Payload Recognition and Detection of Cross Site Scripting Attack

M. Ridwan Zalbina
Computer Engineering Dept., Faculty of Comp. Science
Sriwijaya University, Inderalaya, Indonesia
deris@unsri.ac.id

Tri Wanda Septian
Computer Engineering Dept., Faculty of Comp. Science
Sriwijaya University, Inderalaya, Indonesia
wanda@ilkom.unsri.ac.id

Deris Stiawan
Computer Engineering Dept., Faculty of Comp. Science,
Sriwijaya University, Inderalaya, Indonesia
zalbinaridwan@gmail.com

Moh. Yazid Idris
Faculty of Computing,
Universiti Teknologi Malaysia, Johor Bharu, Malaysia
yazid@utm.my

Ahmad Heryanto
Computer Engineering Dept., Faculty of Comp. Science
Sriwijaya University, Inderalaya, Indonesia
hery@ilkom.unsri.ac.id

Rahmat Budiarto
College of Computer Science & IT
Albaha University, Albaha, Saudi Arabia
rahmat@bu.edu.sa

Abstract— **Web Application becomes the leading solution for the utilization of systems that need access globally, distributed, cost-effective, as well as the diversity of the content that can run on this technology. At the same time web application security have always been a major issue that must be considered due to the fact that 60% of Internet attacks targeting web application platform. One of the biggest impacts on this technology is Cross Site Scripting (XSS) attack, the most frequently occurred and are always in the TOP 10 list of Open Web Application Security Project (OWASP). Vulnerabilities in this attack occur in the absence of checking, testing, and the attention about secure coding practices. There are several alternatives to prevent the attacks that associated with this threat. Network Intrusion Detection System can be used as one solution to prevent the influence of XSS Attack. This paper investigates the XSS attack recognition and detection using regular expression pattern matching and a preprocessing method. Experiments are conducted on a testbed with the aim to reveal the behaviour of the attack.**

Keywords—component; Cross Site Scripting (XSS), NIDS, web application security, data payload, regular expression

I. INTRODUCTION

Web-based service is a modular software that is published and located in every corner and can be accessed worldwide. Due the characteristics of distributed and openness (open nature), make web application technology consequently more sensitive to security [1][2][3]• . Cross Site Scripting (XSS) is generally defined as HTML injection, according to World Hacking Incident Databased (WHID), in 2011 approximately 12.58% of attacks in the world due to the XSS. XSS is one type of major attacks on the web application, based on the client and server side attacks that exploit the user's browser to execute malicious content on a website. Impacts of this attack include: the attacker can manipulate user sessions, cookies stealing, phishing, page redirecting and etc. Every system that vulnerable of this type can be fooled by this attack technique [4]• .

This paper utilizes Snort to capture packets data from the testbed network, furthermore regular expression technique is used to investigate the behaviour of the XSS attack. create these components, incorporating the applicable criteria that follow.

II. RELATED WORKS

Web Penetration Testing is performed at application level via HTTP or HTTPS traffic. There are several types of web application attack like Injection attack (SQL and Command Injection), Cross Site Scripting, Cross Site Request Forgery, Insecure Direct Object Reference and other types of attacks. Open Web Application Security Project (OWASP) is an official online community that specifically addresses web application security attacks and defenses.

In 2013, OWASP classified types of web application attack called OWASP TOP 10. The XSS is one of the top 3, this is due to the fact that attacks often occurred globally. It is also represented by the data obtained in the previous year in 2011, published by World Hacking Incident Database, which publishes the percentage of worldwide attacks that occurred in the Internet, again XSS became one of the types of attacks that is in the top of the list. Furthermore, according to standard of Mitre CWE – 79 (Common Weakness Enumeration) [5] XSS is explained as "improper neutralization of input during web page generation". The impact is very high to exploit. Vulnerability is also largely derived from the plug-ins such as java, flash (ActiveX), and browser extensions [6].

