Http Rule Base Intrusion Detection and Prevention System

D.Sathya, S.Sangeetha

Abstract— The objective of HTTP Rule Base Intrusion Detection and Prevention System (IDPS) is to provide security for one of the application layer protocols namely HTTP (Hyper-Text Transfer Protocol). Such an HTTP based Intrusion Detection System (IDS) detects header attacks and attacks in payload (includes HTML and scripting). Misuse detection uses signature based approach where predefined patterns are defined. The input text or pattern is compared with the predefined signatures to detect malicious activity. Furthermore new types of attacks are continuously created. The new attacks created by attacker are also detected by these IDS, only if attacks are in the form of signatures. Signatures are defined either in a single-line or by complex script languages and are used in rule base to detect attacks. These signatures and rules have to be updated periodically as the attacks are continuously changing its nature of attacks.

Keywords — IDS, HTTP, Rule Base

I. INTRODUCTION

Application layer IDPS, blocks the HTTP Attacks that occur in application layer. The network layer intrusion detection system cannot block the application layer attacks. Firewalls in the network layer IDPS blocks the attacks entering through the unauthorized port [3]. Some complex threats can enter through authorized port (HTTP 80) and but goes undetected [1,2]. Those types of attacks can be detected by the application layer IDS. Misuse detection uses the signature based approach where attacks are identified by comparing with the predefined patterns. New type of attacks cannot be detected by the misuse based IDS. In the proposed system, the IDS are updated with the rules and attack patterns to detect the new types of attacks.

II. ARCHITECTURE OF HTTP RULE-BASED IDPS

The architecture of the HTTP Rule-Based Intrusion Detection and Prevention System is shown in Fig.1. The block diagram shows the overall architecture of HTTP Rule Based IDPs (RIDS) [4] and tells how various modules process the incoming data. Different modules involved in Intrusion Detection are Proxy Server, RIDS, Prevention [13, 15].

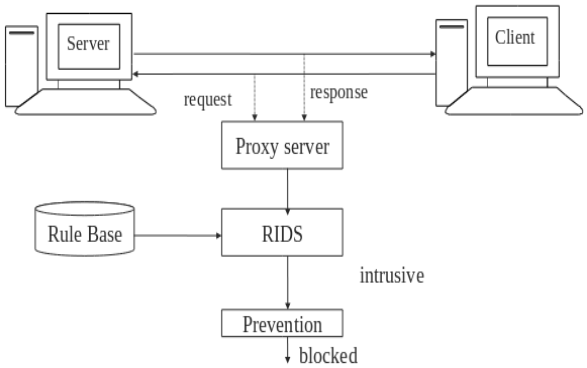

Fig.1 Architecture of HTTP Rule-Based IDPS

Various modules in this proposed system are Proxy Server, Rule-Based Intrusion Detection System and Prevention System [11, 12].

A. Proxy Server

An HTTP Proxy collects the HTTP traffic at application-layer from the network [10]. The proxy server gets the request from the browser and forwards the same to the service provider. The response obtained from the web server is also captured by the proxy before being forwarded to the browser.

B. Rule Based Intrusion Detection System

The data captured by the proxy server is separated into header and payload parts in RIDS. The Header Analyzer examines the header and compares it with the list of rules in the rule-base. Similarly the Payload Analyzer parses the HTML data and searches for misappropriate usage of tags and attributes and also checks for JavaScript based attacks injected in the HTTP by comparing with the rules in the rule base.

C. Prevention System

The intrusive patterns that are detected by the Rule-Base Intrusion Detection System are given as input to the prevention block. The requests / responses that are intrusive are blocked [5,7]. So the server / client doesn't know about that attack. The patterns that are blocked by the prevention block are stored in a database. These data can be used for analysis process.

The Network Intrusion Detection System detects the attacks which come through network layer protocol [8,9]. Some of the attacks which come through the application layer are not detected by the Network Intrusion Detection

Revised Manuscript Received on July 18, 2019.

D.Sathya, Assistant Professor II, Dept. of CSE, Kumaraguru College of Technology, Coimbatore. T.N, India.

S.Sangeetha, Assistant Professor, Dept. of CSE, SNS College of Technology, Coimbatore. T.N, India.

System. To detect application layer attacks this system is proposed. Some of the header and payload based attacks are as follows.

