Web Application Firewall With Telegram Bot Integration

Mohamad Amshar Solaiman
Universiti Kuala Lumpur
UniKL MIIT, Jalan Sultan Ismail,
Kuala Lumpur, Malaysia
amshar.solaiman@gmail.com

Herny Ramadhani Mohd Husny
Universiti Kuala Lumpur
UniKL MIIT, Jalan Sultan Ismail,
Kuala Lumpur, Malaysia
herny@unikl.edu.my

Dalilah Abdullah
Universiti Kuala Lumpur
UniKL MIIT, Jalan Sultan Ismail,
Kuala Lumpur, Malaysia
dalilah@unikl.edu.my

Norsuhaili Seid
Universiti Kuala Lumpur
UniKL MIIT, Jalan Sultan Ismail,
Kuala Lumpur, Malaysia
norsuhaili@unikl.edu.my

***Abstract* - With the increasing trends of web-based attacks and successful hacks through vulnerability in web application, many organizations has turned to Web Application Firewall (WAF) as a countermeasure to secure their website from intruders. Despite its effectiveness, WAF is still subjected to advanced attack in which web administrators may need real-time attack notification in order to mitigate successfully. Apart from that, most enterprise level WAF on the market is expensive and feature administration interface that is complex to use (steep learning curve). Staffs need to be trained to handle these devices and that in turn add up to more cost. The project aim to provide solution for these problems by developing a WAF that can be managed through the Telegram chat interface. This enable administrator to receive real-time notification while an attack is ongoing and also makes management of the WAF less complex due to the simplistic interface of the Telegram client. As a plus, the resulting product is also more cost effective compared to most enterprise-level WAF on the market, thus this project is targeted for SME companies, which mostly run e-commerce websites. A SME company could save on resources while still be able to afford baseline security for their crucial web application, which may contain very sensitive financial information about their clients and the company itself**.

Keywords - Telegram, bot, Web Application Firewall, web hacking, security

I. INTRODUCTION

1. As business grows and demand increased, many organizations had taken their business online where their reach could be more prevalent and cost-effective. Unfortunately, this also open to a new attack surface where cyber criminals are always

probing for sensitive personal information among other things to capitalize on. Report by [1] had placed the retail sector as the number one breached sector by number of identities exposed in 2014, which cost the sector 205 million in revenue. Web application based attacks is one of the most common and increasingly popular attack surface on many organizations. A successfully mounted attack could potentially disclose sensitive information.

The OWASP Top 10 List is a popular, frequently cited source for classifying the types of web application vulnerability and their criticalness degree to an organization even by various government organizations all over the world, [2]. This provides a solid base to justify the selections of vulnerabilities that will be covered by the Web Application Firewall (WAF) since it has sufficient real world merits and can be further supported by statistics from numerous quarterly reports by security companies all around the world. Furthermore, the notification system build was not very user friendly to operate. It takes more time and technical knowledge to learn the all the functionality as its design to be operated by experienced security professional and as such the learning curve is steeper compared to a generic WAF.

The research aims to develop a WAF with basic protection features that integrates Telegram instant notification system to enable the webmasters to easily and quickly receive notification on ongoing attacks on their websites as well as manage the WAF through the Telegram application itself. This integration features are deemed more user friendly and easier to use.

II. RELATED WORKS

A. Web Application Vulnerabilities.

The Open Web Application Security Project (OWASP) is a worldwide nonprofit educational charity dedicated to educate the masses about the risk and vulnerabilities faced by web application and its criticalness degree to an organization.

- SQL injection is the vulnerability that exist due to the improper sanitation of data from users that will be used to query the database. According to [3], malicious users could craft input from various functionality (Search, contact form, etc.) so that once it fused with the original query, it will execute those query according to the intend of the attacker, potentially revealing crucial data from the database. SQL injection is placed first on the latest OWASP Top 10 List (2013) under the subset of "Injections" type vulnerability and can be considered one of the most popular and dangerous types of cyber threat faced by companies especially small and medium sized enterprises.