A. Network Intrusion Detection System (NIDS)

As the network level security, NIDS can be used as an all-in-one package for integrated enforcement of various types of attacks either on the level of non-web application or web application services. NIDS on most system work on the 3rd or 4th layer of OSI reference, with the addition of plugins and modularity of the system, it can be used to detect a service that works at the Application Layer. Snort itself, utilizes HTTP Inspect preprocessor as input plugins on duty to handle all incoming HTTP requests so that malicious content that traffic on the network can be monitored. It is also coupled with the ability of the system to speed up the matching of data packets through the regular expression and hexadecimal encoding that make detection of web application attack can be realized, yet again with architectural support to the detection engine which allows the use of rule options that provide flexibility, so the various forms of attack vector can be mitigated [7].

A study in [8] discussed comprehensively Snort rules and how those rules can improve security on a web application with several types of attacks such as XSS, SQL and Command Injection. The study also used Damn Vulnerable Web Application (DVWA) framework as a target system, which is used for the sake of testing and evaluating the rules. Furthermore, authors in [9] discussed the SQL Injection Attack that also utilized Snort NIDS as a detection system and compare another application layer software for detecting SQL Injection Attack.

B. Web Application Firewall (WAF)

As an active system which also performs prevention, WAF is a solution that can be an option, for web application attack prevention [10]. With a wide variety of platforms are available either open source or commercial such as, Modsecurity (Trustwave SpiderLab), ESAPI(OWASP), OpenWAF, Ironbee, Aqtronix, Webcastellum, Profense, and other platforms. Some systems use a rule-set in detecting patterns of web application attack.

III. Methodology

This paper mainly utilizes Snort for investigating the behaviour of the XSS. Snort itself can be used in several modes, namely Packet Sniffer, Packet Logger, Network Intrusion Detection System and Prevention System. For the purpose of this experiment in this paper, Snort is activated for all components for packet decoder, pre-processors, detection engine, and output plug-ins.

A. Rules Structure

The study emphasizes the use of rules to detect patterns contained in the request payload for non-persistent and persistent XSS Attack. The attack patterns are studied in advance in order to obtain a rule that can later on be applied. Rules are made based on the classification of the following: General, Payload, Non-payload, and Post-Detection. From each classification, rules are made to be able to detect series of attacks that will be tested on the system. In Fig. 1, the basic format of the rules contained in the detection engine is exhibited. Fig. 2 shows an example of XSS rules.

```
Rule action> | <Protocol> | <Source IP> | <Source Port> ->
<Destination IP> | <Destination Port> | <Action>
```

