Log Visualization of Intrusion and Prevention Reverse Proxy Server Against Web Attacks

Teddy Mantoro[1], Normaziah binti Abdul Aziz[2], Nur Dalilah binti Meor Yusoff[2], Nor Aishah binti Abu Talib[2]

[1]Faculty Science and Technology, Universitas Siswa Bangsa International, Jakarta, Indonesia.
[2]Department of Computer Science, KICT Technology, International Islamic University Malaysia, Kuala Lumpur, Malaysia.

***Abstract*— SQL Injection Attack (SQLIA) has made to the top of the OWASP, Top 10 Web Application Security Risks in 2013 and in 2010. The explosive use of web application with very little emphasis lay on securing it make this attack becoming more popular. Various methods have been discussed and proposed as countermeasure to the attack. Unfortunately, most of them are seen to be not comprehensive enough to address any kind of issues an organization might have when it comes to hardening the web security such as technical and financial matter for instance. This study presents a way to prevent and detect intrusion through the deployment of reverse proxy with an intrusion and prevention mechanism built in against web attacks especially SQLIA. With the flexibility offered in server logging process, we obtain and analyse preferred data to visualize the type of attack based on logs information. Our graph visualization development monitors three web security aspects, i.e. the top traffic blocked attempted by IP address, number of regular expression rules violated and detect the rules of intrusion detection.**

Keywords- SQL Injection Attack, Network Intrusion Detection , Firewall, ModSecurity, Reverse Proxy.

I. INTRODUCTION

Web applications have become the most prevailing way of delivering service to people today. Ranging from commercial businesses, corporate bodies to education institutions, web application is seen to be a convenient way to provide services due to its ubiquity. Unfortunately, putting a database on the Internet may actually danger an organization's security. The vulnerability of web applications invite threats and web attacks by malicious hackers who intend to access sensitive data in the databases.

SQL Injection attack (SQLIA) is one of the techniques used to attack databases through a website. This attack tries to gain access to sensitive data directly by injecting malicious SQL codes through web application. The wide spread of SQLIA is worrying. The concept of making databases to be publicly accessed has made this attack become a popular approach for adversary to compromise data. According to UK Security Breach Investigations Report on Analysis of Data Compromise Cases released by 7Safe, a whopping 60% of all breach incidents examined involved SQL injections [1]. This situation raises the need for a security mechanism that can prevent such vulnerability from being exploited by irresponsible parties.

Realizing the above fact, some organizations rely on firewall to do all the detection and prevention of attack. Unfortunately, the widespread deployment of firewalls and network intrusion detection systems (NIDSes) can only provide limited protection to the majority of databases used by web-based applications. Others on the other hand, choose to train the web developers and programmers on good programming practice and corrective measures. The awareness of security implications of source codes has been a known activities amongst programmers. Nonetheless, these approaches are not be enough to secure the legacy system since the process of revising and rewriting the entire existing code is not easy and cost-effective. The complexity of the web services available today is pretty challenging since it is difficult to encounter all possible exceptions to the expected behaviour of web servers and web applications.

An intense research has been going on to figure out the best security mechanism, for detecting and preventing SQLIA from happening. Among the known countermeasure are Web Framework, Prepared Statement, Static Analysis, Dynamic Analysis and combination of both static and dynamic analysis. Details on each technique will be reviewed in the related work section. Meanwhile, apart from those techniques, there is also an approach to employ a reverse proxy.

This paper studies a reverse proxy with an intrusion and prevention mechanism built in against web attacks like SQLIA. This method offers protection over servers residing in internal network while providing services to external client such as in [2]. A very good security mechanism, ModSecurity, provides a complete separated layer of security for the application level of web applications. ModSecurity can be deployed as part of the existing web server architecture without involving much rewriting or revising. A reverse proxy using ModSecurity is capable of handling access control for all internal Web servers. In short, any request sent by any clients will be forwarded to this reverse proxy first before being parsed to web servers for processing. Leveraging the location of where a proxy resides in network architecture, users come to learn that its function can actually be extended with performing some checking on the client request.

ModSecurity is an Apache module added to the reverse proxy. ModSecurity, a web application firewall engine, detects malicious request based on the rules it uses. This rule, namely the Core Rule can be customized by users, allowing them to analyse every aspect of request using regular expression. A good custom coded rule provides better detection of an SQL injection attack. Otherwise, a lot of attacks could bypass the checking undetected. Apart from parsing and checking requests, ModSecurity also logs the complete version of requests. Through logs, the web administrator will be able to monitor the network traffic for any anomaly or suspicious traffic. Unfortunately, analysing

logs could be very time-consuming and tiresome [2]. The log text messages make it almost hard for administrator to perform auditing since the text message could be extremely long and repeated message might occur in certain period of time [3].

