Web Application Firewall (WAF)

Alexander Endraca, Bryan King, George Nodalo, Maricone Sta. Maria, and Isaac Sabas

***Abstract*—Web Servers are core components within the networking industry and as such, the need for security for these critical elements is tremendous. Thus, a Web Application Firewall is deployed to protect the web server against possible vectors of attack. The Web Application Firewall is focused on the 7th layer; The Application layer of the OSI Model. Access Controls are implemented by using Access Control Lists as its rules to allow or reject traffic. The use of iptables userspace application (A part of the Linux kernel) is used to queue the packets at the kernel level and to direct the packets to go through the WAF first before it enters into the web server. The thorough inspection of the packets and the decision making for packets are done in the kernel level. All accepted packets are forwarded to the user level where the web server is running. The Web Application Firewall is able to compare the Access Control List, which is configured by the administrator through any text editor, against the incoming HTTP packets from the traffic before it reaches the web server itself. The algorithm used to compare the payload of the packet is simply pattern checking with the use of regular expressions. The testing results are proof on how accurate the Web Application Firewall is in detecting and rejecting different types of attacks in accordance of the top 10 web application attacks from OWASP.**

***Index Terms*—Access control, firewalls, layer 7 information, web server.**

I. Introduction

Web servers are assets to the business and the Internet in general because these machines are the source of information and where information is stored. These web servers deliver data such as web pages, images, videos and other types of data available to the client when requested.[1], [2] A Web server can to refer to a web server software or the server hardware. Within the Internet infrastructure, web servers are run in a constant state to serve requests from the other networks (Internet). The Apache HTTP Server is one example of a web server program and is one of the most popular web server programs in use today. Its strengths lie in the fact that it is an open source program maintained by a community under the Apache Software Foundation, and it caters to wide variety of operating systems such as Unix, Linux, Microsoft Windows and Mac OS X.

Because of the important role a web server plays and where it is situated in a logical diagram of a network; it is subject to malicious attacks perpetuated by hostile parties and/or entities. The usual goals of these attacks vary from disrupting the function and availability of the web server to gaining unauthorized access to hosted web content. To prevent these attacks, there are network appliances that are added to the computer network such as Intrusion Prevention System (IPS) and Intrusion Detection System (IDS) [3], [4]. Both IPS and IDS help monitor the network but are only limited to detecting and notifying administrators about the abnormal network behavior and can still succumb to complex attacks or attacks that may not have been recognized by the system. IPS checks the signature of the attacks and must rely on patterns to determine if there is an attack.

As security risks arise, various, and ever-changing attacks are carried out against the web server such as: Injection attacks, Cross-Site Scripting (XSS), Insecure direct object references to name a few. Injection attacks refer to injections perpetuated through SQL, OS and LDAP. Through these attacks, malicious data is sent to the website and executed by the program. Cross-Site Scripting also known as XSS occurs when untrusted data goes through invalidated fields [5]. Whilst with Insecure Direct Object References, files and directories may be accessed by unauthorized people through exposure of referenced objects. Different techniques and practices are used to defend against these threats. To mitigate injection attacks, white-listing and input validations are implemented to prevent such attacks. For Cross-Site Scripting (XSS), proper coding of SQL Queries and white-listing is also used. For Insecure Direct Object Reference, session management and check access are used [6].

Firewalls are the first line of defense for web servers and by extension the rest of the network. Firewalls allow connections to pass through by following rules managed by network administrators. However, these rules are inadequate as time passes because it is difficult to distinguish whether a packet pattern is malicious or not, thus some legitimate connections are blocked, and some illegitimate connections are permitted. To better protect the network, the state-full packet inspection (SPI) firewall was developed. A SPI firewall checks the header and footer of a packet ensuring that it belongs to a valid session, but it does not check the data inside the packet. Which may still contain malicious content [7]. Finally, a third generation of firewall known as Application Firewalls was developed, which checks not only the header and footer part of the packet but also the data portion. Based upon on the content of the packet the firewall now decides on whether to allow the packet or reject it and controls what type of traffic can be passed to the application layer of network service.

The web application firewall is a type of firewall that

Manuscript received May 5, 2013; revised July 26, 2013.