Fig. 1. Format Rules

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"[HTTP ATTACK] Terdeteksi XSS img tag PCRE";
flow:to_server; pcre:"/((\%3C)|<)((\%69)|i|(\%49))
((\%6D)|m|(\%4D))((\%67)|g|(\%47))[^\n]+((\%3E)|>)/I";
classtype:web-application-attack; sid:2000013;)
```

Fig. 2. Format Rules

B. Proposed Rules

Rules must have an identity in order to avoid overwriting. Snort itself has classified the boundaries of the rules that would be referenced and set, then SID number that is monitored if there is a renewal. In the experiment, the use of SID interval starting from 2000001 through 2000019. Each SID number is in the form of normal (web application activity) in priority-2 namely, GET and POST request, while for the form of malicious request (web application attack) in priority-2 are Non-Persistent and Persistent XSS Attack. The proposed SID rules are illustrated in Fig. 3.

IV. Experiment

In this experiment two types of XSS Attacks namely Non-Persistent and Persistent attacks will be detected. The data payload will also being involved and analyzed. The expereiments took place in Laboratory of COMNETS (Communication, Computer Network and Information Security) Faculty of Computer Science, University of Sriwijaya, Indralaya, Indonesia.

A. Procedure

The testbed uses multiple devices such as PCs, servers, and network peripherals such as switches and routers. Each PCs has the function as NIDS, Packet Analyzer, Web Server and Malicious User. Fig. 4 shows the topology of the testbed for the experiments. Table 1 lists the tools and their specifications, and network set up and information are shown in Table 2.

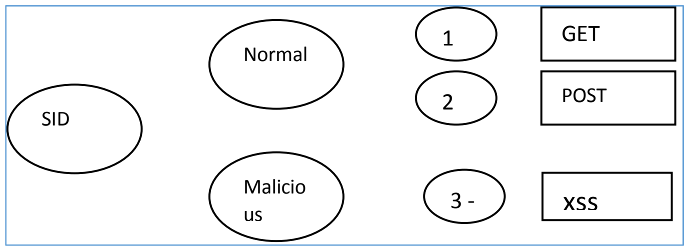


Fig. 3. Proposed SID Rules

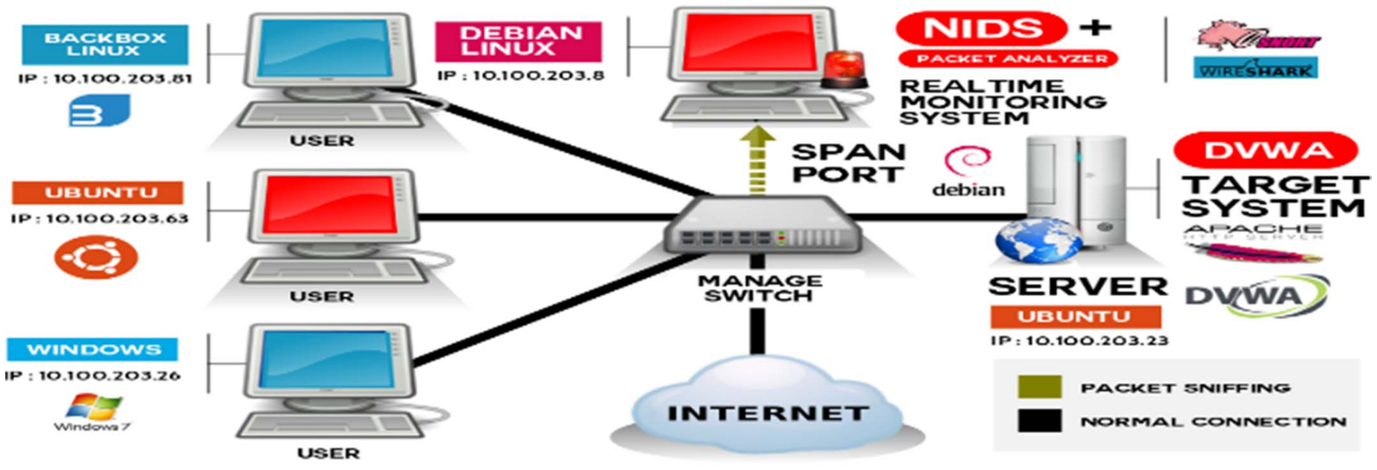


Fig. 4. Testbed topology for the experiment

B. *Attack Scenario*

The steps and scenarios to perform each attack in the experiments are as follows:

1. XSS Non-Persistent test-bed.
   a) Malicious users with IP 10.100.203.70, 10.100.203.74, 10.100.203.102 are trying to access target website (DVWA) that has been hosted in the server.
   b) Malicious users who use Backbox linux, Ubuntu and Windows 7 do the non-persistent XSS attack on the target.
   c) Malicious users insert malicious string, script withmultiple attack vectors to test whether the system is vulnerable to such attacks.
   d) The results of the malicious script in input field will be reflected in he URI with "GET" request method.
   e) If the system is vulnerable, then the response will be executed.
2. XSS Persistent test-bed
   a) Malicious users have found that by testing the non-persistent XSS system is exetermly vulnerable to attack.
   b) Malicious users insert malicious string, or script with different attack vector, and then take advantage of the mistakes made during the process of data entry or during the process of clicking occurs
   c) The results of the test on persistent attack are not reflected to the URI, but the process is directly stored in database with the "POST" as request method.

TABLE I. HARDWARE SPECIFICATION

Tools/ OS	Specification	Notes
Backbox Linux 4.4	Intel *Core* i3 2.4 GHz, 2GB RAM, Gigabit Ethernet *Interface*	1 unit
Ubuntu 14.04 LTS	Intel *Core* i5, 2.4 GHz, 4GB RAM, Gigabit Ethernet *Interface*	1 unit
Ubuntu 14.04 LTS	Intel *Core* i3, 2.1 GHz, 2GB RAM, Gigabit Ethernet *Interface*	1 unit
Debian *Server* 7.0	Intel *Core* i5, 2.7 GHz, 4GB RAM, Gigabit Ethernet *Interface*	1 unit
Windows 7	Intel *Core* i3 1.8 GHz, 2GB RAM, Gigabit Ethernet *Interface*	1 unit
Manage Switch	TP-Link *Smart Switch* TL-SG108E 8 *port* Gigabit Ethernet *Interface*	1 unit

TABLE II. NETWORK INFORMATION AND SETTINGS

Interface	Host	IP	Nmask
Port 1	WAN *Port*	10.100.203.0	/24
Port 2	*Malicious User*	10.100.203.70	/24
Port 3	*Server*	10.100.203.17	/24
Port 4	NIDS & *Packet Analyzer*	10.100.203.7	/24
Port 5	*Malicious User*	10.100,203.74	/24
Port 6	*Malicious User*	10.100.203.102	/24

V. Results and Analysis

Data were collected simultaneously at the time of the testing, therefore the data obtained at the same time as well. Data is dumped into several standard formats such as syslog, unified logging, full, fast, console alert and comma separated value (csv). Fig. 5 shows examples of the allerts detected by the regular expression during the experiments. Fig. 6 illustrates the correlation of the data during the attack, telling the pattern of relation among the data from sourced from the attacker to the target machine. From Table 3, for Non-persistent XSS can be seen that there is a normal request such as GET and POST are classified as a web application activity, then the web application attack there are some keywords for a variety of attack vectors such as SID:2000004 has the highest number of each IP address. Then image in keyword "img PCRE 2", next the image html tag in the form of at least PCRE detected [11].

