Int. J. of Advanced Networking and Applications
Volume: 01, Issue: 05, Pages: 314-317 (2010)

Performance Analysis of Internal vs. External Security Mechanism in Web Applications

Supriya Gupta
Dept. of Computer Sc. & IT, University of Jammu, Jammu-180 006, India
Email: mangotwin22@gmail.com

Dr. Lalitsen Sharma
Dept. of Computer Sc. & IT, University of Jammu, Jammu-180 006, India
Email: lalitsen@yahoo.com

ABSTRACT

Most of the applications now-a-days are developed web based. The applications of public access are highly exposed to security threats. The increasing number of web based attacks which result in loss of data and unauthorized access to application has drawn the attention of organizations toward web application security. The most commonly employed defense mechanism is to use solutions that rely on security service tools like firewalls, intrusion detection and prevention systems etc. Most of the commonly used tools such as SNORT are based upon the payload inspection that detects an attack by searching for the occurrence of known signature patterns in the packet. But using these devices for protecting web applications against common input based attacks is an inefficient process. It consumes a significant amount of time, memory and CPU cycles for each packet while scanning through a list of rules. Implementing security features within applications' logic is an effective alternative. In this paper we analyzed the performance of two experimental web applications, one with security implemented within the code and the other checked by external security system called SNORT using a web application testing tool (WAPT 3.0). Our experiment showed that the application with secure code showed better performance statistics in terms of response time. The paper also discusses various issues regarding the use of security devices as protection against application layer attacks.

Keywords - webapplications, security, intrusion detection and prevention, snort

Date of Submission: Jan 04, 2010 Accepted: March 03, 2010

1. INTRODUCTION

The ubiquity and popularity of World Wide Web has attracted the developers to develop the applications web based. In a competition to develop online services for general public, web applications have often been deployed with minimal attention to security risks, as a result most applications are surprisingly vulnerable to attacks [1]. According to a recent report[1], SQL injection, cross-site scripting, and buffer overflow were the most prevalent of the application layer attacks encountered in 2008. These attacks are mostly the result of weak input validation and can be checked by using secure coding practices . Secure coding is about implementing security functions like input validation, sanitization, and exception handling etc. within applications' logic so that the application becomes resilient to malicious attacks. This is here in called Internal Security. However, in most organizations, it is difficult to ensure that all application developers are adequately trained in secure coding practices and keep updated on new vulnerabilities. The solutions used often rely on external security products, which include application firewalls [2], intrusion detection and prevention systems etc. that improve security by blocking application hacking techniques. Securing applications using these devices is here in called External Security. Most commonly used devices are the deep packet inspection systems such as SNORT that look within the application payload of a packet or traffic stream and make decisions based on the content of that data.

Snort[2] is an open source network intrusion detection and prevention system (IDS/IPS) developed by Sourcefire. It captures the data packets traveling on the network media (cables, wireless) and matches them to a database of known attack signatures. Depending upon whether a packet is matched with a signature, an alert is generated and the packet is logged to a file or database. The signatures of vulnerabilities and malicious activities are represented as a set of rules in a standard industry format used by security professionals worldwide. Besides string based matching for the identification of malicious signatures SNORT utilizes PCRE (Perl Compatible Regular Expression) engine for regular expression based matching in a packet payload [3]. Using PCRE any generic or concise signatures that cover a particular application can be written to detect certain types of SQL injection and cross-site scripting attacks as they occur. It can run on the web server itself or on another computer within that same network and with the right rule-set very few attacks stay undetected. But these devices take up a considerable amount of time, memory and CPU cycles. The packets are first grabbed off the wire,

[1] Web Application Security Trends Report, Q3-Q4 2008 Cenzic, Inc. at http://www.cenzic.com/downloads/Cenzic_AppSecTrends_Q3-Q4-2008.pdf

[2] http://www.snort.org.

stored in some type of data structure. All of these data structures need to be cleared out to make room for other packets. Various pattern matching algorithms are then used for content analysis [4].

