase the WAF performance, bandwidth, WAF developed using artificial intelligence techniques: Artificial Neural Networks [17], fuzzy logic [18]. These systems address two objectives: to set models for 'normal' traffic and to detect 'abnormal' traffic, blocks it. However, this system is still experimental; challenging them to "train" also can't "see" part of attacks (up to 20%).

Since WAF can not to detect all attacks, it is not suitable to protect applications which use application-level encryption of data streams, so it uses an "inner" security measures integrated into the application when developing, there are established methods [19]. However, this method is work intensive, development of web application becomes more expensive and takes longer. After detecting new type of vulnerabilities may be a need to redevelop web application.

Our proposed solution – Embedded Web Application Firewall

Considering the experience of WAF development and usage, to eliminate main problems we are offering to embed WAF into web application i.e. embedded firewall model (EWAF), whose structure is represented in fig.2.

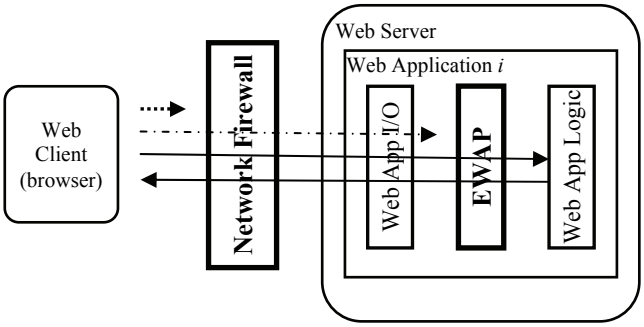


Fig. 2. Embedded Web Application Firewall structure

EWAP is a modular system and consists of two main parts (Fig. 3).

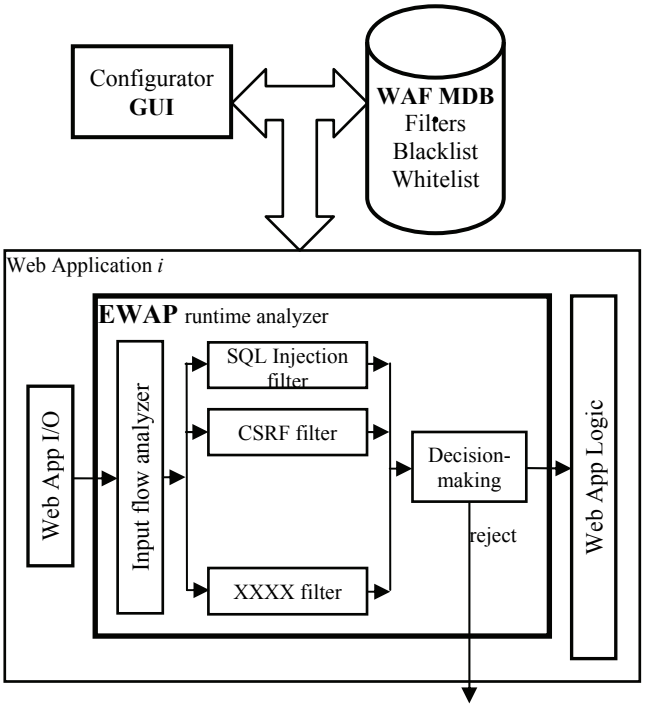


Fig. 3. A functional scheme of EWAF

The main components of EWAF: EWAF main database (MDB): vulnerability detection modules (filters) for each type of vulnerability: Injection, XSS, CSRF, etc., whitelist and blacklist. Particular modules are based on the

positive, negative and Web traffic filtering session mechanisms to discover vulnerabilities in input flow.

Configurator, the graphical user interface module: to create a firewall for each web application and supporting, through the MDBs. It works in two modes - manual (the initial setting, the correction) and automatic (EWAF filters, whitelist and blacklist update).

The primary input traffic analyzer. The analyzer parses the request (if necessary, decodes it) and determines the filters that apply to the subsequent analysis of the data input (the vulnerability detection).

EWF filter parses the input data to detect certain type of vulnerability.

EWAF decision-making module evaluates results of filters and rejects request if the vulnerability is detected or redirects the flow of data entry to the web application logic part.

EWAF is embedded into any web application and configured according to its needs by activating the appropriate filters and firewall rules sets.

Disadvantages*:* more administration because of the need to configure for each application separately; The advantages: bandwidth, efficiency (because of problem-oriented), flexibility – it's possible to analyze encrypted input flow.

EWAF evaluation

To evaluate our approach, we have developed a prototype of our solution EWAF to implement and use a firewall to ensure security of web applications. We realized the SQL injection detection and neutralization module and used this to secure web applications.

