Anomaly Detection Using Negative Security Model in Web Application

Auxilia.M
Department of Information Technology
Christ College of Engineering & Technology
Puducherry, India
auxilia.michael@gmail.com

Tamilselvan.D
Department of Information Technology
Christ College of Engineering & Technology
Puducherry, India
tamil_man19@yahoo.co.in

***Abstract*—Today's combat zone for both ethical and unethical hackers is the web. Rapid growth of web sites and web applications gives way to deliver complex business applications through the web. As the web dependency increases, so do the web hacking activities. Web applications are normally written in scripting languages like JavaScript, PHP embedded in HTML allowing connectivity to the databases, retrieving data and putting them in the WWW site. A web application is vulnerable to many kinds of threats and attacks. In order to detect known attacks, some set of attack rules and detections are needed. In this paper, a negative security model based on misuse of web applications is used. This negative security model provides a Web Application Firewall(WAF) engine with a rule set, to ensure critical protection across every web architecture. WAFs are deployed to establish an increased external security layer to detect and/or prevent attacks before they reach web applications. This paper has been tested with apache web server's log file. We have tested successfully almost all the common attacks. This paper also allows for HTTP traffic monitoring and real-time analysis with little or no changes to existing infrastructure.**

Keywords-; Web intrusion, web application firewall, sql injection,cross-site scripting, attacker , rule set, logging.

I. INTRODUCTION

The Internet services and use of web application are increasing rapidly around the world. But unfortunately it has been found easy to disturb the functionality of the Internet by attacking its infrastructure taking advantage of Internet services, and protocols. Thus hackers feel that the website and web application, in general web as the warfare.

In order to hack a web application a web hacker needs the following (a) A web browser (b) An Internet connection (c) a port scanner, netcat, whisker(vulnerability checker) OpenSSL etc.. Web hacks falls under the following categories 1. URL interpretation attacks 2. Input validation attacks 3. SQL injection attacks 4. Impersonation attacks 5. Buffer overflow attacks. These attacks disallow regular services from being accessed by legitimate users. In today's world, web servers and web based applications are the popular corporate applications and become the targets of the attackers.

Anomaly or misuse detection of web attacks consists of rules and descriptions. These detections do not consider pre- - defined signatures. This type of attack detections became the challenging task for the researcher's.

Our proposed approach of anomaly based intrusion detection in web applications has the following features. 1. Full HTTP transaction logging is possible, allowing complete requests and responses to be logged 2. The HTTP traffic can be monitored in real time in order to detect attacks 3. Immediately prevents attacks from reaching your web applications 4. The WAF can either be deployed as a part of web server infrastructure or as a reverse proxy server in the network. The Core Rule set provides generic protection from unknown vulnerabilities often found in web applications, which are in most cases custom coded.

PAPER ROADMAP

Rest of the paper is organized as follows: In section 2, we have presented the related work and observations. In section 3, we have given the background of our proposed scheme. In section 4 we have given the proposed approach of attack detection. In section 5, we have given the implementation steps of our proposed approach. In Section 6, we have concluded with future work.

II. RELATED WORKS

The beginning of Intrusion Detection System is from the government and Ministry of Defence in 1980s [3]. The purpose of the system is to monitor network malicious behavior and was popular in the market in mid-1990s.

Once an attack is being detected, based on supervised clustering technique, the system administrator is informed and can take the corrective measure [8]. A clustering algorithm for intrusion detection based on the concept of fuzzy connectedness known as FCC algorithm is also proposed[7]

An anomaly-based system learning the profiles of the normal database access performed is also developed using a number of different models[6]. In [5], a detection system correlates the server-side programs referenced by client queries with the parameters contained in these queries. The system analyzes HTTP requests and builds data model based on the attribute length of requests, attribute character distribution, structural inference and attribute order.

Based on the survey done, following observations have been made:

* Limited work on anomaly based web intrusion detection;

* Detection of SQL injection attack and HTTP overflow attacks are challenging issue for researchers;

* Possibility of attack due to poor and careless web application coding by programmers.

To address this evolving area of research, we propose an anomaly based intrusion detection mechanism using WAF. Next, we provide the background of our work.

III. BACKGROUND

Web applications typically interact with a database to retrieve persistent data and then present the data to the user as dynamically generated HTML web pages. Accordingly, user inputs are treated as secluded lexical units which, if not properly disinfected, can cause the web application to generate unintended output. Some categories of web attacks are reported below.

