IMPLEMENT WEB ATTACK DETECTION ENGINE WITH SNORT BY USING MODSECURITY CORE RULES

Chung-Huang Yang and Chung-Hsiang Shen*

Graduate Institute of Information and Computer Education, National Kaohsiung Normal University
Kaohsiung, TAIWAN
E-mail: zongshen@mail2000.com.tw

Abstract. In the Web 2.0 generation, network system faced the racket - “Web attack”. Traditional network security devices like Firewall and Intrusion Detection System deal can hardly confront the threat of Web attacks since Hackers often use multi-level or multi-type encoding attack to evade Intrusion Detection Systems. The Intrusion Detection System usually uses the attack signature and Regular Expression to detect web attack, but with limited efficiency. The open source Web Application Firewall “ModSecurity” could use Core Rule to detect SQL Injection, Cross Site Scripting, Insecure Direct Object Reference and Cross Site Request Forgery attacks. The purpose of this paper is provide the ability of web attack detection for Snort by implementing the web attack detection engine using the Core Rule Sets of ModSecurity.

Keywords: Intrusion Detection System, Web attack, Web Application Firewall, Regular Expression.

1 INTRODUCTION

In this network generation, many kinds of resources and services are provided by the Internet is popular, such as e-shopping, business transactions and on-line game industries. However, the more convenient it is, the more danger it brings. For example, cyber-crimes occur frequently in on-line games, customer information of Citibank leaked out and cyber bank robbery happened to one of the domestic bank. Therefore, enterprises have deployed many kinds of network security facilities like Honeypot, Intrusion Detection System (IDS), and Firewalls.

When the network system suffers from attacking, it relies on some devices like Intrusion Detection System (IDS) to monitor the network behaviors and to analyze the attack methods. By the report from IDS, system administrators are able to fix the insufficiencies and to tighten the security of the system.

In the Web 2.0 generation, network companies have offered many web services like web page space, web album, blog, etc. Everyone can leave personal opinions to wherever one has browsed on the Internet, which brings a new type of attack, the Web attack.

According to Gartner’s report, network attacks using the weaknesses of Operation Systems, Applications or Protocols have turned to Web attacks, which are 75% of all attack activities.

However, the ability of Intrusion Detection System to detect various and flexible Web attack activities is limited. For example, the widely spread email forged to be from CNN in August 2008, contained the Phishing site, Flash malware and BOT NET. Tradition Intrusion Detection System using Regular Expressions to pattern matching attack signatures becomes inefficient and useless when it comes to those smart Web attacks.

The Open Source Web Application Firewall, ModSecurity, which uses rule based analysis engine, is more flexible than pattern matching signature. Recent Web attack methods usually use multi-level or multi-type encoding to avoid being detected. Therefore the IDS could not make correct analysis on HTTP Requests with bad intention. On the contrary, by using Core Rule of ModSecurity, one could effectively detect the SQL Injections, Cross Site Scripting, Insecure Direct Object References and Cross Site Request Forgery from the Web attacks.

The purpose of this paper is to use the Core Rule architecture of ModSecurity and to modify Snort preprocessor to load Core Rule to detect above malicious Web attack behaviors.
2 RELATED WORKS
2.1 The History of Intrusion Detection System
The beginning of Intrusion Detection System is from the government and Ministry of Defence in 1980s \cite{1}. The purpose of the system is to monitor network malicious behavior and was popular in the market in mid-1990s.

2.2 The philosophy of Intrusion Detection System designs
The Intrusion Detection Systems can be divide into 3 types according to it’s design \cite{2,3}. Network-based intrusion detection system
It uses sniffer mode to capture packets transferring in the internet. Then, put it into to the built in export system for pattern match. Therefore, it needs powerful computing capability, so most enterprise Intrusion Detection System sales in Hardware devices. The advantage is:
  - Deployment easily: Network Intrusion Detection System using passive mode to sniffer packets from the LAN and analysis them.
  - Lower cost: In large network environments just only deploy a few Sensors to monitor the area.
  - Large area: In sniffer mode it could monitor the large areas for abnormal activities, the area is bigger than Host Intrusion Detection System.
Host-based intrusion detection system
It is used to monitor critical Host systems by checking the users, system activities and attack behavior in the host. The Advance HIDS also provides Policy auditing, Access Control, Data Forensics ability. The advantage is: The more detail the logging data is, then more effect. Because it monitors the host system logs.
Network-node Intrusion Detection System
Also called as Distributed Intrusion Detection System; the processing method is the same with Network-based Intrusion Detection Systems. The difference is DIDS will forward logs to a back-end management analysis platform. It is more suitable with large network environments.