Alexander Endraca, Bryan Genesis King, George Nodalo, and Maricone Sta. Maria are with the College of Computer Studies, De La Salle University- Manila (email: endraca.alex@gmail.com, bryan_genesis_king@yahoo.com, george_nodalo@dlsu.ph, maricone.stamaria@yahoo.com).

DOI: 10.7763/IJEEEE.2013.V3.277

checks the data level of the packets to protect the application layer of the OSI model. By checking the data portion of the packets, more detailed information is revealed which is referred to as the granularity of a packet [8]. For example, inside the HTTP header there would be http requests and inside http request would be user agents, cookies and more. Now being able to see this information, a more informed decision is now made in regards to the security controls for specific packets passed to the application. Existing appliances and open source solutions are available such as the Barracuda Web Application Firewall and the modSecurity open source web application firewall. Each implementation makes use different algorithms and policy models. The Barracuda Web Application Firewall uses its Barracuda web filter architecture to filter out legitimate traffic from malicious traffic. By letting it passes on to a hierarchical mechanism which would evaluate each packet [9]. The modSecurity open source web application firewall makes use of different security models to decide whether to allow or reject traffic: Negative Model where all known bad requests are rejected, Positive Model where all known requests that are valid are allowed and everything else is rejected, and the Extrusion Detection Model where outbound data is blocked, identified and monitored [10].

The protection models used by existing web application firewalls are pertaining to traffic control. The inclusion of an access control on web application entities such as pages and files provide additional security. Examples of access control models are Role-based Access Control (RBAC) and Mandatory Access Control (MAC) [11]. RBAC model which is an access control technique where administrators can specify privileges and roles to provide access. This could be used to protect that web application from traffic flood from illegitimate users. MAC on the other hand provides control over file access. Access can be granted on a particular file based on the particular permission set [12].

The Web Application Firewall is installed as a running service in the web server or system it needs to protect, particularly the application layer level. Its main purpose is to check all incoming HTTP traffic, then accepts and drops the incoming HTTP traffic according to the rules that was set by the network administrator. The administrator through a text editor configures the rule-sets of the Web Application Firewall. A manual is provided for the syntax and format of the rules. The structure of the rules has the keywords "allow" or "reject" as its basis for the decision, followed by the different options of HTTP request headers and the value which the administrator wants to be checked in the payload.

II. Web Application Firewall Engine

The WAF Engine is the main component of the Web Application Firewall. The Web Application Firewall is installed in the same machine as the Web Server. The Web Application Firewall is limited only to Apache HTTP Web Server. The WAF Engine consists of two modules namely the Packet Analyzer Module and the Configuration Module.

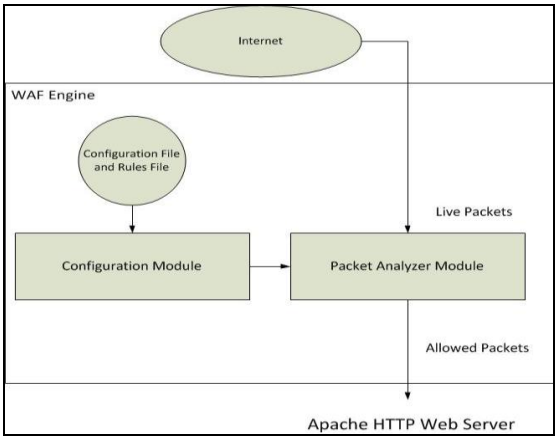


Fig. 1. System architecture.

A. Packet Analyzer Module

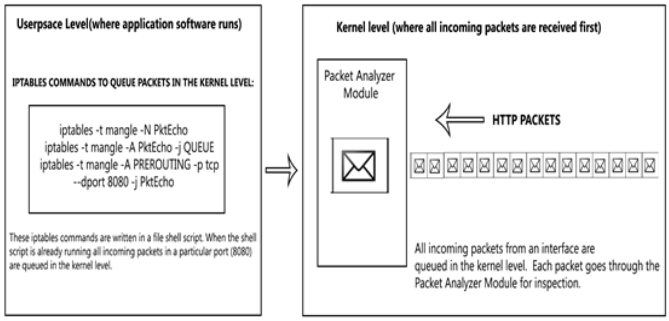


Fig. 2. Packet analyzer.