III. HEADER BASED ATTACKS

There are numerous HTTP Attacks that can bring a system to a compromised state. The following part will give an overview of the various HTTP attacks and the extent to which the intruders can compromise the systems or gain information about the system. As the HTTP is a stateless protocol any intrusion that is possible to occur can be caused by adding the command to the standard HTTP request. Some of the header based attacks are as follows.

*i) " * " Requests*

Rule Description

Format : *

Hexadecimal Equivalent : \%2A

Description: wild-card character attack.

Attack explanation

The attackers use an asterisk as an argument to a system command. Asterisk is a wild-card character which is normally used for representing zero or more characters. If it is used in a request then it may represent any possible string of text. So an intruder may use * to substitute for zero or more characters without explicitly giving a text.

Attack Detection

The attacker can create a HTTP request that contains '/*' in the URI which will match all possible combinations of characters that may come after the '/'. So if the objects representing '/' and '*' occur one after the other, then this attack is detected and is shown in Fig.2.

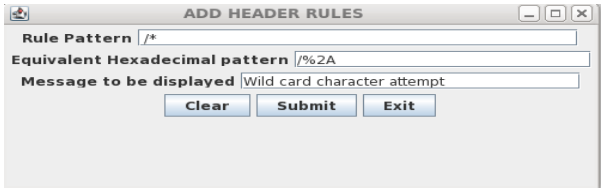

Fig.2 Wild character attempt

ii) "~" Requests

Rule Description

Format: \~

Hexadecimal Equivalent: \%7E

Description : attempt to find the valid user.

Attack explanation

http://host/~userids

The ~ character is used by attackers to determine who is a valid user. This request looks for a user named "userids" on the remote system. Often users will have web space and if the attacker manages to visit a web page, or get a 403 error (Denied error) then a user exists. Once an attacker has a valid user name, they may try guessing passwords, or brute forcing until they get a valid password

Attack Detection

The attacker can create a HTTP request that contains '/~' in the URI which will match the users with user name specified after the '~' character. So if the objects representing '/' and '~' occur one after the other, then this attack is detected.

iii) "chsh" command attempt

Rule Description

Format: /(chsh\%20)|

Hexadecimal Equivalent : (\%63\%68\%73\%68\%20)

Description: Attempt to change the user shell.

Attempt explanation

Using this command, an attacker may change the shell of a user to outfit their own needs. By changing the shell an attacker may further compromise a machine by specifying a shell that could contain a Trojan horse component or that could contain embedded commands specially crafted by an attacker.

Attempt Detection

The attacker can make a standard HTTP request that contains '/bin/chsh' in the URI which can then change the shell of a user present on the host. So preceding to this object, an object with '/bin' should be matched so that false alarms can be reduced.

iv) "gcc" command attempt

Rule Description

Format: /(gcc\%20)

Hexadecimal Equivalent: (\%67\%63\%63\%20)

Description : Attempt to compile C or C++ program.

Attempt explanation

This is an attempt to compile a C or C++ source on a host. The gcc command is the GNU project's C and C++ compiler used to compile C and C++ source files into executable binary files. The attacker could possibly compile a program needed for other attacks on the system or install a binary program.

v) "ps" command attempt

Rule Description

Format: /(ps\%20)

Hexadecimal Equivalent: (\%70\%73\%20)

Description : Attempt to gain intelligence of the processes run in the web server.

vi) "uname" command attempt

Rule Description

Format: /(uname\%20a)

Hexadecimal Equivalent: (\%75\%6e\%61\%6d\%65\%20\%2d\%61\%20)

Description : Attempt to gain intelligence about the operating system used.

vii) "chown" command attempt

Rule Description

Format: /(chown\%20

Hexadecimal Equivalent: (\%63\%68\%6f\%77\%6e\%20)

Description: Attempt to change the ownership permissions on a machine.

Attempt explanation

This is an attempt to change the file ownership permissions on a machine.

viii) "kill" command attempt

Rule Description

Format: /(kill\%20)

Hexadecimal Equivalent:(\%2b\%69\%6c\%6c\%20)

Description: Attempt to send a destructive signal to a specified process.

Attempt explanation

This is an attempt to send the specified signal to a specified process on a machine. Using this command, an attacker may send a destructive signal to a specified process running in the server which can make the system process to be critical.

ix) "chgrp" command attempt

Rule Description

Format: /(chgrp\%20)

Hexadecimal Equivalent: (\%63\%68\%67\%72\%70\%20)

Description: Attempt to change ownership permission of files.

Attempt explanation

This is an attempt to change the group of ownership of each given files to the named group on a machine.