The contribution of this study is the development of graph visualization in monitoring three web security aspects, i.e. the top traffic blocked attempted by IP address, number of ModSecurity rules violated and SQLIA rule intrusion detection (ID) detected. We discuss a method that helps the web administrator to analyse log files in more efficient way by transforming the data and information contains in the log files into a graphical representation. The visual transformation of data reduces the workload of the administrator in the sense that they can make analysis and comparison without having to read the log line by line.

The rest of this paper is organized as follows: Section 2 discusses some related work in the field of SQLIA detection and prevention. Section 3 describes the propose framework and development of our visualization system based on log files of any possible web attack. Section 4 demonstrates the evaluation of the framework together with the discussion of our result findings. The Section 5 present the conclusion of this study.

II. RELATED WORK

As mentioned earlier, various works have been done to protect web applications from SQL injection attack. This section will brief on SQL Injection detection technique, and also touches visualization of server logs.

A. SQL Injection Detection Mechanism

Several popular techniques in detecting an attack were being brief as follows:

1) Static Analysis: Static Analysis statically screens web application's source code for vulnerabilities. One of the research conducted used this approach to identify all points on the application code that issues SQL queries to underlying database. The researcher identified all the 'execute' methods of the Statement class in Java as vulnerable [4]. This method aims to identify SQLIA during to validate user input before being used to generate query [5].
2) Dynamic Analysis/Runtime Monitoring: This technique is the compliment of the Static Analysis, which is Dynamic Analysis or Runtime Monitoring. Unlike the static, this analysis locates vulnerabilities of SQLIA during runtime environment and eliminates the need to modify the web application codes. Since these two techniques are complementary, many countermeasures choose to combine both analyses to have a better defense mechanism [4, 6, 7, and 8].
3) Prepared Statement: Prepared statement is a fixed query "template" which is predefined explicitly, providing type-specific placeholders for input data [9, 10]. This method converts vulnerable SQL statements to prepared statements. The automation of prepared statement approach detects SQL injection vulnerability from the source code and creates a recommended secure statement code structure for it. This secure prepared statement will segregate the SQL statements from its input. An experiment done using this method has yielded a 94% of success in removing SQL Injection Vulnerability.

Table 1 show the comparison of the techniques used in various prevention and detection tools for attack available today. The comparison was against five variables; detection accuracy, false positive, removal of vulnerability, overhead or processing delay and deployment complexity. Deployment complexity includes needless to modify underlying source code and ability to be employ for legacy system.

TABLE I. COMPARISON OF SQLIA TOOLS

Technique	Ref.	Variable				
		Detection accuracy	False positive	removal of vulnerability	Overhead/ processing delay	Deployment complexity
Static Analysis + Runtime Monitoring	[8]	Yes	0%	N/A	Reduced	N/A
	[4]	Yes	N/A	N/A	Reduced	Low
Proxy server	[5]	Yes	N/A	N/A	16.7% to 25.7%	Low
Prepared statements	[9]	No	N/A	94%	N/A	N/A
Reverse Proxy + Anomaly	[11]	Yes	24 were served	N/A	N/A	Low
	[12]	Yes	90% reduced	N/A	N/A	Low

B. Reverse Proxy with ModSecurity

Employing reverse proxy for web security is one of the server-side solutions. It can be generalized that most of the common mechanism of a server side solution is to use these proxies as application-firewalls to filter out malicious code or injected request in the case of SQL Injection Attack. Some researchers argue the additional overhead associated with the checking process done by server side solutions may result in performance degradation [13]. On the other hand, reverse proxies provide a complete separated layer of security for the application level of web applications. They are capable of terminating TCP and SSL protocols besides controlling TCP and SSL handshakes to the clients as well as to the servers[14]. In short, no packets flow to the back-end until it is inspected and validated by reverse proxies. Consequently, any bad request like an injected code can never pass to the real web servers. Apart from the network isolation concept, this approach is also very effective if there are several web servers to protect. For all web servers, one proxy may enforce all the access controls needed accordingly.