The Intrusion Detection Systems could divide to 3 types according to the detect mechanisms used \cite{4,5}.
Signature-Based Intrusion Detection System
The signature is to analyze previous attacked information by experts. It is based on pattern match with packet and signature. The advantage is: It could detect known attack, low false alarms, and more efficient and the disadvantage is it could not detect unknown attack, and needs to update the signature database frequently.
Anomaly-Based Intrusion Detection System
It uses the built in normal communication model to analyze, when against the model then justices to anomalous. The advantage is: It could detect unknown attack method, but the disadvantage is the false alert alarms higher and less effort.
Hybrid Intrusion Detection System
It combines Signature-based and Anomaly-based characters. It is the future trend.

2.3 Web Attack
According to Gartner’s research report, in more than 300 websites they have analyzed, 97% are with security weakness. And 75% attacks are on application level. In the OWASP 2007 report, there are 10 major security weaknesses in the Web attack methods, and here are those related to program coding \cite{6}:
Cross Site Scripting \cite{7}
By using unchecked input attributes, cause of hackers could embed executable malicious code. It may be embedded with Trojan or directed to a phishing site.
Injection Flaw
By the weakness of unchecked input attributes, hackers could modify unauthorized access command or database. On some special environment, then even may obtain system administrator privilege. It is often used Drive-by Download method.
Malicious File Execution
It is an imperfection from web system design and may cause remote malicious codes to be executed remote malicious code. It usually happens on PHP based Webpages.
Insecure Direct Object Reference
Because of the defect in the system function design of file reading, hackers could arbitrarily access files under any paths. This threat may cause the exposure of major system files. For example, the password file would lead to accounts and passwords hacked, and hackers would get the system administrator privilege.
Cross Site Request Forgery
It is based on the Cross Site Scripting extends attack method. Hackers inject the malicious code, causing unauthorized code to automatically get executed on an authorization user id.
Why is the detection of web attack so difficulty \cite{8}? When the weaknesses exist
in web applications, attacks from hackers are normal http requests, and they could penetrate firewalls and evade Intrusion Detection System without making alerts. Besides attack methods today usually use multi-level or multi-type encoding to evades Intrusion Detection Systems. As a result, there are certain related detection solutions heuristic \[15\], statistics \[3,8\], Data Mining \[9\], Attack Graph \[10\] etc.

3 SYSTEM DESIGN AND IMPLEMENTATION

3.1 Snort

Snort is a lightweight Intrusion Detection System developed by Marty Roesch in 1998 \[2,10\]. It is open source and has good processing effect, and is also the most popular Intrusion Detection System in Open platform. The most importance issue in the Signature-based Intrusion Detection System fields is the requirement of frequently-updated signature database. Snort has a customized rule set language. Users could construct the signature database they demand once familiar with the language. The system architecture is illustrated in Figure 1.

![Figure 1. System architecture](image)

Packet Decode

Snort uses PCAP library to capture the packets transferred/received in the LAN in which contain the captured time, packet length, and link type (for example: Ethernet, FDDI etc.). It also creates a pointer pointing each packet 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, Snort will transfer them 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.

Detection engine

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

Audit Logs

When the system determine attacked, it will generate logs and make an alert containing related information of the attack for the administrator to remove the attack. Snort has 2 mechanisms for alerts, Event Queue and Thresholds. When an attack happens and violates multiple rules, it would generate alerts in the order of the priorities defined for each rule in advance. Therefore the administrator gets to exclude less urgent items with Event Queue. The Threshold mechanism is to generate only one alert when a huge amount of the same attack behaviors happen in a short time. It is helpful when Deny of Service or the worm attack happens, which often causes lots of alerts, to decrease the same alerts and to simplify the complexity of tracking the source of attacks.

Output Module

Snort supports various output modules for users to choice from under different environments and objects.

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

3.2 ModSecurity

The open source Web Application Firewall “ModSecurity” is developed by Ivan Ristic \[1,13\]. It adopts rule-based detection engine which is more flexible than regular expression detection engine. Therefore ModSecurity is effective on detecting more kinds of Web attacks, such as SQL Injection, Cross Site Scripting, Insecure Direct Object Reference and Cross Site request Forgery.
3.3 Web attack detection engine implementation

The signature database of Snort is constructed by protocols and keywords. Then, it uses PCRE library for regular expression pattern matching which has better effect on specific signature attacks than on elastic Web attacks. Therefore, the purpose of this paper is to construct a Web attack detection engine by modifying the preprocessor of Snort “HTTP Inspect” to be able cooperate with the Core Sets of ModSecurity. The development environment is under Windows XP using GCC 3.4.5 (built-in MinGW 5.1) for implementation of the following functions. The functions are illustrated in Figure 2.

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 Chinese words. 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 speed up the analysis them, as well as those that are similar to malicious attacks but normal Http request.