A. *Sophisticated HTTP Attacks*

Sophisticated HTTP attacks are one the most popular hacking techniques. The attacks are usually performed using HTTP port 80 or other HTTP communications. Web server must control port 80 to transfer an HTTP request for a web page in order to operate the website. Since the requests are processed through HTTP, attackers manipulate or change these HTTP requests to gain entry into the web server. Teamed with the legitimate entry, they get the 'green signal' to bypass firewalls and several other security standards. Easy access to the web server assists them to pose any kind of attack. Defectively written applications are more susceptible to such attacks.

B. *SQL Injection Attacks*

SQL injection is a basically a trick to inject SQL command or query as a input mainly in the form of the POST or GET method in the web pages. Most of the websites takes parameter from the form and make SQL query to the database. With SQL injection attack, a intruder can send a crafted SQL query from the URL of the product detail page and that could possibly do lots of damage to the database. And even in worse scenario, it could even drop the database table as well.

Example of SQL Injection Attack in PHP:
Let's look at the usual query for user login in PHP,

```
$sql="SELECT * FROM tbl_user WHERE username= '".$_POST['username']."' AND password= '".$_POST['password']."'";
$result=mysql_query($sql);
```

Most of us think that only the valid user can log in inside the system but that's not true. But anybody can log in to that website with a simple trick.Let's suppose that a intruder injected x' OR 'x'='x in the username field and x' OR 'x'='x in the password field. Then the final query will become like this

SELECT * FROM tbl_user WHERE username='x' OR 'x'='x' AND password='x' OR 'x'='x'; but that query is always true and returns the row from the database. As the result, the malicious guy could log in to the system. Next, we present our approach which attempts to detect this attack using Web Application Firewall along with rule set.

C. *Cross-Site Scripting*

Cross-site scripting attacks occur when user input is not properly sanitized and ends up in pages sent back to users. This makes it possible for an attacker to include malicious scripts in a page by providing them as input to the page. The scripts will be no different than scripts included in pages by the website creators, and will thus have all the privileges of an ordinary script within the page—such as the ability to read cookie data and session IDs. If markup such as *<a href="...">* is allowed to be input by users in blog comments, forum posts, and similar places, then be aware that simply filtering out the *<script>* tag is not enough, as this simple example shows:

```
<a href="http://www.google.com"
onMouseOver="javascript:alert('XSS
Exploit!')">Innocent link</a>
```

This link will execute the JavaScript code contained within the *onMouseOver* attribute whenever the user hovers his mouse pointer over the link. Even if the web application replace *<script>*tags with their HTML-encoded version, an XSS exploit would still be possible by simply using *onMouseOver* or any of the other related events available, such as *onClick* or *onMouseDown*.

IV. PROPOSED APPROACH

The proposed approach is to use WAF which is deployed to establish an increased external security layer to detect and/ or prevent attacks before they reach web applications. It provides protection from a range of attacks against web applications and allows for HTTP traffic monitoring and real-time analysis with little or no changes to existing infrastructure.

The WAF can be either inserted as a part of web server infrastructure or it can be deployed as a reverse proxy server in network. Figure 1 depicts the deployment of the WAF

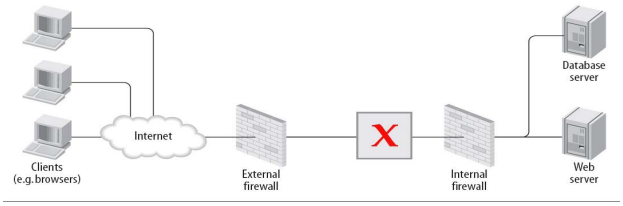

Figure 1. Deployment of the WAF in the web server infrastructure as well as external reverse proxy server

Web Application Firewall engine can provide very little protection on its own. In order to become useful, it must be configured with rules such that its advantages are fully exploited. The significance of using rule set is that it is not signature based and can provide generic protection from unknown vulnerabilities found often in a web application. To achieve this rule set use the following techniques:

• HTTP protection - detecting violations of the HTTP protocol and a locally defined usage policy.

• Common Web Attacks Protection - detecting common web application security attack.

• Automation detection - Detecting bots, crawlers, scanners and other surface malicious activity.

• Trojan Protection - Detecting access to Trojans horses.

• Error Hiding - Disguising error messages sent by the server.

The rule set can be used along with directives. The advantage of using the directives is that the rules can be evaluated for its correctness if specific "allow" rules are need to implement or to correct any false positives in the rules as they are applied to a site. The following are some of the rules to detect some of the common attacks.