Fig. 2 illustrates how packets are queued in the kernel level and where the packets are inspected. Iptables is a userspace command to queue the packets in the kernel level and to direct the packets to go through the Packet Analyzer first. [13] The iptables is used by running the following commands:

```
iptables -t mangle -N PktEcho
iptables -t mangle -A PktEcho -j QUEUE
iptables -t mangle -A PREROUTING -p tcp --dport 8080
-j PktEcho PktEcho
```

Fig. 3. Iptables commands.

These commands are written in a file shell script. This shell script should run first before the other modules of WAF so that all incoming packets do not go directly to the web server. When the shell script has run, all incoming packets in a particular port of a web server (e.g. 8080) are queued in the kernel level. Each queued packet in the kernel level goes through the Packet Analyzer Module where the inspection of packet is done.

Packets that access the Apache HTTP Web Server are sniffed by the Packet Analyzer for analysis of data. The data of the packets that had been analyzed are used to decide to allow or reject the packets. The information about the rejected packets is logged in order to be used for analysis in the future. The response of the Web Server is not monitored since it is assumed that the Web Server that is protected is safe.

B. Configuration Module

The Configuration Module applies the settings for the

Web Application Firewall and the Access Control List of the Web Application Firewall. The settings and Access Control List are configured via text files. There is a specific syntax followed for both the settings and Access Control List of the Web Application Firewall. The Configuration Module checks both configuration files to see if there are errors in the input and in the syntax. When no errors are found in the configuration files, the Configuration Module applies the setting and the Access Control List in the Web Application Firewall.

III. Results

A. Access Control List Configuration File

The Web Application Firewall works by reading the Access Control List Configuration text file. The Access Control List Configuration text file is already premade with basic configurations and may be edited by the administrator to satisfy the security needed by the Web Application. The file is parsed and generated automatically into "ACL.rules" file which is read by the next module to synchronize its purpose to the firewall.

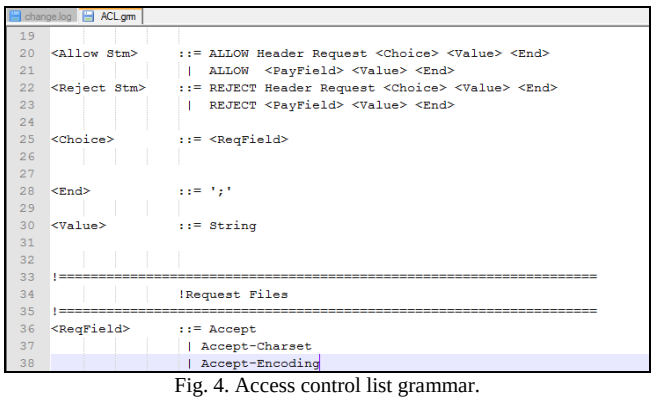


Fig. 4. Access control list grammar.

Fig. 4 above shows, how the syntax of the Access Control List Configuration file are formed in order to be parsed. The attributes that can be set as rule parameters in the Allow or Reject Statement are listed also in the Access Control List Grammar file. Some example of the attributes are "method", "form" for the <PayField> rule parameter. For the <Choice> parameter, the attributes are "accept", "user-agent". And for the <Value>, the attributes that are set depends on the data the administrator wants to input. The Access Control List Configuration file must follow the Allow and Reject Statement word per word, otherwise the file has a syntax error when parsed and cannot be used by the Web Application Firewall since it is not updated.

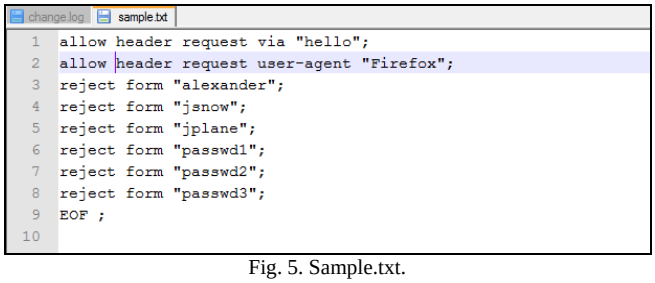


Fig. 5. Sample.txt.

