Network Intrusion Detection and Analyzing User-agent Field XSS Attack Log Files from Web Application

Aye Aye Thu
University of Computer Studies (Yangon)
Suchiq13@gmail.com,ayeayethu13@gmail.com

Abstract

Today web site hacks are on the rise and pose a greater threat than the broad-based network attacks as they threaten to steal critical customer, employee, and business partner information stored in applications and databases linked to the Web. Organizations collect vast amounts of data every day, including firewall logs, system logs, and intrusion detection alerts. Analyzing web traffic out of log files has advantages over analyzing traffic from the network. Web server log files contain only a fraction of the full HTTP request and response. A network Intrusion Detection System (NIDS) is placed in the network infrastructure where it can see the traffic to and from the web application. Cross-Site Scripting (XSS) attacks are a type of injection problem, in which malicious scripts are injected into the otherwise benign and trusted web sites. In this paper describes the detection of attacks on web application by analyzing user-agent field XSS log files from web servers (like Apache and IIS).

Keywords

Network Intrusion detection system, HTTP, Cross-Site Scripting, XSS log, Web Server

1. Introduction

Internet usage and online application are experiencing spectacular growth. This growth in popularity has not gone unnoticed by the criminal element and the simplicity of the HTTP protocol makes it easy to steal and spoof identity. Attacks on web application are on a constant change. Attackers are being finding flawed web applications using Google and other search tools. Attacks like XSS target the applications 'users, while all the other attacks target the web application itself. Standard web servers like Apache and IIS generate logging messages by default in the Common Log Format (CIF) specification. To detect attacks against web applications, the intrusion detection mechanism have to be application layer aware and see the relevant traffic.

Cross-site scripting (XSS) attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user [4]. Malicious user agents can also be responsible for denial of service and security bypass attacks [1].

The biggest benefit of log files is the relative simple availability and analysis of their content.

The rest of the paper is organized as follows: Section 2 presents Network Intrusion Detection System. Section 3 is the proposed framework and analyzing XSS attack log files in section 4. In section 5 presents the conclusion.

2. Network Intrusion Detection System (NIDS)

Network Intrusion detection systems (NIDS) are an essential component of defensive measures protecting computer systems and network against harm abuse [3]. It usually resides on its own machine and analyzes the web traffic without touching the firewalls and the application itself. Snort, the most powerful open source IDS, has over 800 rules for detecting malicious web traffic. This type of IDS captures network traffic packets such as TCP, UDP and IPX/SPX) and analyzes the content against a set of RULES or SIGNATURES to determine if a POSSIBLE event took place. False positives are common when an IDS system is not configured or "tuned" to the environment traffic it is trying to analyze [7]. Figure (1) shows the network based Intrusion Detection System architecture.

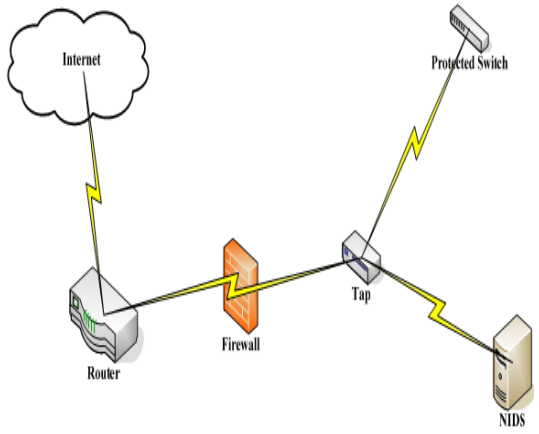

Figure 1.Network-Based IDS

3. Proposed Framework

Web applications are running on the OSI Layer 7-the application layer. Attacks can be detected at different zones and devices in the network infrastructure (see in Figure 2). Each place has a

different view of the traffic. This paper is now going to explore each of these places in the network.

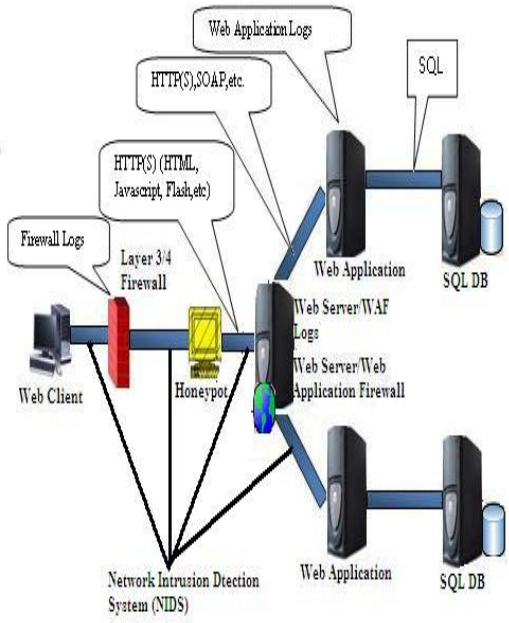

Figure 1. Network-Based IDS for Web application

3.1. Layer 3/4 Firewall

A traditional (stateful and non-stateful) firewall is working on OSI layers 3 (Network Layer) and 4 (Transport Layer). The firewall analyzes traffic based on the common protocols like TCP, UDP and ICMP and their corresponding ports or types/codes. Firewalls can detect anomalies in the protocols they are aware of like fragmented IP traffic, but they are generally not the best place to detect attacks on the application layer. Firewall log files usually do not contain application layer data like HTTP data, only layer 3 and 4 information, so they are not very helpful in detecting what is going on higher layers.