A. Negative Security Model

Negative security model monitors requests for anomalies, unusual behaviour, and common web application attacks. It keeps anomaly scores for each request, IP addresses, application sessions, and user accounts. Requests with high anomaly scores are either logged or rejected altogether.

B. System Architecture

The proposed system architecture is given below in Figure 2

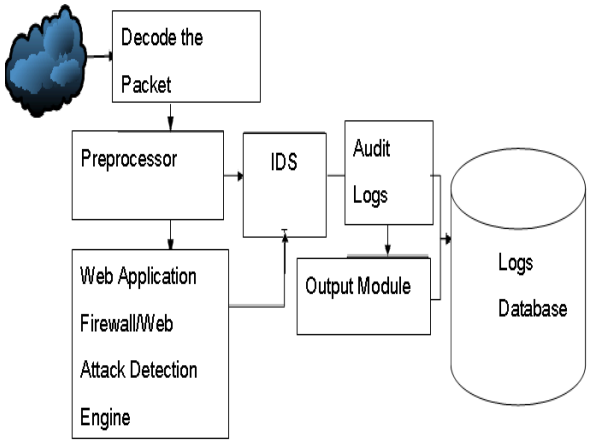

Figure 2. Proposed System Architecture

Decode the Packet

PCAP library of Snort is used to capture the packets transferred/received in the LAN which contains the captured time, packet length, and link type (for example: Ethernet, FDDI etc.). It also creates a pointer pointing each packer for efficient analysis. With the inline mode, it has additional function of firewall such as packet transfer, packet modification, rejecting specific packets or dropping them.

Preprocessor

After packets are captured, they are transferred to preprocessor for packet repacking and normalizing based on the format of each protocol. The preprocessor also analyses statistics of the network traffic and detects unregulated attacks such as deny of service and worms.

IDS

The IDS is the core of the proposed system. Users could download the signature database from the official website. With suitable settings detection of network attacks can be effective. If the packet captured matches any signature pattern, the system will alert an attack alarm to audit logs.

Audit Logs

When the system determines an attack, it will generate logs and make an alert containing related information of the attack for the administrator to remove the attack.

Output Module

Various output modules are supported such that users can choose suitable for different environments and objects.

- Default Logging
- SNMP traps
- XML Logging
- Syslog
- SMB Alerting
- PCAP logging
- SnortDb
- Unified Log

C. Rule Set to Detect Some Attacks

Some of the sample rule sets adopted are the following

1. To detect HTTP attacks the following rules are used

```
# Defeat HTTP fingerprinting
#
# Change server signature
SecServerSignature "Microsoft-IIS/6.0"
# Deny requests without a host header
SecRule &REQUEST_HEADERS:Host "@eq 0"
"phase:1,deny"
# Deny requests without an accept header
SecRule &REQUEST_HEADERS:Accept "@eq 0"
"phase:1,deny"
# Deny request that don't use GET, HEAD or POST
SecRule REQUEST_METHOD !^(get|head|post)$
"phase:1,t:lowerCase,deny"
# Only allow HTTP version 1.0 and 1.1
SecRule REQUEST_PROTOCOL !^http/1.(0|1)$
"phase:1,t:lowercase,deny"
# Add X-Powered-By header to mimic IIS
Header set X-Powered-By "ASP.NET 2.0"
# Remove the ETag header
#Header unset ETag
```

2. To prevent SQL injection in Cookies the following rule is used

```
#Prevent SQL injection in cookies
```

SecFilterSelective COOKIE_VALUES "((select|grant|delete|insert|drop|alter|replace|truncate|update|create|rename|describe)[[:space:]]+[A-Z|a-z|0-9|*|\,]+[[:space:]]+(from|into|table|database|index|view)[[:space:]]+[A-Z|a-z|0-9|*| |\,]|UNION SELECT.*\'.*\'.*,[0-9].*INTO.*FROM)" "id:300011,rev:1,severity:2,msg:'Generic SQL injection in cookie'"

Generic filter to prevent SQL injection attacks

Understand that all SQL filters are very limited and are very difficult

to prevent false postives and negatives.