SQL injection detection and neutralization module parses input data from HTTP requests GET, POST, REQUEST, COOKIE. Module is based on positive and negative filtering mechanisms. DB of vulnerabilities (whitelist and blacklist) is stored in XML format.

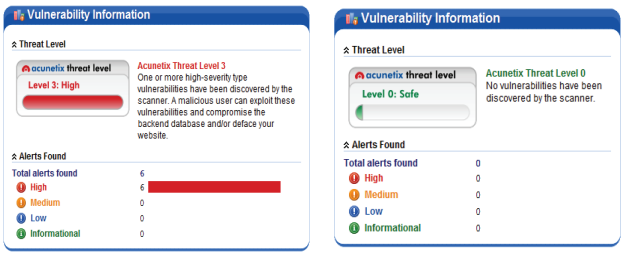
Evaluation criteria of developed *EWAF*:

- Qualitative characteristics - accuracy of the firewall (if detects all modeled vulnerabilities):
  - detection of positive vulnerabilities,
  - detection of negative vulnerabilities (false-positive).
- Quantitative characteristics:
  - Time delay (or latency) during execution of web application,
  - Increase of server load.

To measure accuracy we used *Acunetix Web Vulnerability Scanner* [20] and based on OWASP recommendations threats and threats fractions list. Specific web application form was tested. Test results shown: *Vulnerability Scanner* detects 6 vulnerabilities while firewall was off; and 0 after turning firewall on (Fig. 4).

Special text file with the familiar 408 false attack symptoms was used to test for response to false vulnerability. EWAF did not respond to false attacks.

To evaluate quantitative performance EWAF was simulated access to the web application transactions, using a predefined randomly generated data file access and *NeoLoad* software [21], which is used to test web applications by simulating the same time existing customers.



a) b)

Fig. 4. Vulnerability detection test: a – without a firewall; b – with a firewall

Were measured at login time and the number of queries to the web server and DB per second when EWAF was turned off and turned on later, by simulating the number of users competing for 4 to 10 and sequentially changing the number of connections from 100 to 10000. Table 2 shows average results of measurements of turned off /on and the percentage change.

Table 2. Query execution time

Simulated competitive users	4	6	8	10
Average response time(s)	0,49 0,531 8,4%	0,738 0,774 4,9%	0,971 1,06 9,2%	1,25 1,34 7,2%
Queries per second server	7,1 6,7 -5,6%	7,1 6,7 -5,6%	7,2 6,7 -6,9%	7,1 6,8 -4,2%
Queries per second server DB	7273 6780 -6,8%	7143 6742 -5,6%	7273 6957 -4,3%	7194 6849 -4,8%
Test duration (s)	55 59 7,3%	84 89 6%	90 95 4,5%	139 146 5%

As can be seen from the table, while EWAF turned on average query execution time increases by 6.5%, the number of requests served by web server is reduced by approximately 5.6%, and queries to the database - 5.2%. Time depends on the number of simulated users, but it also depends on the server capabilities.

The goal of stress test is to ensure the EWAF does not crash in conditions of insufficient computational resources unusually high concurrency, or denial of service attacks. The following studies to evaluate the blacklist and whitelist mechanisms of influence on the stress test results. They simulate the flow of queries from 100 to 10,000 during the, holding that each request is an attack and a whitelist or blacklist checking filters. Table 3 shows average results of measurements of blacklist and whitelist filters.

As can be seen from the table, WAF with whitelist filters works faster, with a high number of queries should be limited to whitelist filters. Using a blacklist filters, server load average are at 350 requests per second,

approaching the limit (0.5) value and a very sharp upward trend.

Table 3. Stress test results

Queries scanned	**100**	**1000**	**5000**	**10000**
Duration (s)	0,33 0,11	2,96 0,83	14,51 4,61	28,53 8,48
Queries per second	303 909	338 1205	345 1065	351 1179
Server load averages change	0,01 0,01	0,01 0,01	0,15 0,06	0,4 0,14

Conclusions

Security of web application is actual problem – only 8 – 16% of web applications are secured, 75% of vulnerabilities goes to Application layer. There are various tools to improve security of web applications: WAS, WAF and safe programming methodology. However, using of these technologies has some problems.

In this work offered to use Embedded Web Application Firewall (EWAF) for ensuring web applications security. EWAF model was created by using blacklist and whitelist filters, prototype of it was tested.

Research was done by using well known web applications evaluation tools, it had showed that created EWAF had detected all tested vulnerabilities, average query execution time increases by 6.5%. Stress test analysis shown that whitelist filters comparing with blacklists are better when loads a higher.

EWAF model can be used in developing new or improving existing security measures on any web application platform. Created SQL injection filters also can be applied as a single security measures in existing web application.