**Examining Black List**

Collect known malicious web sites 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 need to be examined by further analysis of their behaviors. Therefore, this research refers to the black list released from web security forums. 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 site automatically.

**Examining Http Request**

This function mainly implements the ability to recognize the attacks toward the four major web application vulnerabilities from OWASP 2007, which include SQL Injection, Cross Site Scripting, Insecure Direct Object Reference and Cross Site Request Forgery. For example, the Web Shell detection function. When a user uploads data with web programs, such as ASP, PHP or JSP, etc., this can be determined a Web Shell attack. Table 1 shows the Attack keywords in the detect engine. Table 1 shows the Attack keyword in the detect engine.

<table>
<thead>
<tr>
<th>Function</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_sql_injection()</td>
<td>or , -- , -- , and , exec , select , insert , update , delete , drop , where , dbo , cast( , char( , union</td>
</tr>
<tr>
<td>is_xss()</td>
<td>javascript , &lt;script , /script&gt;, document.write, document.cookie , url(, eval(, expressionf, &lt;object, onload, onmouseover , onerror , windows.open, &lt;iframe, function, .location, (', ));</td>
</tr>
<tr>
<td>is_idor()</td>
<td>.ini , ./ , .\ , . , . , .swf? , boot.ini , etc/ , /passwd</td>
</tr>
</tbody>
</table>
| is_webshell() | <form, <%, <?, <php,
Table 2. Unknown attack analysis results

<table>
<thead>
<tr>
<th>Date</th>
<th>Normal</th>
<th>SQL Injection</th>
<th>XSS</th>
<th>Insecure Direct Object Reference</th>
<th>Web Shell</th>
<th>Encoding</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAY1</td>
<td>522,805</td>
<td>344</td>
<td>55</td>
<td>11,187</td>
<td>2</td>
<td>0</td>
<td>534,393</td>
</tr>
<tr>
<td>DAY2</td>
<td>137,803</td>
<td>448</td>
<td>95</td>
<td>4,901</td>
<td>0</td>
<td>0</td>
<td>143,247</td>
</tr>
<tr>
<td>DAY3</td>
<td>357,922</td>
<td>281</td>
<td>102</td>
<td>568</td>
<td>35</td>
<td>0</td>
<td>358,908</td>
</tr>
<tr>
<td>DAY4</td>
<td>126,123</td>
<td>46</td>
<td>34</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>126,205</td>
</tr>
<tr>
<td>DAY5</td>
<td>199,069</td>
<td>98</td>
<td>71</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>199,241</td>
</tr>
<tr>
<td>DAY6</td>
<td>145,509</td>
<td>16</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>145,536</td>
</tr>
<tr>
<td>Total</td>
<td>1,489,231</td>
<td>1,233</td>
<td>367</td>
<td>16,659</td>
<td>40</td>
<td>0</td>
<td>1,507,530</td>
</tr>
<tr>
<td>Present</td>
<td>98.79%</td>
<td>0.18%</td>
<td>0.02%</td>
<td>1.11%</td>
<td>0.0%</td>
<td>0%</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Known attack analysis results

<table>
<thead>
<tr>
<th>Normal</th>
<th>SQL Injection</th>
<th>XSS</th>
<th>Insecure Direct Object Reference</th>
<th>Web Shell</th>
<th>Encoding</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>0</td>
<td>0</td>
<td>312</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Present</td>
<td>0%</td>
<td>0%</td>
<td>99%</td>
<td>0.95%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

**Core Rule Detection**

This paper implements a detection engine to adopt the Core Rule Sets of ModSecurity to detect Web attacks. The advantage is: Snort could make use of another attack database to increase the detection rate.

4 EXPERIMENTAL RESULTS

Tradition Intrusion Detection Systems usually use the DARPA Dataset for intrusion detection testing, but it hasn’t been updated since 2000. Therefore, this paper uses the web server logs of more than 1.5 million records from a Non-Government Organization as the testing dataset. The testing adopts off-line mode to analyze logs within 6 days. The result shows in Table2.

The other testing dataset provided by XSS Attacks – Cross site scripting exploits and defense, contains 315 malicious web sites [6]. The analysis result shows in Table3.

5 CONCLUSION

This paper is to improve the inadequacy of Web attack detection for Snort by implementing a Web attack detection engine on the basis of the Http Inspect preprocessor. In addition to detect SQL Injection, Cross Site Scripting, Insecure Direct Object Reference and Cross Site Request Forgery, this detection engine also provides the ability to detect Web Shell, encoded attacks and malicious web sites on black lists. Besides, this paper implements the detection engine with the Core Rule Sets of ModSecurity to provide Snort with another attack database in order to detect various attack methods.

REFERENCES