In this paper we analyzed the performance of two applications one with internal security and the other protected with an external security system, SNORT, open source network intrusion prevention and detection system. For this purpose two similar web applications were developed. One of the applications was embedded with security functions which can provide proper input validation in order to protect the application from the commonly known SQL injection and cross-site scripting attacks. And the other application was protected with SNORT. SNORT was also embedded with the rules to counter similar attacks. Response time of both the setups was noted. By comparing the response times obtained significant results could be estimated regarding whether secure coding or security service tools meet better performance requirements.

2. EXPERIMENTAL SETUP

Two applications were developed for searching data from a database of around 1500 records after taking input from the user. The experimental setup is shown in figure 1.

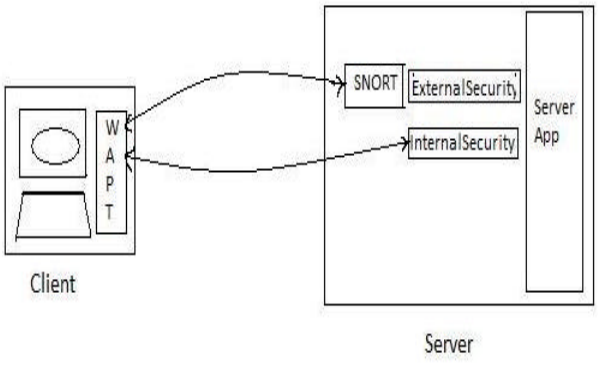

Figure 1. Experimental Setup

In one of the applications named, Internalsecurity, a special code was placed in application's source code so that every time a user entered the input (roll number or name), it is checked for malicious attacks and the inputs were processed only if the request was found to be valid. A portion of sample code is listed below which detects any occurrence of following SQL meta-characters <, >, =, ?, ;, -, (,), ', &, #,+,%, * in the input string and block all input strings containing one or more occurrences of any one of these characters. This approach is known as negative validation.

```
Protected Sub search1_Click(ByVal sender As Object,
ByVal e As System.EventArgs) Handles search1.Click

    Dim nm As String = TextBox1.Text

IfRegex.IsMatch(nm,"(<|>|=|\?|;|-|\)|\(|&|#|\+|\*|'|%)+") Then

 error1.Text = "Invalid character."

Else

Response.Redirect("result1.aspx?rno="&TextBox1.Text)"

End If

End Sub
```

Figure 2. Sample code of Internal security application which checks the input variable 'nm' against a regular expression.

The other application, Externalsecurity, was developed to provide the similar functionality but with the difference that the application was protected with SNORT to detect same set of malicious inputs. SNORT Version 2.8.4-ODBC-MySQL-FlexRESP-WIN32 (Build 26) [3] was used. Snort required some software packages before installation like packet capturing software, Perl Compatible Regular Expression library etc.

WinPcap 4.02, an open source packet capturing software available at www.winpcap.org and PCRE 7.4 which is also an open source library available at http://www.pcre.org/ were used. To host web applications IIS 7 was used. SNORT was installed on the machine where the web applications were hosted and configured to run in network intrusion detection mode. In this mode it doesn't record all packets but only the packets that triggered rules specified in "snort.conf". Also the default rule set in SNORT did not contain signatures for detecting cross-site scripting and SQL injection attacks, but those rules were extended to watch out for any occurrence of SQL meta-characters such as the single-quote, semi-colon or double-dash and angled brackets that signify HTML tag to avoid CSS attacks. A Sample rule is listed that contains hex-encoded values of meta-characters <, >, =, ?, ;, -, (,) ,+, /, #, %, &, *.

```
alert tcp 192.168.0.122 any -> 192.168.0.1 80
(msg:"sql
injection";pcre:"/(%3C)|(%3E)|(%3D)|(%3F)|(%3B)|
(%2D)(%2D)|(%2B)|(%2A)|(%29)|(%28)|(%27)|(%2
3)|(%26)/"; sid:7214;)
```

Figure 3. Sample SNORT rule.