Earlier, a research by [4] found that almost half of respondents (49 percent) say the SQL injection threat facing their company is very significant and on average, respondents believe 42 percent of all data breaches are due, at least in part, to SQL injections. In 2013 data breach of some of the largest payment processing companies, retailers, and financial institution by four hackers who then sold those data which consist of usernames and passwords, personal identification information and 160 million credit and debit card numbers, [5].

- Cross-site Scripting is a type of attack which exploits an application's output function that references poorly sanitized user input**.** An attacker could craft a malicious client-side scripting code such as JavaScript and feed it to an application that then stored it in its database (persistent XSS) or return that code back to the browser as a response (non-persistent XSS). This vulnerability could lead to session hijacking in which the attacker could send the malicious encoded Uniform Resource Locator (URL) to victims and the victim would be forced to give up his/her session cookies to the attacker. A successfully mounted attack could allow the attacker to assume the identity of the victim and stole the victim's information on the database. In the case of the administrator account being compromised, the effect will be much greater, as the administrator had broader authorization upon other users.

  According to security experts, cross-site scripting is one of the most common threats in web applications. Many major websites including internet giants such as Google, Yahoo, Facebook, eBay, Twitter and even security company such as McAfee had had their fair share of XSS problems in the past. [6] cited that in 2010, XSS ranked first in the Mitre Common Weakness Enumeration (CWE) and third in the OWASP Top Ten list in 2013.
- File inclusions in general, is caused by the insufficient restriction and control over user input that determines what files will be included inside a dynamically generated HTML page. This in turn enable and attacker to steal files inside the web server which contains sensitive information about the website or even other visitors. File inclusion attacks can be separated into two distinct classes. Local File Inclusion (LFI) is Remote File Inclusions (RFI). As stated by [7], RFI attacks take advantage of vulnerable PHP Web application parameters by including a URL reference to remotely host

malicious code which in turn embed the malicious code inside the generated webpage.

The impact of this vulnerability range from remote code execution to the complete compromise of the target web server. LFI attacks on the other hand works almost exactly as LFI except that the URL referenced is that of the web server own address or files instead of a remote URL. This significantly reduces the possibility of remote code execution but still can cause major breach as the attacker could view critical files in the server. Referring to Table 1, [7] concluded that although the frequency of LFI/RFI attacks is not as high as SQLi and XSS, but it does bring up quite a significant number on attack magnitude on particular incidents which takes into account the number of request and attack duration on any given attack.

Table 1 Distribution of Magnitude of Attack Incidents [7]

	SQLI	RFI	LFI	DT	XSS	HTTP	Spam
Median	81	961	46	91	95	67	41
Max	57235	199545	874	47555	12012	38341	1253
1st Quartile	42	169	36	49	38	41	32
3rd Quartile	160	1092	65	274	412	148	59

- Directory Brute Force attack is a means of gathering information located on the target web server by bombarding the server with request that is generated by either permutation of alphanumeric characters or based on a set of prepared wordlist. Its goal is to obtain some hits on the arrays of request made and reveal files and directories that could be entry point of the attacker. Some of the information that could be obtains are software installation directories and software version, backup files such as MySQL (.sql) and source backup (.bak) and internal documents (reports, memos). Directory Brute Force attacks as it's one of the easiest ways an attacker can use to disclose information about the company.

B. Web Application Firewall

Web technologies are constantly evolving and for the most part consist of many moving parts that made up a single functioning system to archive a set of goals. These moving parts are usually referred collectively as 'stacks' in the web application development community where a particular system could be assembled by different combination of component to make one complete system stack. These components serve different purposes on the system. For example, MySQL is one of the popular back-end database while PHP is the language used for processing data and generating dynamic content. On the server side, apache2 is commonly used as the web server software where it supports web-related protocols (HTTP, HTTPS, SMTP and etc.) so that user could reach services on their browser. Most of the times bugs and vulnerabilities will happen on and between these components and here lies the importance of WAF that can monitor

these components to catch bugs and vulnerability that a hardware firewall simply unable to.