It is important for these application-firewalls to be installed with effective intrusion detection and prevention module that can detect and prevent SQL injection attacks as much as it can without producing high false positive case. ModSecurity is an Apache module that functions closely as any intrusion detection systems (IDS). It works by implementing a comprehensive set of rules to perform analysis on every aspect of request like its header or on the response's body to prevent information leaks. Several

functionalities that make people choose ModSecurity configured in a reverse proxy are as follows [15]:

1) Real Time Monitoring and Attack Detection. ModSecurity monitors HTTP traffic in real time to assist attack detection. Similar to other web intrusion detection tool, necessary action is taken prior to the detection of suspicious event.
2) Attack Prevention and Patching. ModSecurity prevents attack from reaching the web application immediately.
3) Flexible Rule Engine. ModSecurity allows custom rules to come into action. It makes the common operations simple and complex operation possible through combinations of several rules.
4) HTTP Traffic Logging. ModSecurity logs each of HTTP session; both requests and responses based on the user's preferences on the relevant of each data providing a fine granulated filtering

With regards to the performance overhead argued earlier, it can be minimized to an acceptable amount, if is configured properly [2]. Thus, the purpose of ModSecurity is to increase web application security by protecting them from known and unknown attacks [16]. Besides offering an implicit load balancing and scaling, a reverse proxy with ModSecurity might be the best way for organizations that have legacy systems to protect.

C. Visualization of Server Logs

Web servers normally are capable of logging traffic in a useful form for marketing analyses, but somehow fail to do so to web applications; especially when it comes to logging the request bodies. That is why most attacks today are performed via POST requests and rendering the systems blind [15]. With HTTP traffic logging, ModSecurity has the mechanism to perfectly collect logs for traffics coming in and out from the server. Nevertheless, having the log data only without knowing how to transform it into useful security information is meaningless. From the log, a web security administrator can really understand what is happening to the network, uncover hidden pattern, thus identify and respond to the attacks accordingly.

On the other hand, to analyse each entry in a log file is really tiresome due to its big file in size and unpresentable feature. Several researchers have seen that one way to analyse the logs is through visualization [17, 18, 19]. Through visualization, any complex protocol or other security-related ideas can be well-depict. Not to mention how helpful a visualized log will be in providing the analyser with better intuition while reducing their workload. Besides that, security analysts will be able to explore logs that are maintained and updated by security protocols. Since manual manipulation of these logs can be cumbersome and ineffective, this paper proposes a method to visualize majority attach for web server based on the web server log. This visualization is a very important tool in securing web application, which we adopted from [28].

III. PROPOSED FRAMEWORK

The generic system architecture is illustrated in Figure 1. It has a reverse proxy resides in between client and the web server hosting the applications. The internal servers could be hosting database application for data storage. For every request that came in, reverse proxy the following step:

- Parse the request.
- Perform canonization and anti-evasion actions to transform the input into a form that is suitable for analysis. Evasive technique might be used includes multiple slash character.
- Perform special built in checks which consists of complicated validations such as URL encoding.
- Execute the input rules.
- The request is then allowed to pass to the respective web server. Upon the output responded, proxy will:
- Execute output rule to prevent any attempt of information leaks.
- Log the request.

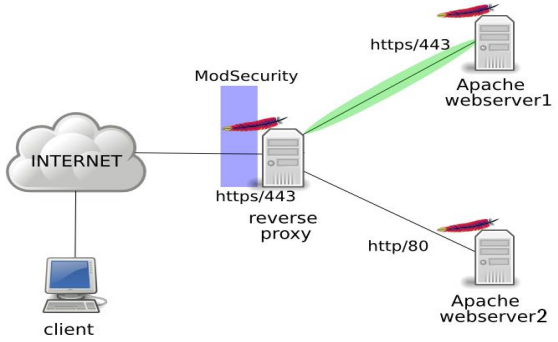

Figure 1. The generic intrusion detection proxy server system architecture.

Referring back to the ModSecurity functionality on logging HTTP session, ones can actually specify what data to be logged in depending to its relevance. Our system has been configured to only log for request that triggers a ModSecurity response. The log files can be found in directory etc/apache2/logs namely modsec_audit.log. Graphs representing the information from log are then visualized accordingly.

IV. RESULT AND DISCUSSION

The environment for the proposed framework was setup to demonstrate the discussed technique. Following the system architecture specified above, we configure one Apache server to be in reverse proxy mode with ModSecurity installed using Core ModSecurity Rule Set (currently ver. 2.2.4 is used). We had one internal server, hosting web and database application within it. Then, we monitored the network activity through log visualization for seven weeks.

There are 16 base rules of core rules in our experiment. Figure 2 highlights top four rules detected during the evaluation. It shows that SQLIA is the overall highest detected rules in all seven weeks.