Fig. 5 above shows a sample of an Access Control List Configuration file that is made by the administrator and is parsed and undergone syntax error checking and outputs a file seen below in Fig. 6. The options that are used in the "ACL.txt" are based on the grammar file that is followed in creating an Access Control List Configuration file. An example would be the Request fields Option which can be User-Agent, Via, Warning and any of the options that are listed in the grammar file. In creating an Access List Configuration file, the order of the Access Control must be in higher to lower priority since the Web Application Firewall reads the Access Control List Configuration file line by line.

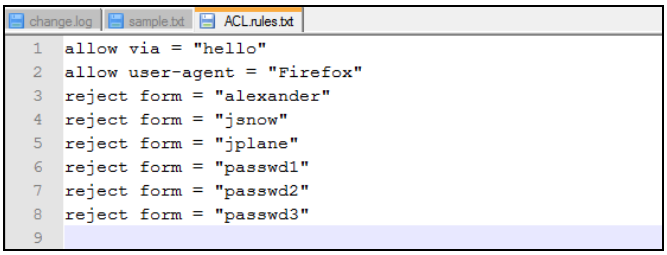


Fig. 6. ACL.rules.

Fig. 6 above shows the "ACL.rules" which is the result of parsed ACL configuration text file using the grammar file as basis for the syntax. The data that is first parsed in the Access Control is the method that has the value "POST" or "GET". After determining whether the method is "POST" or "GET", the "Request Field" or "PayField" would be the next to be determined. The parsing of the .txt file shows a new format that the Web Application Firewall can recognize to be used by for allowing or rejecting the packet.

B. Packet Parser

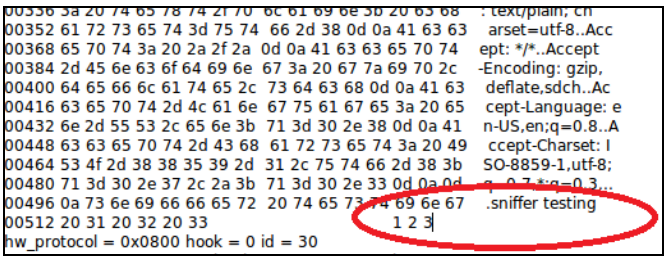


Fig. 7. Parsed packet.

Fig. 7 above shows an example of a parsed packet that is checked by the Web Application Firewall whether it can access the Web Server or be dropped by the Web Application Firewall. The highlighted part in the figure is the data of the payload section of the packet, showing that the system is able to parse values and data in the application level. The parsed packet contains data such as the header values and the payload. Based on the access control list that was made the web application firewall knows what packets should be allowed or should not be allowed to access the web server.

C. Data Comparison

The Web Application Firewall reads the Access Control List first and lists everything that is allowed to access the Web Server and the packets to be dropped. It then searches the whole packet including the headers and the payload.

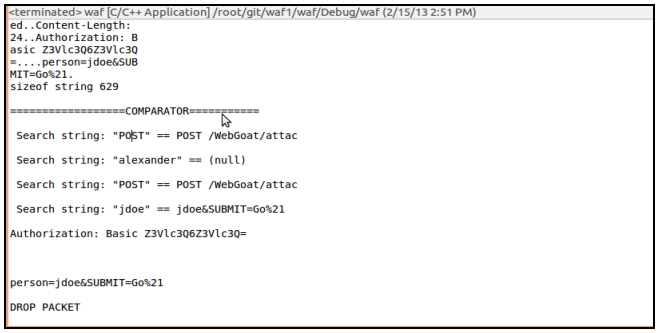


Fig. 8. Dropped packet.

Fig. 8 above shows the process on how the Web Application Firewall compares data and decides whether to allow or drop the packet. For example, the word "jdoe" must be dropped because it was declared by the administrator in the rules. When the Web Application Firewall parses the packet and finds the word "jdoe", then it drops the packet.

Before dropping the packet, all the data parsed from the packet are compared against the rules that have been made by the administrator. The algorithm that is used in detecting values is a regular expression pattern check algorithm. This algorithm uses regular expression patterns that were generalized carefully to match attack patterns or malicious patterns that are common in the packet. [14]

In the Data Comparator, the algorithm searches for the value inside the packet, if the value matches any of the patterns that was taken from the rules then necessary actions are being made depending on the rules that were associated with it.