C. Attack Detection Method.

The need for WAF arise when there is a need for webmasters to monitor attacks that is sophisticated and web application-centric. These attacks are 'invincible' to a regular hardware-based firewall which operates by inspecting packets size and signatures of known attacks on much lower transport layer. While this approach had been proven to be working for network related attacks, web applications attack proves to be more complex to deal with on a daily basis. A WAF is an application level (layer 7) software that is used to control input, output and access to and from applications that is running on a web server [12] by employing various approaches in blocking and filtering content. According to the Web Application Firewall Evaluation Criteria document published by the OWASP organization [16] some of the prevalent model for the WAF are the negative security model (blacklisting) and positive security model (whitelisting) which can be divided into two implementations that is the signature-based (also known as pattern based) and Rule-based.

- Rule based implementation allows more complex logic to be constructed and compared based on a predefined list of logic sequence. It allows for request filtering based on a user behavior sequence. And much like the signature based method, its effectiveness rely highly on the quality of the wordlist or filter list.
- Signature based implementation detect attacks by comparing the incoming request with a predefined list of known attacks pattern or attacks payload and is only as effective as the list itself. Webmasters need to update their list on a regular basis to increase the effectiveness of filtering dangerous request.

The research project will be utilizing the signature-based blacklisting / negative approach and will introduce some of the WAF evading technique pattern to be added to the predefined filter list as well as to put emphasize on the fresh look of integrating the management interface of the WAF via Telegram Bot. To enhance the effectiveness of the list, there will also be an analysis on the commonly used technique to bypass WAF filtering so that a countermeasure wordlist could be produced to counter the bypassing method. One this particular project, the blacklisting model will be implemented due to several constraints and justification.

D. User input sanitization method.

Every input from the user or any input to the system for that matter needs to be checked and escaped from any character sequence that could potentially lead to it being maliciously interpreted by the respective programs processing those inputs. The job of preventing those vulnerabilities will be passed on to the

request filtering mechanism, which decides what request to block and what is allowed to go through.

- PHP htmlentities. From [8], the htmlentities function in PHP convert all characters that have HTML character entity equivalents into their respective equivalents. This will effectively prevent browsers from interpreting the HTML characters and thus prevent some variant of XSS injection attack.
  Code example:
  ```
  $input = $_POST[‘input’];
  $clean_Input=
    htmlentities($input);
  echo $clean_Input;

  Input:
    <script>alert(‘XSS’)</script>
  Output:
    &lt;script&gt;alert(&#039;XSS&#039;)&lt;/script&gt;&lt;
  ```
- **PHP htmlspecialchars.** Functions almost exactly as the htmlentities function but differes in the amount of characters converted. Htmlspecialchars will only convert some characters to its HTML characters’ entities while htmlentities will convert all characters that is passed on to it.
  Code example:
  ```
  $input = $_POST[‘input’];
  $clean_Input=htmlentities($inp
  ut);
  echo $clean_Input;

  Input:
  <script>alert(‘XSS’)</script>
  Output:
  &lt;script&gt;alert('XSS')&lt;/s
  cript&gt;v
  ```
- **PHP mysql_real_escape_string.** The mysql_real_escape_string works by escaping special characters from the string that will be used to construct a SQL statement, [9]. This gives a good baseline filtering for commonly seen characters used by attackers during SQL injection attack such as double qoutes, single quotes, the less than “<” and greater than “>” character.
  Code Example:
  ```
  $clean_name=
  mysql_real_escape_string($_P
  OST['username']);
  $clean_pass =
  mysql_real_escape_string($_P
  OST['password']);
  $query = "SELECT * FROM
  users WHERE
  user=’$clean_name’ AND
  password=’$clean_pass’”;
  echo $query;

  Input  : X or '1'='1
  Output : X or \'1\'=\'1
  ```

III. PROTOTYPE SYSTEM

A. Telegram Bot

Telegram bot is the interface/account that enables user to integrate telegram (a popular internet-based instant messaging system) functionality on their programs or applications. Essentially it is a separate account that telegram user could create to

be used to create application that automate task in which the user could control the account (send files, messages, receive files, messages etc.) using only the telegram API calls based on the logic of their application. Telegram bot are, as the official documentation says: special accounts that do not require an additional phone number to set up. These accounts serve as an interface for code running somewhere on your server."[10]