Finally to evaluate the performance of the applications we used web applications testing tool (WAPT). WAPT 3.0 was

[3] By Martin Roesch & the Snort Team: http://www.snort.org/team.html, Copyright (C) 1998-2009 Sourcefire, Inc., et al

Volume: 01, Issue: 05, Pages: 314-317 (2010)

installed on client machine from where the request to applications was made. Both the applications were then tested through WAPT 3.0 and the following section presents the results associated with security mechanisms that we used.

3. RESULTS AND DISCUSSION

The test was conducted to evaluate the performance of two experimental web applications developed: 1) Internalsecurity, 2) Externalsecurity. WAPT was configured to simulate the test for twenty virtual users that perform a batch run from 1 to 20 in step of 1 with thirty iterations performed by each virtual user. The performance metric used was the response time.

First the server was protected with SNORT and a request for the application Externalsecurity was made. Then SNORT was terminated and a request for the application Internalsecurity was made. Both these applications were assessed for the average response time encountered by virtual users.

Figure 3 presents the average response time of the two applications as recorded by our testing tool.

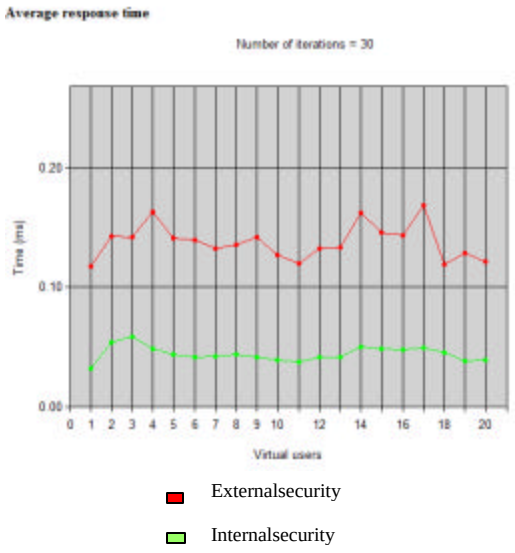

Figure 4. Average response time of the applications as measured by WAPT.

It was found that the application with secure code showed better performance as compared to the application protected with SNORT. The excessive response time in case of the application Externalsecurity can be attributed to the latency introduced by SNORT. There are four main areas in SNORT that consume considerable amount of time: getting packets off the wire, clearing out data structures, pattern matching and checksum verification. It must perform these tasks for every packet that comes across its interface. The growing number of the vulnerabilities has led to an ever growing number of rules in the SNORT ruleset. As a result, the SNORT IDS ends up with using increasing number of CPU cycles for each payload while scanning through the list of rules [5]. Also, these systems have a finite capacity queue [6], which means they have a buffer which can store a finite number of packets, and when this buffer fills up, further packets are discarded rather than processed. And it causes a major dropout in performance during high speeds of internal networks.

Another drawback with using these devices is that these attack signatures can be applied only to situations in which the context of the event is not important. For example, in the SNORT ruleset that we used, we added a simple string matching signature that triggered an alert action whenever the traffic that it was analyzing, contained **'<, > or ='**. When this simple string signature was applied to monitor TCP traffic, the alerts were generated even when those characters were valid for some part of application. As a result the alerts generated by SNORT were the false one as show in Figure 4 and definitely slow down the performance of the application.

Figure 5. False alert generated by snort.

Moreover the approach, used to prevent cross-site scripting and SQL injection attacks, was based on blocking certain possible malicious characters in the packet payload. This is known as negative validation. It can defend against specific known attacks but it is very difficult to define all possible malicious inputs. The best practice recommended for input validation is to provide a list of valid inputs so that only valid inputs are allowed and rest all is blocked. This approach is known as positive validation. Positive validation approach cannot be used in deep packet inspection systems because

using this approach in these devices will result in large number of non-contextual alerts that would prevent these systems to perform as intended. IPS's are useful to detect known attacks, but are inadequate to protect against new types of attack targeting the web applications and they can't check for traffic secured by SSL (Secure Sockets Layer), the security protocol on the Internet.