Pplease report false positives/negatives to mike@gotroot.com

SecFilterSelective REQUEST_URI "!((/wp-admin/post|privmsg|/ticket/admin|/misc|tiki-editpage|/post|/horde3?/imp/compose|/posting)\.php|/modules\.php\?op=modload&name=(Downloads|Submit_News)|/admin\.php\?module=NS\-AddStory\&op=|/index\.php\?name=PNphpBB2&file=posting&mode=reply.*|/phpMyAdmin/|/PNphpBB2-posting\.html|/otrs/index\.pl|tiki-index\.php\?page=|/index\.php\?title=.*&action=edit|/_mmServerScripts/|/node/[0-9]+/edit|/_vti_bin/.*\.exe/)" "chain,id:300013,rev:1,severity:2,msg:'Generic SQL injection protection'"

SecFilter "((select|grant|delete|insert|drop|alter|replace|truncate|update|create|rename|describe)[[:space:]]+[A-Z|a-z|0-9|*|\,]+[[:space:]]+(from|into|table|database|index|vi

3. To detect cross-site scripting the following rules are used:

```
#
# Add HttpOnly flag to session cookies
#
SecRule RESPONSE_HEADERS:Set-Cookie
"!(?i:HttpOnly)"
"phase:3,chain,pass"
SecRule MATCHED_VAR "(?i:sess(ion)?id)"
"setenv:session_
cookie=%{MATCHED_VAR}"
Header set Set-Cookie "%{SESSION_COOKIE}e;
HttpOnly" env=session_cookie
```

D. Logging

Using the proposed approach, it is possible to log full HTTP transaction, allowing complete requests and responses to be logged. Here is some of the sample shots of how the request is logged

```
==========================================
Request: 192.168.0.2 - - [[18/May/2003:11:20:43 +0100]] "GET /cgi-bin/printenv?p1=666 \
HTTP/1.0" 406 822
Handler: cgi-script
----------------------------------------
GET /cgi-bin/printenv?p1=666 HTTP/1.0
Host: wkx.dyndns.org:8080
User-Agent: security regression test utility
Connection: Close
WAF-message: Access denied with code 406. Pattern match "666" at \
ARGS_SELECTIVE
action: 406

HTTP/1.0 406 Not Acceptable
==========================================
```

A typical audit log entry may look like this:

```
--67458b6b-A--
[15/Jul/2005:11:56:52 +0100] G3yTd38AAAEAAAM7BLwAAAAA \
192.168.2.11 4236 192.168.2.101 8080
--67458b6b-B--
POST /form.php HTTP/1.1
Host: 192.168.2.101:8080
User-Agent: Mozilla/5.0
Accept: */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://192.168.2.101:8080/form.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 5

--67458b6b-C--
f=111
--67458b6b-E--
403 (Response body)
--67458b6b-F--
HTTP/1.1 403 Forbidden
Last-Modified: Fri, 08 Jul 2005 14:25:30 GMT
ETag: "decb4-3-34b96a80"
Accept-Ranges: bytes
Content-Length: 19
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
--67458b6b-H--
ETag: "decb4-3-34b96a80"
Accept-Ranges: bytes
Content-Length: 19
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
--67458b6b-H--
Message: Pattern match "111" at POST_PAYLOAD \
[id "1"] [rev "2"] [msg "3"] [severity "4"]
Apache-Handler: application/x-httpd-php
Stopwatch: 1126536042708000 11024 (7276* 7375 9842) --67458b6b-Z—.
```

V. IMPLEMENTATION

The purpose of this paper is to modify the preprocessor in the system architecture [Refer Figure 2] to be able cooperate with the Rule Sets specified in the above section. The

following functions must be implemented to achieve this. The functions are illustrated in Figure 3.

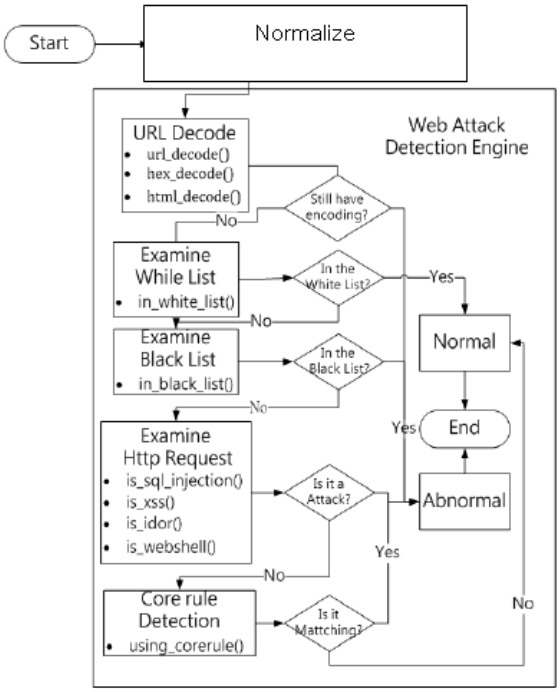

Figure 3. Implementation Steps

Normalization

Since the web server log is taken as input data, which Snort could not analyze directly this paper uses off-line mode to analyze collected web server logs. First, preprocess the web server based on Apache and IIS format to get full URLs, and then put them in the Web attack engine for analysis.