B. Telegram Bot API

The Application Programming Interface (API) allows Telegram users to connect their programs to the telegram server in which acts as a meditiary between the end user of the program and the program itself. The flow of communication could be explained with the diagram below using the previous football trivia bot as an example:

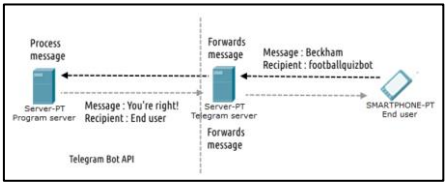

Fig. 1: Telegram communication flow though the Telegram bot API (Telegram APIs, 2016)

Telegram server acts as a middle-man that forward communication between the end user and the program on the server and the process of data forwarding from the Telegram server to the program server and vice versa is formatted using the API The dynamics of the API ranges some basic text messaging functions to a full fledge file sharing functionality that handles almost every types of file, much like what you could do as a normal Telegram user. One could also argue that this much support for file sharing had transform Telegram into a 'Social File Sharing' service. In fact, there had been bots being developed to do just that: provide the end user spaces to store and manage all their files on the bot's account on behalf of the user.

By default, the API response is in the format of a JSON object with a mandatory 'ok' Boolean field. It supports GET and POST method and four ways of passing parameters in bot API request, which are:

- URL query string
- Application/x-www-form-urlencoded
- Application/json
- Multipart/form-data

Core Telegram Bot API's currently available for developers and users alike.

- The unique authentication token format
- Fetching updates/messages
- Testing for authentication token
- Sending messages

C. Use Case

Figure 2 illustrates the features of the system including some core modules that is inside the Telegram client. A user or webmaster will be able to control the WAF from watching the Telegram client once WAF is fully integrated with the target system. Some of these features, include the ability to ban a user from accessing the website or system based on the user IP address, ability to do a WHOIS query to determine the information bound to a particular ip

address, the ability to do a file integrity scanning via hashing (md5) to ensure the integrity of files within the protected system and also the basic functionality of querying attacks statistics logged by the WAF

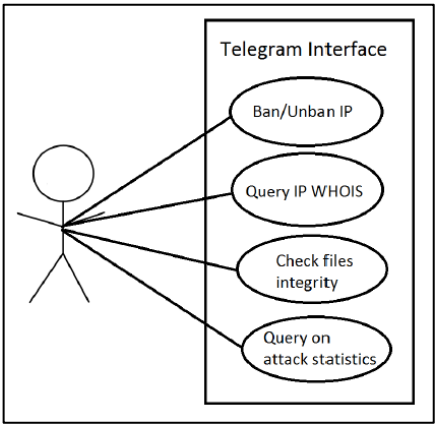

Fig. 2: Use case diagram on the Telegram User Interface

D. Graphical User Interface (GUI) Design

Since the Management of the WAF is done via Telegram, naturally the interface would be same as a regular mobile Telegram client. The user or webmaster could send various format of command to change the settings and interact with the WAF.

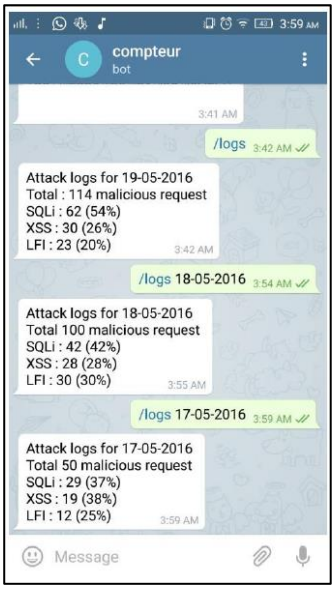

Fig. 3: Real-time attack notification

Fig. 4: IP-based access control

Fig. 5: IP WHOIS Query

Fig. 6: File Integrity Checking

The Management of the WAF is done via Telegram thus; the interface would be same as a regular mobile Telegram client. The user or the webmaster could send various format of command to change the settings and interact with the WAF. Figure 3, 4, 5 and 6 shows the interface of managing WAF through telegram.