3.2. Web Application Honeypots

A web application honeypot (WAH) is a basic web server with an attack surface. This attack surface is the public HTML content which is indexed by search engines. It contains links to files with known vulnerabilities. The real vulnerability is not present but the web server advertises its existence and thereby attracts the adversaries. In order to handle an attack properly it is need to classify the request. This is almost identical to what the web application firewalls are trying to achieve except that they are prone to extensive false negative as they have to deal with classifying a lot of legitimate traffic. The honeypot, by contrast, should see little legitimate traffic, which dramatically simplifies this classification process [2]. On the web application Honeypot, request handlers are responsible for classifying and handling each incoming request.

3.3. Application layer firewall

Web application firewalls are designed to work on the OSI layer 7 (the application layer). They are fully aware of application layer protocols such as HTTP(S) and SOAP and can analyze those requests in great detail. Compared to a layer ¾ firewall, rules can be defined to allow/disallow certain HTTP requests like POST< PUSH, OPTIONS, etc., set limits in file transfer size or URL parameter argument length. WAF log files contain as much information as those from a web server plus the policy decisions of the filter rules. A WAF provides a wealth of information for filtering detection purposes and is thus a good place for the detection of attacks.

3.4. Web Server

The web server is the end device of an HTTP request. Standard web servers like Apache and IIS are logging by default in the common Log Format (CLF) specification. Web server logs do not contain any data sent in the HTTP header, like POST parameters. The HTTP header can contain valuable data, as most forms and their parameters are submitted by POST requests. This comes as a big deficiency for web server log files. A web server can also act as a web application firewall (WAF). WAF detects attacks by filtering all incoming HTTP and HTTPS traffic through configurable network and application layer controls. WAF's core security parameters are based on ModSecurity, an industry standard and trusted rule set that detects and prevent common exploitation techniques such as SQL Injection and Cross Site Scripting (XSS).

3.5. Web Application

A web application consists of a framework (PHP, ASP, J2EE, etc.) which implements the business logic. It is considered to be best practice to perform input/output validation in this tier. A strong input validation policy will detect malformed and malicious input and can log security related information to a log file. The application has access to the full user trail- each step a user takes (logging in, making a transfer, logging out, etc.). A comprehensive logging at the application tier enables the detection of misuse and fraud and allows a full reconstruction of a user's steps.

4. Cross-Site Scripting (XSS)

Cross-Site Scripting typically involves executing commands in a user's browser to display unintended content, or with the intent of stealing the user's login credentials or other personal information. This information can then be used by the attacker to access web sites and services for which the compromised credentials are valid (e.g., identity theft). In some cases, the attacker might be able to use this information to hijack or further compromise the user's HTTP sessions. Cross-site scripting **(**XSS**)** is a type of computer security

vulnerability typically found in web applications which allow code injection by malicious web users into the web pages viewed by other users. Examples of such code include HTML code and client-side scripts. There are Three Types of XSS.

(1) Persistent (Stored) XSS
Attack is stored on the website server.

(2) Non Persistent (reflect) XSS
User has to go through a special link to be exposed.

(3) DOM-based XSS
Problem exists within the client-side script.

In this paper, it is focus on the non persistent XSS attack.

4.1 User-agent field XSS

In this paper, aids intrusion analysts in understanding the user agent field and how it can be used to detect attack log files. Malicious attacks using the user agent field in HTTP request headers [1]. Modern examples of user agent are Mozilla Firefox, Internet Explorer and Safari. The user agent is defined by RFC2616.The user agent header field contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations.

User agents should include this field of requests. The field can contain multiple product tokens and comments identifying the agent and any sub products, which form s significant part of the user agent. By convention, the products tokens are listed in order of their significance for identifying the application.

User-Agent =" User-Agent" ":" 1*(product | comment)"

The above quotation from RFC 2616 shows that in 1991 the user agent field had three functions. Firstly it was to be used for statistical purposes. Websites can track what user agents are connecting to them and can use this information to help guide developers as to how to best display information to users. An example would be if developers may tailor the site better for iPhones.

The second function was for the tracing of protocols violations. This is an error control feature for user agents. The third function, to tailor responses based upon the user agent, is what the user agent is mainly used for today.