Attempt detection

For the attacks from iv through ix, when the objects '/' , 'bin/' and the corresponding commands occur, rules written for these command attempts will detect and report to the administrator.

x) Directory Traversal Attack

Rule Description

Format: \.\.\/

Hexadecimal Equivalent: \%2e\%2e\%2f

Description: Attempt to traverse the directories.

Attempt explanation

This is an attempt to traverse through the directory levels and reach the root directory by issuing 'cd../' or by explicitly specifying '../' preceding a unix command.

Attempt detection

This attempt can be detected by matching the pattern '../' in the URI field of the HTTP request. Mostly, this attempt will be in its morphed form i.e. in ASCII form.

IV. PAYLOAD BASED ATTACKS

When a server is purposely overloaded with lots of requests from an intruder, it causes a denial of access to legitimate users. This attack can also be in the form of an infinite loop that gets executed in the client's browser. Examples

1. A JavaScript that loads cached images. Load this script on popular pages on compromised popular web servers.
2. Have the JavaScript loop a hundred times or so, each time requesting a random graphic or page name from the site targeted by the attack. If more number of users request the graphics at target site will deny the service.

The real source of attack is pretty much untraceable until you can track down at least one of the "users" taking part.

Code example

```
for(;;);
while(1);
do
{ -- -- }while(1); while(!0);
```

An infinite loop can either be a single loop with many iteration or multiple loops with less number of iterations to achieve the same purpose.

```
for (i=0;i<100000;i++);
for(i=0;i<100;i++) for(j=0;j<100;j++)
```

Regular expressions are therefore written that will look for traces of infinite loops. Following are examples of resource-consuming malicious scripts.

```
for (;;);
for (;;) document.write("foobar");
```

The above script is obviously an infinite loop. When a user is browsing a web page, if this script is made to execute, it causes the browser to become irresponsible.

Rule Formulated

The presence of an infinite loop needs to be identified. An infinite loop can be written in various ways. Either a for, a while or a do while can be used. for(;;); is obviously an infinite loop but it need not be the only way. Consider the following case:

```
for(i=0;i<500;i++)
{ for(j=0;j<500;j++)
{ for(k=0;k<500;k++){}
} }
```

The above loop will be executed 500*500*500 times. So, a limited range loop can be nested inside another loop of the same kind to act more or less an infinite loop. There is no need to give infinite for loop or while loop, the attacker can also try some other patterns to intrude the server. Though an exhaustive solution is not possible, a comprehensive solution was made by including the above scenarios is shown in Fig.3.

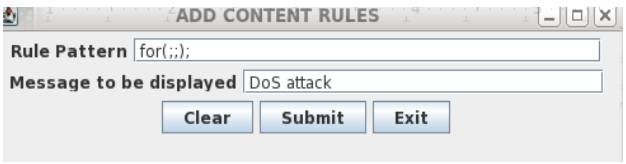

Fig.3 DoS Attack

V. EXPERIMENTS AND RESULTS

The Proxy server stores the header and payload information in separate files.

The output of the regular misuse detection module is non-intrusive for few attacks. In Denial of Service attack, the intruder makes the server busy and this denies access to the legitimate user. The intruder makes server busy by having infinite loops in the HTML payload. Sometimes the intruder executes the loop for larger number of times instead of executing infinite times. The attack which does not have a clear rule entry can also be detected. The detected attacks are stored in a file along with the date and time of detection as shown in the Fig.4.

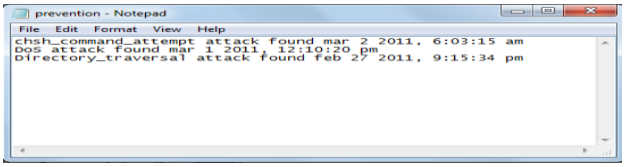

Fig.4 Detail of the attacks detected

The proposed system detects almost all possible attacks which are given in the rule base. The rule base is updated by the administrator whenever a new attack is encountered. This produces good results in detecting the malicious content. The table.1 shows the attack detection time of the proposed system over Header and Payload attacks.