TABLE I: RESULTS OF THE DECISION

ACL (Value)	Decision	Packet Data (Value)	Result
jdoe	Reject	Jdoe	Dropped
jplane	Allow	Jplane	Allowed
alexander	Reject	null	Dropped
passwd1	Allow	passwd1	Allowed
passwd2	Allow	passwd2	Allowed

The table above shows the results wherein continuous packets are checked whether they are accepted to access the Web Application Firewall or not. In the ACL column, the values of the data are listed and it is based on the Access Control List Configuration text file. The values seen in the ACL column are extracted from the Access Control List Configuration text file, rules, which were created by the administrator. For example (allow payload = "jdoe"), the jdoe value will be extracted for comparison purposes. The Web Application Firewall's decision is listed on the Decision column, which is also found in the Access Control List Configuration file. The Web Application Firewall sets the flag to 0 if the decision is reject and 1 if the decision is allow. The Web Application Firewall compares the data of the packet to the Access Control List Configuration file that has been listed one by one to see if the packet contains data similar to the data of the configuration file.

In Table I, the value "jplane" in the Access Control List has the decision of Allow. If a certain packet that is accessing the Web Server has data containing "jplane". The Web Application Firewall acts based on the flag that is set. In the test that was conducted, the Web Application Firewall allowed the packet containing the data "jplane" since the flag is set to 1 because the decision is Allow.

TABLE II: RESULTS OF INJECTION ATTACKS

Attack	Number of Attacks Sent	Number of Attacks Dropped
SQL Injection	50	46
XSS Injection	50	50

The table above shows the results of another test conducted on the Web Application Firewall wherein continuous attacks try to penetrate into the Web Server. In the test, two injection attacks are used in testing the Web Application Firewall, namely SQL injection and XSS injection. Different examples of SQL and XSS injection attacks are used in the test. One example used for SQL injection attack is "SELECT name FROM users WHERE name='" & name & "' AND '" & password & "' ' OR (SELECT COUNT(*) FROM users)>10 AND "='". And for XSS injection attack "><script>alert('CSS Vulnerable')</script><*b a=a*". 46 out of 50 HTTP requests that contain SQL injection attacks and 50 out of 50 HTTP requests that contain XSS injection attacks, were dropped by the Web Application Firewall.

It is important that the Web Application Firewall is able to block not only packet data but also different type of attacks. SQL and XSS injection are used in testing the Web Application Firewall because these two attacks are included in the OWASP's Top 10 Most Critical Web Application Security Risk.

The Web Application Firewall strictly follows the decisions that were made for different data. It does not also allow the packet to Access the web server if the data it contains is not in the Access Control List Configuration file. Like in the Table I, the Access Control List has the "alexander" data. But since no packet has the value "alexander", the flag for the decision is set to 0 and the packet is dropped. In other case, if the administrator sets a particular method to be allowed, for example, "Allow method POST" is set as a low priority in the ACL, then all POST requests that do not contain rejected payload value are accepted.

IV. CONCLUSION

The current implementation of WAF specifically the Packet Analyzer and the Access Control List grammar; provides insight into the workings of firewalls and the processes involved in their function. There are several areas of the system that can be improved upon such as:

1) *Integrating SSL protocol:* Being able to integrate SSL functionalities into the firewall will go a long way into improving the security and, given the need in today's environment, it is a priority improvement once the system is stable.
2) *Increasing the compatibility of the system:* The WAF *currently* runs only for any web server running within the Linux Ubuntu environment. A worthy improvement would be to extend it to other platforms.

ACKNOWLEDGMENT

First off all, the group would like to thank God for all the blessings and wisdom that He has provided us as we go through our thesis. Also, thank you for Sir Isaac Sabas in helping us get through in tight situations and giving us encouragement as our thesis adviser. To our friends and families who gave their all to support us until this time and in the future. We would not be able to do all of these things without you.