Mozilla/5.0 (iPad; U; CPU OS 3_2 like Mac OS X; en-us) AppleWebKit/ 531.21.10(KHTML, likeGecko)Version/4/0/4Mobile/7B334bSafari/531.21.102011-10-1620:23:50

User agent is Mozilla/5.0.Mozilla is common to nearly all modern browsers. The device is possibly an iPad 1 as its running an early 3_2 OS, but it is impossible to tell. The Apple developer site gives more information. (iPad; U; CPU OS 3_2 like Mac OS X; en-us) shows the platform string. In this case an iPad. AppleWebKit/531.21.10(KHTML, like Gecko) shows the Webkit engine build number. Version/4.0.4 shows the safari family version number.4.0.4.Mobile/7B334b shows the mobile version build number. Safari/531.21.102011-10-1620:23:50 shows the safari builder number.

To keep thinking like a hacker, this information is really useful especially if new vulnerabilities are released for particular product versions. The attacks would be adjusted based on the information found in the user agent field.

4.2 Analyzing of User Agent Field XSS Attack Log Files

In this paper, web application honeypots picked up some more XSS attack.

The highlighted data in the Apache access_log holds the User-Agent field token data from the request. In this case, the attacker has inserted some JavaScript code that would use the window. location function to cause the web browser to request the txt2pic.com website. After checking out that location the system finds the following:

$ **curl -D - http://txt2pic.com**
HTTP/1.1 302 Object moved
Server: Microsoft-IIS/5.0
Date: Fri, 30 Nov 2012 14:36:28 GMT
Fun: www.WHAK.com
Connection: close
Location: http://www.imagegenerator.org
Content-Length: 150
Content-Type: text/html
Set-Cookie:
ASPSESSIONIDCQSCSBBC=HCPFGNFAEIIHNDEPAEFEFFHL; path=/
Cache-control: private

Object moved
<h1>Object Moved</h1>This object may be found <a href="http://www.imagegenerator

This server responds with a 302 redirect and sends the user onto the www.imagenerator.org website. So, this attack scenario presumably is simply a method of SPAM linking to increase web traffic hits [5].

Another example is the log files from Scan 31 can be downloaded from the Honeynet Project website analyzing the apache access_log file with the above regular expressions yields interesting findings [6] . Here are two examples requests:

(1) 217.160.165.173 -- [12/Mar/2004 :22: 31 :12 – 0500] “GET /foo.jsp? <SCRIPT> foo </SCRIPT> .jsp HTTP/1.1” 200 578”_” “ Mozilla/4.75 [en] (X11, U; Nesus)”

(2) 217.160.165.173 --[12/Mar/2004 :22: 31: 12 - 0500] “GET /cgi-bin/cvslog.cgi? file= <SCRIPT>window.alert</SCRIPT>HTTP/1.1” 403 302”-“ Mozilla/4.75 [en] (X11, U; Nesus)”

There are two requests of a Nessus scan, trying to find scripts which are vulnerable to XSS. According to the HTTP status code, in the first request the web server responded with a 200 OK, which means that foo.jsp was there and served a paged, It is don’t know if this page is vulnerable, though. The system would have to try this request manually to find out. The second request (cvslog.cgi) was not successful, the server responded with a 403 Forbidden response, which means that the web server denied the access.

5. Conclusion

Cross Site Scripting attacks work by embedding script tags in URLs/HTTP requests and enticing unsuspecting users to click on them, ensuring that the malicious JavaScript gets executed on the victim’s machine. These attacks leverage the trust between the user and the server and the fact that there is no input/output validation on the server to reject JavaScript or other active code characters. The propose framework showed that can detect about 96% XSS attacks. In this case of user agents, hackers have not only found ways to avoid the system which search through user agent logs from NIDS. The system is looking for the smallest mistake or slip up from a hacker.

References

[1] Darren Manners “The user agent field: Analyzing and detecting the abnormal or malicious in your organization”

[2] Honeynet Project Web Application Honeypot “http://en.wikipedia.org/wiki/Cross_site_scripting”

[3] J. Mchugh, A. Christie, and J. Allen, “Defending Yourself: The Role of Intrusion Detection Systems”, IEEE Software, Volume 17, Issue 5, Sep.-Oct., pp. 42-51, 2000.

[4] OWASP (The open Web Applicaion Security project)”https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29”

[5] Ryan Barnett [Honeypot Alert] “User-Agent Field XSS Attacks”|06 December 2012 at 15:43 http://www.imperva.com/resources/adc/adc_advisories_response_secureworks.html).

[6] The Honeynet Project & Research Alliacnce (2007). Know your Enemy: Web Application Threats. http://www.honey.org/papers/webapp/

[7] W. T Work, “Intrusion Detection Systems (IDS)”, National Institute of Standers and Technology, 2003, available at: csrc.nist.gov/publications /nistpubs /800-94/SP800-94.pdf.