```
[**] [1:2000001:0] Terdeteksi GET Request [**]
[Classification: Access to a Potentially
Vulnerable Web
Application] [Priority: 2]04/25-19:24:32.296541
10.100.203.70:47386 -> 10.100.203.17:80
TCP TTL:64 TOS:0x0 ID:19848 IpLen:20 DgmLen:770
DF
***AP*** Seq: 0x12555B47 Ack: 0x9163B5A6 Win:
0xE5 TcpLen:32
TCP Options (3) => NOP NOP TS: 7608001 3141454

[**] [1:2000002:0] Terdeteksi POST Request [**]
[Classification: Access to a Potentially
Vulnerable Web
Application] [Priority: 2] 04/25-22:38:34.208416
10.100.203.102:50222 -> 10.100.203.17:80
TCP TTL:128 TOS:0x0 ID:21612 IpLen:20 DgmLen:747
DF
***AP*** Seq: 0x4EEFF3B1 Ack: 0x82476B9B Win:
0x100 TcpLen: 20

[**] [1:2000004:0] Terdeteksi XSS script [**]
[Classification: Web Application Attack]
[Priority: 1]
04/25-22:02:29.252934 10.100.203.70:47558 ->
10.100.203.17:80
TCP TTL:64 TOS:0x0 ID:56164 IpLen:20 DgmLen:668
DF
***AP*** Seq: 0x10D2E6C9 Ack: 0xB083809D Win:
0xE5 TcpLen: 32
TCP Options (3) => NOP NOP TS: 9977259 5510697