URL decode

Before the browser sends a HTTP Request,it would first encode some symbols and Chinesewords. Therefore, it should be decoded before the analysis, which would lower false alert rate of the system. At the meanwhile, hackers usually use multi-type encoding methods to encode the malicious codes evades Intrusion Detection Systems. This function would decode URLs and then determine those that still contain encoded URLs to be Web attacks.

Examining White List

Many kinds of web servers have special http request pattern and different search engines such as Google, Yahoo, Msn and Baidu spiders. They can be added into the white list in the first place to analyze them, as well as those that are similar to malicious attacks but normal Http request.

Examining Black List

The known malicious web sites are for black list [7,11]. They are usually presented as normal files likes as js, css, and other text files, or malicious binary files that contains shell codes such as gif, ppt and pdf etc. The malicious files could not be analyzed by simply examining the HTTP Requests. They must be examined by further analysis of their behaviors. If an http request contains web sites in the black lists, it will be determined a Web attack. The advantage is low false alert rate, but it could not detect an unknown malicious web site automatically.

Examining Http Request

This function mainly implements the ability to recognize the attacks toward the four major web application vulnerabilities, which include SQL Injection, Cross Site Scripting, Insecure Direct Object Reference and Cross Site Request Forgery.

Result Analysis is shown in Figure 4 and Figure 5

Date	Normal	SQL Injection	XSS	Insecure Direct Object Reference	Web Shell	Encoding	Total
Day 1	522,805	344	55	11,187	2	0	534,493
Day 2	137,803	448	95	4901	0	0	143,247
Day 3	357,922	281	102	58	35	0	358,108
Day 4	126,123	46	34	0	0	0	126,205
Day 5	199,069	98	71	0	3	0	199,241
Day 6	145,509	16	10	1	0	0	145,536
Total	1,489,231	1,223	367	16,659	40	0	1,507,530
Percentage	98.79%	0.18%	0.02%	1.11%	0.0%	0%	

Figure 4. Result Analysis of Unknown Attacks

	Normal	SQL Injectio	XSS	Insecure Direct Object Reference	Web Shell	En-coding	Total
Total	0	0	213	0	0	0	213
Percentage	0%	0%	85%	0%	0%	0%	85%

Figure 5. Result Analysis of Known Attacks

VI. CONCLUSION

In this paper, we have presented an offline web intrusion detection mechanism using negative security model. This paper detects, mitigates or blocks almost all the common

attacks using the web application firewall with Rule set. This paper creates awareness on creation of heightened levels of security. The observation is that it is wise to use formal software engineering methods for developing web applications. Secure coding practices must be used and thorough application testing is needed

Along with the current research projects, more insight into the vulnerabilities and motivation to find a complete solution is needed. The closing thought is that “There is no patch for carelessness”.

REFERENCES

[1] www.snort.org

[2] M. Almgren, H. Debar, M. Dacier, "A lightweight tool for detecting web server attacks", In Proceedings of the ISOC Symposium on Network and Distributed Systems Security, 2000 .sytms, America,Wiley

[3] Crothers,T., 2002, Implementing Intrusion Detection System, America,Willey.

[4] Battachariya, Debasish Das, Utpal Sharma, “A Web Based Intrusion Detection Mechanism Based on Feature Based Data Clustering”, International Advance Computing Conference,2009.

[5] Jung-Ying Lai, Jain-shing Wu, shih-Jen Chen, Chia-Huan Wu, Chung-Huang Yang, “Designing a Taxonomy of Web Attacks”, International Conference on Convergence and Hybrid Information Technology,2008.

[6] C. Kruegel, G. Vigna, "Anomaly Detection of Web-based Attacks",Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS '03),2003, pp. 251-261.

[7] Qiang and Vasileios Megalooikonomou, A Clustering Algorithm for Intrusion Detection. DEnLab, Temple University.

[8] Rebecca Bace, Peter Mell, NIST Special publication on Intrusion Detection Systesm, 16th August 2001.

[9] Sanghyun cho , Sungdeok cha, SAD, Web Session Anomaly detection based on Parameter Estimation, Computer Security,pp312-319,2004

[10] F. Valeur, D. Mutz, G. Vigna, "A Learning-Based Approach to the Detection of SQL Attacks", Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), Austria, 2005.

[11] IIS W3C Extended Log Format,http://www.loganalyzer.net/log-analyzer/w3c-extended.html