Internal security is based on secure coding standards [7] to ensure: 1) the continuing function of an application despite unexpected input or user actions, 2) the confidentiality and integrity of data, 3) provide access to the data when it is required (availability) and only to the right users. Practicing secure coding techniques like source code reviews, implementation of security policies, secure input-output handling, software testing, exception handling etc. helps in avoiding most of the software defects giving rise to vulnerabilities like buffer overflows, SQL injection, Cross-site Scripting etc. and improves the quality of the software. Internal security has been observed as the most flexible way of defending web applications. Different web applications have different security requirements. Checks that are efficient for one application may not be found useful for the other. Complete protection of web applications and web services thus equires a full understanding of the application structure and logic. By using secure coding approach application specific features can be added to cover a particular application. Moreover, unlike external security devices, with internal security approach inputs can be checked according to the context of an event. For example, in the Internalsecurity application that we developed contained a simple input validation code to check the roll number field and the name field. The code displays an error message whenever the input string that it is analyzing, contained malicious characters '<, >, =, ', + etc. Since it is possible to determine which of the two fields encountered malicious input, more appropriate error messages can be displayed which cannot be easily achieved using external security tools.

4. CONCLUSION

Based on the experiment and the facts discussed above it is concluded that internal security is a better way of defending web applications. Although external security products provide efficient protection against network layer attacks but they are unable toprotect efficiently at the application level.

ACKNOWLEDGMENT

The authors are thankful to University Grants Commission (UGC), and Ministry of Human Resource Development (MHRD), Government of India for providing financial assistance to carry out this research work. The authors are also thankful to Prof. Devanand, Head, Department of Computer Science and IT, University of Jammu, for his kind support.

REFERENCES

[1] P. Steve, "Anatomy of a web application: security considerations," Sanctum Inc. July,2001.

[2] A. Frederic, "Firewalls and internet security, the second hundred years," The Internet Protocol Journal, vol. 2, pp. 24-32, June 1999.

[3] K. Sailesh, D. Sarang, Y. Fang, C. Patrick, T. Jonathan, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," In the proceedings of the conference on applications, technologies, architectures, and protocols for computer communications, Pisa,Italy, 2006, pp. 339–350.

[4] A. Mansoor, M. Muthuprasanna, k. Vijay, "High speed pattern matching for network ids/ips," ICNP, 2006, pp.187-196.

[5] M. Abhishek, N. Walid, B. Laxmi, "Compiling PCRE to FPGA for accelerating SNORT," Proceedings of third ACM/IEEE symposium on architecture for networking and communication systems, Florida, USA, 2007, pp. 127-136.

[6] Z. Stefano, "Flaws and frauds in the evaluation of ids/ips technologies," In the proceedings of 19th annual FIRST conference on computer security incident handling,Sevilla, Spain,2007.

[7] C. Robert, "Secure coding standards," NIST, pp.14-16.

[8] John McHugh, Alan Christie, Julia Allen, "Defending Yourself: The Role of Intrusion Detection Systems," IEEE Software, vol. 17, no. 5, pp. 42-51, Sep./Oct. 2000, doi:10.1109/52.877859

[9] D.E. Denning, "An Intrusion Detection Model," IEEE Trans. Software Eng., Vol. SE-13, No. 2, Feb. 1987, pp. 222–232.

[10] K. Mookhey, B. Nilesh, "Detection of SQL Injection and Cross-site Scripting Attacks", Mar 2004. Retrieved from http://www. securityfocus. com/ infocus/1768 as accessed on 9-12-2009.

Authors Biography

Supriya Gupta is a research scholar at Department of Computer Science & IT, University of Jammu. Her research interests are in the field of Network applications' security.

Dr. Lalitsen Sharma obtained his PhD degree from Guru Nanak Dev University, Amritsar. He is working as an Associate Professor in the Department of Computer Science & IT, University of Jammu. His research interests are in the field of Network applications' security.