There are 11 different IP addresses that is recorded having an attempt to attack the web application. Figure 3 visualizes the top three attacker's IP address where

192.168.42.195 has the highest blocked attempt. Administrator can use the IP address to track location of the offender and also block the IP address from accessing the web server.

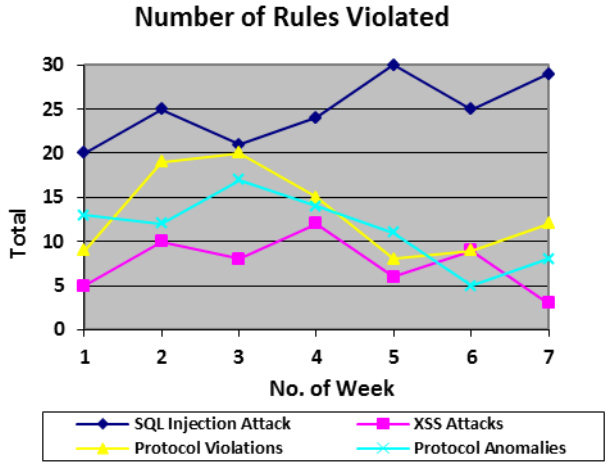

Figure 2. Number of ModSecurity rules violated per week.

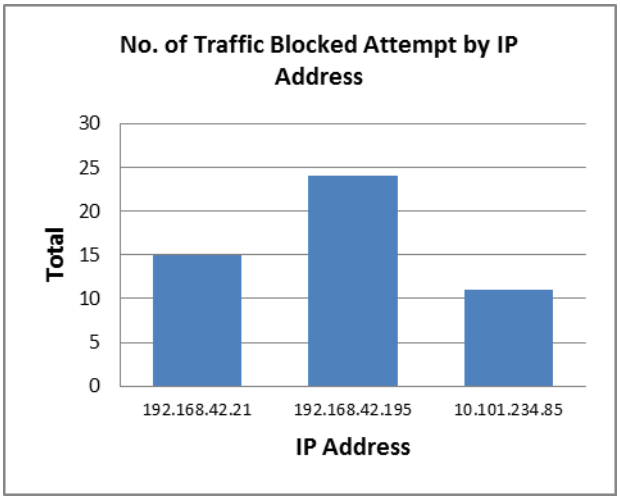

Figure 3. Number of Traffic Blocked Attempt by IP Address.

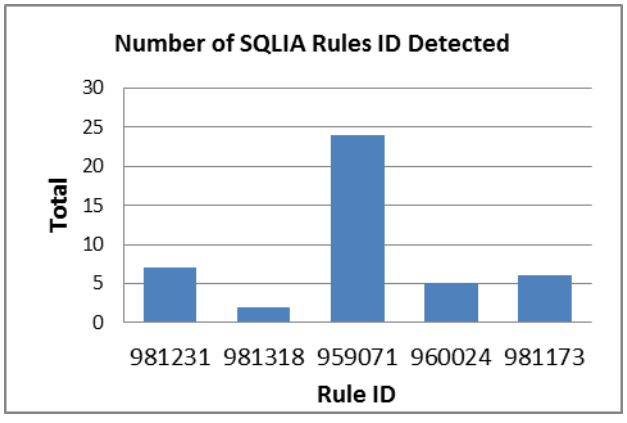

Figure 4. Number of SQLIA Rule ID Detected by ModSecurity.

As one of the objectives of this study is to detect and prevent SQLIA. Thus, we only concentrate on the visualization of SQLIA rules ID. In the rules used, there are currently 37 rule IDs associated with SQLIA rule configuration file. The bar chart in Figure 4 illustrates the top five rule IDs that is detected by our ModSecurity proxy server. Rule ID 959071 depicted as the most matched SQLIA rule ID which it identifies a SQLIA pattern based on quote characters that attackers insert or append to string statement ending. With the visualization of rule ID, it will not only help to identify the most common pattern that malicious user use to attack a web application but also help administrator to identify and debug any false positive rule.

V. CONCLUSIONS

This paper highllights SQLIAs and how the combination of a reverse proxy with ModSecurity can help organizations to detect and block SQLIA in efficient manner.

As proof of concept, an experiment has been conducted to evaluate the proposed framework in protecting an internal web server for seven weeks. We then analysed and visualized the log files of the reverse proxy. We developed three graphs for visualization in monitoring the top traffic blocked attempted by IP address, number of ModSecurity rules violated per week and SQLIA rule ID detected. Based on our seven weeks experiment, it shows that SQLIA is the overall highest detected rules. This is similar to the OWASP Top 10 Web Application Security Risks for 2010 where SQLIA comes out on top of the list [20].