[**] [1:2000018:0] Terdeteksi XSS embed keyword
[**]
[Classification: Web Application Attack]
[Priority: 1]
04/25-21:44:10.159419 10.100.203.74:50067 ->
10.100.203.17:80
TCP TTL:64 TOS:0x0 ID:23514 IpLen:20 DgmLen:917
DF
***AP*** Seq: 0x443C7B4C Ack: 0x39EBDC73 Win:
```

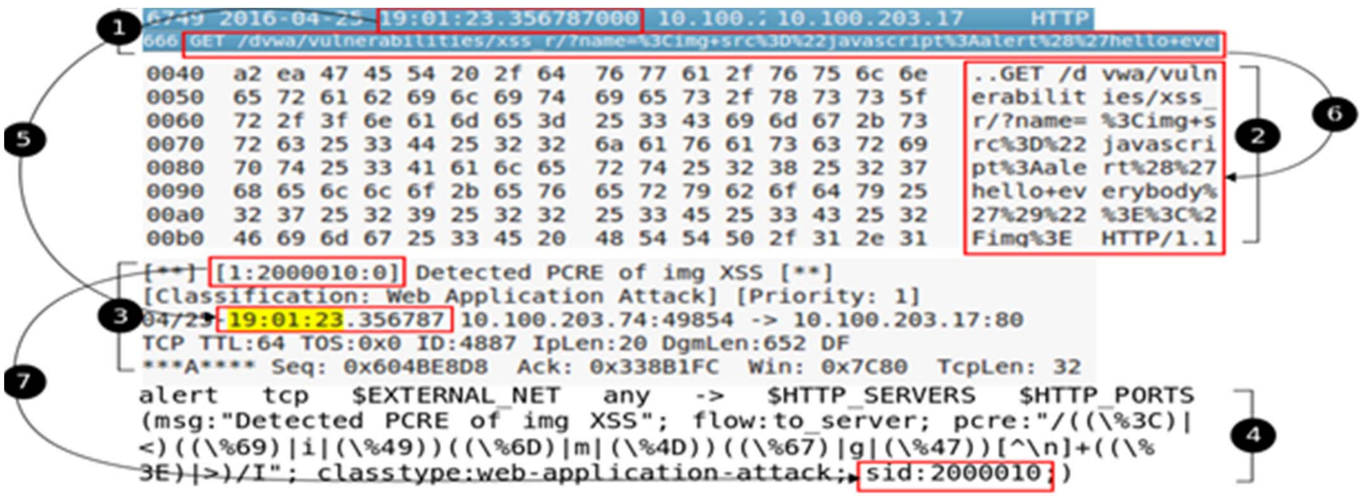
Fig. 5. Detected Alert

TABLE III. Alerts Based On SID Number Reflected XSS (Non-Persistent)

No.	Type of Request/Attack	SID	Priority	IP Address (10.100.203.*)		
				70	74	102
1.	GET Request	2000001	2	25	52	50
2	POST Request	2000002	2	0	2	1
3	XSS script 2	2000004	1	11	9	9
4	XSS iframe	2000005	1	0	2	2
5	XSS image	2000006	1	2	3	4
6	XSS script 3	2000008	1	2	2	1
7	XSS img PCRE	2000010	1	1	2	1
8	XSS html tag 1	2000011	1	2	1	3
9	XSS img PCRE 2	2000013	1	1	0	1
10	XSS html tag 4	2000017	1	2	1	1
11	XSS embed keyword	2000018	1	0	1	2

TABLE IV. Alerts Based On SID Number Stored XSS (Persistent)

No.	Type of Request/Attack	SID	Priority	IP Address (10.100.203.*)		
				70	74	102
1	GET Request	2000001	2	9	32	18
2	POST Rquest	2000002	2	21	20	22
3	XSS script 2	2000004	1	7	2	4
4	XSS iframe	2000005	1	3	2	5
5	XSS image	2000006	1	3	2	3
6	XSS script 3	2000008	1	2	2	0
7	XSS html tag 1	2000011	1	1	2	2
8	XSS html tag 2	2000012	1	1	1	1
9	XSS action script	2000015	1	1	2	2
10	XSS xml	2000016	1	0	1	0
11	XSS html tag 4	2000017	1	1	2	2
12	XSS embed keyword	2000018	1	2	2	2
13	XSS html tag 5	2000019	1	1	2	1