REFERENCES

[1] A. Razzaq, A. Hur, S. Shahbaz, M. Masood, and R. Ahmad, "Critical analysis on web application firewall solutions," in *Proc. IEEE Eleventh International Symposium on Autonomous Decentralized Systems*, 2013, pp. 1-6.

[2] S. Lin, "From web server security to web components security," in *Proc. IEEE 37th Annual 2003 International Carnahan Conference on Security Technology*, 2003, pp. 176-182.

[3] G. Young. (2011, October 7). HP TippingPoint Next-Generation Intrusion Prevention System (NGIPS). [Online]. Available: http://docs.media.bitpipe.com/io_10x/io_104864/item_535989/HPTippingPointNGIPS.PDF

[4] K. Scarfone and P. Mell. (Feburary 2007). Guide to Intrusion Detection and Prevention Systems (IDPS). [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf.

[5] A. E. Nunan, E. Souto, E. M. dos Santos, and E. Feitosa, "Automatic classification of cross-site scripting in web pages using document-based and URL-based features," in *Proc. IEEE Symposium on Computers and Communications*, 2012, pp. 702-707.

[6] OWASP. (2010). Owasp top 10 - 2010: The ten most critical web application security risks. [Online]. Available: https://www.owasp.org/images/0/0f/OWASP_T10_-_2010_rc1.pdf

[7] A. Frederic, "Firewalls and Internet Secuirty," *The Second Hundred Years*, vol. 2, no. 2, 1999.

[8] SANS Institute. (2012). The OSI Model: An Overview. [Online]. Available: http://www.sans.org/reading_room/whitepapers/standards/osi-model - overview_543

[9] Baraccuda Networks. The Barracuda Web Application Firewall: Best practices for planning and defending against attacks by anonymous. [Online]. Available: https://www.barracuda.com/docs/White_Papers/Barracuda_Web_Application_Firewall_WP_Defending_Against_Anonymous.pdf

[10] SANS Institute. (2011). Using web application firewall to detect and block common web application attacks. [Online]. Available: http://www.sans.org/reading_room/whitepapers/webservers/web-application-firewall-detect-block-common-web-application-attacks_33831

[11] Y. Fan, Y. Zhao, J. Liu, and Z. Han, "A mandatory access control model with enhanced flexibility," in *Proc. International Conference on Multimedia Information Networking and Security 2009, MINES '09*, vol. 1, 2009, pp.1120-124.

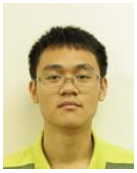
[12] W. Zhou and C. Meinel, "Team and task based rbac access control model," in *Proc. Latin American Network Operations and Management Symposium*, 2007, pp. 84-94.

[13] J. Xi, "A design and implement of ips based on snort," in *Proc. Computational Intelligence and Security (CIS)*, 2011, pp. 771-773.

[14] A. Heninger. (2004). Analyzing Unicode Text with Regular Expressions. [Online]. Available: http://www.icu-project.org/docs/papers/iuc26_regexp.pdf.

Alexander D. Endraca was born in Manila, Philippines in 1992. He is currently an undergraduate of Computer Science with specialization in network engineering of De La Salle University, Manila, Philippines.

He has worked at SMART Telecommunications Inc. as a Programmer during his internship.



Bryan Genesis W. King was born in Manila, Philippines. He is currently an undergraduate of Computer Science with specialization in network engineering of De La Salle University, Manila, Philippines.

He has worked at DLoads Inc. as a Web Application Programmer during his internship.

George S. Nodalo Jr. was born in San Pablo City, Laguna, Philippines in 1990. He is currently an undergraduate of Computer Science with specialization in network engineering of De La Salle University, Manila, Philippines.

Maricone A. Sta. Maria was born in City of San Fernando, Philippines in 1992. She is currently an undergraduate of Computer Science with specialization in network engineering of De La Salle University, Manila, Philippines.

She has worked as a programmer at Infor during her internship

Issac H. Sabas is currently a graduate of De La Salle University, Manila, Philippines, with a degree of Computer Science specializing in Network Engineering. He has worked at Perimeter E-Security USA (now Silversky) as a security analyst in the security operations center (SOC). Currently works at Internet Fraud Watchdog Online (IFW Online) as chief investigator. He currently focuses on network intrusion analysis and malware analysis.