References

1. **NETCRAFT.** Web Server Survey, 2010. Online: http://news.netcraft.com/.
2. **WhiteHat.** Website Security Statistic Report**,** 2011, 11t ed. Online: https://www.whitehatsec.com/assets/WPstats_winter11_11th.pdf?doc=WPstats_winter11_11th.
3. **WebCohort.** WebCohort's application defense center reports results of vulnerability testing on Web applications [EB/OL], 2004. Online: http://www. imperva.com/news/press.html
4. **Lanowitz T.** Now Is the Time for Security at the Application Level. – Gartner, 2005. – 8 p.
5. **Rainys R.** Network and Information Security. Assessments and Incidents Handling // Electronics and Electrical Engineering. – Kaunas: Technologija, 2006. – No. 6(70). – P. 69–74.
6. **NIST.** National Vulnerability Database Version 2.2. Online: http://nvd.nist.gov/home.cfm
7. **The WASC Threat Classification** v2.0. Web Application Security Consortium. Online: http://projects.webappsec.org/w/page/3246978/Threat%20Classification.
8. **OWASP.** The Ten Most Critical Web Application Security Risks, 2010. Online: https://www.owasp.org/index.php/Top_10.
9. **Common Weakness Enumeration** (CWE™). Online: http://cwe. mitre.org/.
10. CWE/SANS Top 25 Most Dangerous Software Errors, 2011. Online: http://cwe.mitre.org/top25/
11. **WASC** (Web Application Security Statistics). Online: http://projects.webappsec.org/w/page/13246989/Web%20Application%20Security%20Statistics.
12. **Imperva's Web Application Attack Report.** – 2011 Online: http://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed1.pdf.
13. **Fong E., Okun V.** Web Application Scanners: Definitions and Functions // Proceedings of the 40th Hawaii International Conference on System Sciences. – IEEE, 2007. – P. 280b–280b.
14. **Becher M.** Web Application Firewalls. – VDM Verlag, Saarbrücken, Germany, 2007. – 168 p.
15. **Kapodistria H., Mitropoulos S., Douligeris Ch.** An advanced web attack detection and prevention tool // Information Management & Computer Security, 2011. – Vol. 19. – Iss. 5. – P. 280–299.
16. **Desmet l., Piessens F., Joosen W., Verbaeten P.** Bridging the gap between web application firewalls and web applications // Proceedings of the fourth ACM workshop on Formal methods in security. – Alexandria, Virginia, USA, 2006. – P. 67–77.
17. **Krueger T. , Gehl Ch., Rieck K., Laskov P.** TokDoc: a self–healing web application firewall // Proceedings of the 2010 ACM Symposium on Applied Computing. – Sierre, Switzerland, 2010.
18. **Moosa A.** Artificial Neural Network based Web Application Firewall for SQL Injection // World Academy of Science, Engineering & Technology, 2010. – Vol. 64. – P. 12–21.
19. **Guide to Building Secure Web Applications and Web Services.** – OWASP, 2005. – 293 p.
20. **Acunetix Web Vulnerability Scanner.** Online: http://www.acunetix.com/vulnerability–scanner/.
21. **NeoLoad.** The load testing solution for all web applications. Online: http://www.neotys.com.

Received 2011 11 02
Accepted after revision 2011 12 15

E. Kazanavicius, V. Kazanavicius, A. Venckauskas, R. Paskevicius. Securing Web Application by Embedded Firewall // Electronics and Electrical Engineering. – Kaunas: Technologija, 2012. – No. 3(119). – P. 65–68.

The paper addressed the problem of protection of Web applications. Classifications of Web application vulnerabilities and existing protection methods had been analyzed. The paper proposed an embedded web application firewall model based on a "black" and "white" lists of vulnerabilities which allows flexible and efficient development of Web applications firewalls, to assess the specifics of each program. Developed and tested a prototype of a firewall, to show an advantage of embedded Web application firewall. Ill. 4, bibl. 21, tabl. 3 (in English; abstracts in English and Lithuanian).

E. Kazanavičius, V. Kazanavičius, A. Venčkauskas, R. Paškevičius. Interneto programų apsauga naudojant įterptines užkardas // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2012. – Nr. 3(119). – P. 65–68.

Darbe sprendžiamos interneto programų apsaugos problemos. Išnagrinėtos interneto programų pažeidžiamumo klasifikacijos, esami apsaugos metodai. Pasiūlytas „juodaisiais" ir „baltaisiais" pažeidžiamumo sąrašais paremtas įterptinės interneto programų užkardos modelis, kuris leidžia lanksčiai ir efektyviai kurti interneto programų užkardas, įvertinančias kiekvienos programos specifiką. Sukurtas ir ištirtas užkardos prototipas, parodyti įterptinės interneto programos užkardos sudarymo pranašumai. Il. 4, bibl. 21, lent. 3 (anglų kalba; santraukos anglų ir lietuvių k.).