```
6749 2016-04-25 19:01:23.356787000  10.100.: 10.100.203.17      HTTP
666 GET /dvwa/vulnerabilities/xss_r/?name=%3Cimg+src%3D%22javascript%3Aalert%28%27hello+eve
0040  a2 ea 47 45 54 20 2f 64  76 77 61 2f 76 75 6c 6e   ..GET /d vwa/vuln
0050  65 72 61 62 69 6c 69 74  69 65 73 2f 78 73 73 5f   erabilit ies/xss_
0060  72 2f 3f 6e 61 6d 65 3d  25 33 43 69 6d 67 2b 73   r/?name= %3Cimg+s
0070  72 63 25 33 44 25 32 32  6a 61 76 61 73 63 72 69   rc%3D%22 javascri
0080  70 74 25 33 41 61 6c 65  72 74 25 32 38 25 32 37   pt%3Aale rt%28%27
0090  68 65 6c 6c 6f 2b 65 76  65 72 79 62 6f 64 79 25   hello+ev erybody%
00a0  32 37 25 32 39 25 32 32  25 33 45 25 33 43 25 32   27%29%22 %3E%3C%2
00b0  46 69 6d 67 25 33 45 20  48 54 54 50 2f 31 2e 31   Fimg%3E  HTTP/1.1
[**] [1:2000010:0] Detected PCRE of img XSS [**]
[Classification: Web Application Attack] [Priority: 1]
04/25-19:01:23.356787 10.100.203.74:49854 -> 10.100.203.17:80
TCP TTL:64 TOS:0x0 ID:4887 IpLen:20 DgmLen:652 DF
***A**** Seq: 0x604BE8D8  Ack: 0x338B1FC  Win: 0x7C80  TcpLen: 32
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"Detected PCRE of img XSS"; flow:to_server; pcre:"/((\%3C)|
<)((\%69)|i|(\%49))((\%6D)|m|(\%4D))((\%67)|g|(\%47))[^\n]+((\%
3E)|>)/I"; classtype:web-application-attack; sid:2000010;)
```

Fig. 6. Data Correlation

TABLE V. TOTAL UNWANTED ALERT

No.	Keyword	Priority	Total
1.	MISC UPnP malformed advertisement	3	3866
2.	BAD-TRAFFIC*	3	112
3.	ICMP*	3	310
4.	SCAN UPnP	3	4874
5.	DNS SPOOF	2	24
6.	NETBIOS SMB IPC unicode share access	2	4
7.	COMMUNITY WEB-MISC Proxy Server Access	2	14

The experiments was performed as for each IP address was executed for 20 attack vector, with nearly 85% of the system successfully detected with a maximum of three attack vector fail to be detected for each test, as shown in Table 4. Table 5 shows unwanted alerts, produced during the experiments.

VI. CONCLUSION AND FUTURE WORK

Snort NIDS is a security system that effective enough to detect and recognize the payload of XSS Attack. With the modularity of the architecture it is not possible in the future, almost any attack vector will be covered or detected. Although the implementation of the system does not perform the prevention or deterrence, however, for integrated security without the application of new technologies it is sufficient as a first defense barrier. This work has shown the way to investigate pattern or behaviour of the XSS attacks.

We are aware there are many deficiencies that can be considered as a foundation for further research, as examples include: the false positive that arise, the hardware resource, and also how to prevent the network level by involving technologies as an approach.

REFERENCES

[1] I. Hydara , A.B. Md. Sultan , H. Zulzalil , N. Admodisastro, "Current state of research on cross-site scripting (XSS) – A systematic literature review," *Information and Software Technology, Elesevier*, vol. 58, February 2015, pp. 170–186

[2] M.I.P. Salas and E. Martins, "Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services and WS-security," *Electron. Notes Theor. Comput. Sci. Elsevier*, vol. 302, pp. 133–154, 2014.

[3] W. Kim, O.-R. Jeong, C. Kim, and J. So, "The dark side of the Internet: Attacks, costs and responses," *Information Systems,* vol. 36, pp. 675-705, 2011.

[4] S. Gupta and B. B. Gupta, "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art," *Int. J. Syst. Assur. Eng. Manag. Springer*, 2015..

[5] T. Scholte, D. Balzarotti, and E. Kirda, "Have things changed now? An empirical study on input validation vulnerabilities in web applications," *Comput. Secur. Elsevier*, vol. 31, no. 3, pp. 344–356, 2012.

[6] W. Alcorn, "Cross-site Scripting Viruses and Worms - A New Attack Vector," *Netw. Secur. Elsevier*, vol. 2006, no. 7, pp. 7–8, 2006.

[7] L. Yang and D. Weng, "Snort-based Campus Network Security Intrusion Detection System Information Engineering and Applications." vol. 154, R. Zhu and Y. Ma, Eds., ed: Springer London, 2012, pp. 824-831.

[8] M. Dabbour, I. Alsmadi, and E. Alsukhni, "Efficient Assessment and Evaluation for Websites Vulnerabilities Using SNORT," *Int. J. Secur. Its Appl.*, vol. 7, no. 1, pp. 7–16, 2013.

[9] K. K. Mookhey and N. Burghate, "Detection of SQL Injection and Cross-site Scripting Attacks," *Symantec*, 2010.

[10] M. Becher, Web Application Firewalls, VDM Verlag Saarbrücken, ACM Publisher, 2007.

[11] J. Shangjie, L. Meijian, and W. Zhentao, "Research and Design of Preprocessor plugin based on PCRE under Snort Platform," Singapore, 2011, pp. 1-4.