Table 1 Time taken to detect the attacks by HTTP Rule-Based Intrusion Detection and Prevention System

Number of Rules	Time taken(ms)
10	0.2
20	0.4
50	1.0

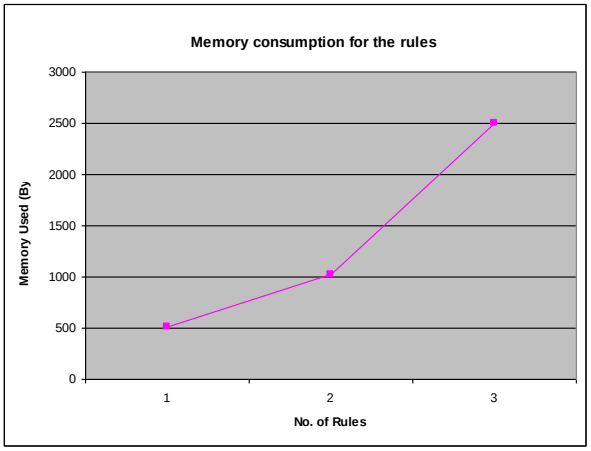

Fig. 4 Memory consumption for the rules

The proposed system uses less amount of memory, (i.e) memory consumption is based on the number of rules in the rule base and the content which are retrieved from the web browser are deleted after processing has been finished. The table 2 shows the amount of memory used for the number of rules.

VI. CONCLUSION

Thus, Intrusion Detection and Prevention System focuses on identifying possible malicious behavior that occur on application layer and block the detected maliciousness and finally it reports to the server. Rule-Based Misuse Detection follows a signature match approach which makes the system more efficient. Because of the continuously changing nature of attacks, the proposed system will update the signatures and rules. The system has been tested in a web environment and the results are presented. The results show the detection rate and time taken to detect an intrusion of the proposed system is better than regular instrusion detection existing techniques.

REFERENCES

1. T.Abbes,A. Bouhoula and M. Rusinowitch (2004), 'Protocol Analysis in Intrusion Detection Using Decision Tree', In the Proceedings of International Conference on Information Technology, Coding and Computing (ITCC' 04), IEEE.
2. E. Amoroso and R.Kwapniewski, (1998) ' A Selection Criteria for Intrusion Detection Systems', Proc. 14th Ann. Computer Security Applications Conf, IEEE Computer Soc. Press, Los Alamitos, Calif, pp. 280-288.
3. Andrew S. Tanenbaum, 'Computer Networks', 2nd Edition, Prentice Hall of India.
4. A.Anitha and V. Vaidehi 'Content based Application Level Intrusion Detection System'.
5. D.E. Denning, (1987) 'An Intrusion Detection Model', IEEE Trans. Software Eng., Vol. SE-13, No. 2, pp. 222-232.
6. R. Durst et al,(1999) 'Testing and Evaluating Computer Intrusion Detection Systems', Comm. ACM, Vol. 42, No.7, pp.53-61.
7. John McHugh, Alan Christie and Julia Allen (2000), 'Defending Yourself: The Role Intrusion Detection Systems', IEEE SOFTWARE pp. 42-51.
8. Karen Scarfone and Peter Mell, 'Guide to Intrusion Detection and Prevention Systems(IDPS)'.
9. C.Krugel and T.Toth (2003), 'Using decision trees to improve signature-based Intrusion Detection' in the proceedings of the 6th International Workshop on the recent advanced in Intrusion Detection(RAID'2003), LNCS v.2820, pages 173-191.
10. J. Mogul, R. Fielding, J. Gettys, H. Frystyk, L. Masinter, P. Leach and T. Bemers-Lee June 1999.RFC2616: Hypertext Transfer Protocol - HTTP/1.1.
11. Nick Ierace, Cesar Urrutia and Richard Bassett 'Intrusion Prevention Systems'.
12. S. Northcutt,(1999) 'Network Intrusion Detection', New Riders, Indianapolis.
13. V. Paxson, (1998) 'Bro: A System for Detecting Network Intruders in Real-Time', Computer Networks (Amsterdam, Netherlands: 1999), vol. 31, no. 23-24, pp. 2435– 2463.
14. M.V.Ramana Murthy, P.Ram Kumar, E.Devender Rao, A C Sharma, S.Rajender and S.Rambabu, ' Performance of the Network Intrusion Detection Systems', IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.10.
15. M. Roesch, (2003) 'Snort: the Open Source Network Intrusion Detection System', Development Paper at www.snort.org.
16. Duraisamy Sathya, Pugalendhi Ganesh Kumar, "Secured remote health monitoring system", Healthcare Technology Letters, pp. 1–5.
17. Sathya.D, Krishneswari.K, "Cross Layer Intrusion Detection System for Wireless Sensor Networks",Journal of Scientific and Industrial Research, Vol.75, pp.213-220, 2016.