Based on the analysis of those three graphs, which representing meaningful information through visualisation, the organizations can utilize it for hardening their security.

As for future work, more visualisation will be developed dynamically to support most OWASP Top 10 Web Application Security Risks in years to come.

REFERENCES

[1] UK Security Breach Investigations Report 2010 Published (2010) Retrieved from http://www.corporate.7safe.com/uk-security-breach-investigations-report-2010-published-2. Accessed on 10 April 2012.

[2] Ristic, I. (2010). Web Security Appliance with Apache and mod_security Retrieved from http://www.symantec.com/connect/articles/web-security-appliance-apache-and- modsecurity. Accessed on 10 April 2012.

[3] Koike, H., & Ohno, K. (2004). SnortView: visualization system of snort logs. The proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, pp. 143-147..

[4] Muthuprasanna, M., Wei, K., & Kothari, S. (2006). Eliminating SQL Injection Attacks - A Transparent Defense Mechanism. 2006 Eighth IEEE International Symposium on Web Site Evolution WSE06, pp. 22-32.

[5] Liu, A., Yuan, Y., Wijesekera, D., & Stavrou, A. (2009). SQLProb : A Proxy-based Architecture towards Preventing SQL Injection Attacks. System, pp. 2054-2061.

[6] Lam, M. S., Martin, M., Livshits, B., & Whaley, J. (2008). Securing web applications with static and dynamic information flow tracking. Proceedings of the 2008 ACM SIGPLAN symposium on Partial evaluation and semantics based program manipulation PEPM 08, pp. 3-12.

[7] Kim, J.G. (2011) Injection Attack Detection Using the Removal of SQL Query Attribute Values, International Conference on Information Science and Applications (ICISA) pp. 1-7, 26-29 April 2011.

[8] Fernando, H. and Abawajy, J.H. (2011). Securing RFID Systems from SQLIA. ICA3PP (2) 2011, pp. 245-254.

[9] Thomas, S., Williams, L., & Xie, T. (2009). On automated prepared statement generation to remove SQL injection vulnerabilities. Information and Software Technology, 51(3), pp. 589-598.

[10] Amirtahmasebi, K., Jalalinia, S. R., & Khadem, S. (2009). A survey of SQL injection defense mechanisms. Internet Technology and Secured Transactions 2009 ICITST 2009 International Conference for (p. 1–8).

[11] Valeur, F., Vigna, G., Kruegel, C. & Kirda, E. (2006), "An anomaly-driven reverse proxy for web applications". In Proc. of the 2006 ACM symposium on Applied computing, pages 361-368.

[12] Vigna, G. et al. (2009), "Reducing Errors in the Anomaly-based Detection of Web-based Attacks Through the Combined Analysis of Web Requests and SQL Queries", Journal of Computer Security, Vol. 17, pp. 305-329.

[13] Ismail, O., Etoh, M., & Kadobayashi, Y. (2004). A proposal and implementation of automatic detection/collection system for cross-site scripting vulnerability. IEEE 18th International Conference on Advanced Information Networking and Applications 2004 AINA 2004 (pp. 145-151).

[14] Moosa, A., & Alsaffar, E. M. (2008). Proposing a hybrid-intelligent framework to secure e-government web applications. Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance, 52-59.

[15] ModSecurity: Overview (n.d). Retrieved from http://www.modsecurity.org/projects/ modsecurity/apache/. Accessed on 10 April 2012.

[16] Bleda F.D. and Martorella C (2005). Advanced Web Application Defense with ModSecurity. Retrieved from http://www.isecauditors.com/downloads/present/ WhatTheHack_2005_modsecurity.pdf. Accessed on 10 April 2012.

[17] Marty, R. (2008) Security Visualization. Retrieved from http://secviz.org/content/ applied-security-visualization. Accessed on 10 April 2012.

[18] Lamagna, W. M. (2011). An Integrated Visualization on Network Events VAST 2011 Mini Challenge # 2 Award: "Outstanding Integrated Overview Display. Challenge, pp. 317-319.

[19] Tamassia, R., Palazzi, B., & Papamanthou, C. (1995). Graph Drawing for Security Visualization. (L. G. Tollis & M. Patrignani, Eds.) Computer, 21 (December), 1-20. Springer Berlin Heidelberg.

[20] OWASP (2011). OWASP Top 10 2010. Retrieved from OWASP Top 10. Accessed on 10 April 2012.