IV. Result and Conclusion

This research project is utilizing the signature-based blacklisting approach and some of the WAF evading technique pattern has been added to the predefined filter list as well as to put emphasize on the fresh look of integrating the management interface of the WAF via Telegram Bot. To enhance the effectiveness of the list, an analysis on the commonly used technique to bypass WAF filtering is done and as result, a countermeasure wordlist is produced to counter the bypassing method. In this project, the blacklisting model is implemented.

Security testing has been done and the result is as shown in Table 2.

Table 2 Security testing results of the WAF

Vulnerability	Tools used	Expected outcome	Actual outcome	Score
SQLi	SQLMap	Detect attack, stop, log and send notification	Detect attack, stop, log and send notification	4/4
XSS	XSSer	Detect attack, stop, log and send notification	Attack detected, stopped and logged	3/4
LFI/RFI	Fimap	Detect attack, stop, log and send notification	Detect attack, stop, log and send notification	4/4
Directory Bruteforce	Custom script	Detect attack, stop, log and send notification	Attack detected, stopped, logged and notification sent	4/4

V. REFERENCE

[1] 2015 Internet Security Threat Report. Retrieved March 04, 2016, from Symantec website: https://www.symantec.com/content/en/us/enterprise/other_resources/21347933_GA_RPT-internet-security-threat-report-volume-20-2015.pdf, 2015

[2] Industry: Citations - OWASP. Owasp.org. Retrieved 8 April 2016, from https://www.owasp.org/index.php/Industry:Citations, 2016

[3] Sonewar, P., & Mhetre, A Novel Approach for Detection of SQL Injection and Cross Site Scripting Attacks. IEEE. Retrieved from http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=7087131, 2016

[4] The SQL Injection Threat Study. Retrieved from http://www.dbnetworks.com/pdf/ponemon-the-SQL-injection-threat-study.pdf, 2014

[5] Chak, S., Managing Cybersecurity as A Business Risk For Small And Medium Enterprises (Undergraduate). Johns Hopkins University, 2015

[6] Shar, L., & Tan, H., Defending against Cross-Site Scripting Attacks. Computer. http://dx.doi.org/10.1109/mc.2011.261, 2012, 55-62

[7] Web Application Attack Report (WAAR) (Rep.). (n.d.). Retrieved March 05, 2016, from Imperva website: https://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed6.pdf, 2015

[8] PHP: htmlentities – Manual Php.net. Retrieved 8 April 2016, from http://php.net/manual/en/function.htmlentities.php, 2016

[9] PHP: mysql_real_escape_string - Manual. Php.net. Retrieved 8 April 2016, from http://php.net/manual/en/function.mysql-real-escape-string.php, 2016

[10] Telegram APIs. Core.telegram.org. Retrieved 8 April 2016, from https://core.telegram.org/, 2016

[11] Part 1 Telegram Bot Tutorial – APIs and Webbooks. Fullmeter. Retreived from https://fullmeter.com/blog/?p=14, 2015

[12] M. A., Evading all Web-Application Firewalls XSS filters. Retrieved April 07, 2016, from https://www.mazinahmed.net/uploads/Evading All Web-Application Firewalls XSS Filters.pdf, 2015

[13] Tutorialspoint Functional Requirements. Retrieved April 07, 2016, from http://www.tutorialspoint.com/software_testing_dictionary/functional_requirements.htm

[14] Rosenblatt, H. J., Systems Analysis And Design. Boston: Course Technology, 2001

[15] Website Security Statistics Report 2015 (Rep.). (n.d.). Retrieved March 07, 2016, from WhiteHat Security website: https://info.whitehatsec.com/rs/whitehatsecurity/images/2015-Stats-Report.pdf, 2015

[16] OWASP. (2016). Owasp.org. Retrieved 8 April 2016, from https://www.owasp.org/index.php/Main_Page

[17] Valeur, F., Mutz, D., & Vigna, G., A Learning-Based Approach to the Detection of SQL Attacks. Detection of Intrusions and Malware, And Vulnerability Assessment, 2005, 123-140