

Rule conversion mechanism between NIDPS engines

Jang Hyeon Jeong, Si Young Lee*, Gyung Ho Park*, Seong Gon Choi

Information & Communication Engineering, Chungbuk University, Cheongju-si Chungcheongbuk-do, Korea

*Xabyss, Seongnam-si, Gyeonggi-do, Korea

wkdgus4788@cbnu.ac.kr, edward.lee70@gmail.com, pgh5247@xabyss.com choisg@cbnu.ac.kr

Abstract— This paper presents rule conversion mechanism between suricata and snort and implementation results about rule conversion. Rules of one Network Intrusion Detection and Prevention Systems (NIDPS) engine can't use without conversion in other engines. Because each NIDPS engine has different rule formats which available protocols, keywords and rule options. Therefore, rule conversion process is essential to use rule of other NIDPS engines.

Keywords— NIDPS, Suricata, Snort, Rule conversion, IPS, IDS

I. INTRODUCTION

The world is becoming interconnected with the advent of the Internet and networking technologies. There are a lot of personal, commercial, military, and government information in network. So, network security has become a major concern and users try to protect themselves from various attacks in network through firewalls and encryption mechanisms [1].

NIDPS has Intrusion Detection System (IDS) function and Intrusion Prevention System (IPS) function. IDS is system which play an important role for detecting various attacks in network [2]. IPS can analyze and detect various attacks as well as preventing intruder attacks [3]. Representative open NIDPS engines include snort, suricata and bro. NIDPS engines are being applied for real network security. In [4], Suricata is used for security in Software Defined Network (SDN). In [5], the study used snort to do port scan detection. Network security studies using open NIDPS engines are being conducted in [6]-[8].

Snort is an open source NIDPS engine and considered to be the de facto standard for NIDS [9]. Commercial NIDPS engines have problem such as significant system footprint, complex deployment and high monetary cost [10]. These problems are solved by snort. Snort consists of Packet Sniffer, Pre-Processor, Detection Engine, Alerting or Logging [11], [12]. Firstly, Packet Sniffer eavesdrops on network traffic (network backbone). Secondly, Pre-Processor checks packets to determine if the packets are for specific behavior [12]-[14]. Third, the Detection Engine takes incoming packets and matches them and rules. Fourth, when the rules match, the Alert or Logging takes place whether log to the database, or alerts the user [12], [15]-[17].

Suricata is a free and open source. Also, it is mature, fast and powerful NIDPS engine [18]. Architecture of suricata is similar to snort. Suricata has beneficial feature using multi thread (snort uses single thread) to process packets [12],[19].

So suricata maximizes the ability to receive packet. However, snort drop received packets when packets overflow capacity receiving queue [12],[20]. Also, there are differences which applying to rules and writing rules between suricata and snort. One of them is that there are protocols which can detect automatically by rule of each engine. For example, suricata can detect automatically the following application layer protocols: http, dns, ftp, smtp etc. On the other hand, snort can't detect automatically them [21].

If you want rule to match the same pattern as the new rules created by one engine, rules must be created manually in other NIDPS engines. But it must perfectly understand the rule of NIDPS engines and find the part that needs to be converted. Nevertheless, the process of rule conversion is necessary. Because when one engine recognizes a new pattern of attack, it will create and apply new rules to protect the attack. The new pattern of attacks will be threatening other engines and new rules will be required to prevent them. In order to create new rule, there are ways to analyze and create new patterns of attacks and to utilize rule made for other engines. So, we propose rule conversion mechanism between suricata and snort. The rule can detect automatically http in suricata. Authors show simple rule conversion from suricata to snort.

In chapter 2, we describe existing studies on suricata, snort and rules of them. Chapter 3 is described implementation environment for showing simple rule conversion between suricata and snort. In chapter 4, implementation that rule conversion for snort to apply suricata rule is performed. Also, implementation result is presented. Finally, chapter 5 presents conclusions.

II. RELATED WORKS

Suricata and snort are similar, but there are differences of rule between suricata and snort [12],[19],[21]. There are various rule update agents for snort and suricata. Rules of snort are updated by Sourcefire, Shared Object (SO) and proofpoint. Rules of suricata are updated by Sourcefire and proofpoint [20],[21]. Therefore, rules that each agent creates can be different. For these reasons, rules for one engine created by one agent may can't apply for other engines. To solve this problem, the rules must be analyzed and made applicable to the rules of other engines.

Rule format of suricata and snort consists largely of action, header, and rule options. The action determines what happens when rule matches. The header consists of protocol, IP addresses, ports and direction of rule. The rule options define

the specifics of the rule [21],[22]. Suricata can detect automatically application layer protocols that snort can't detect automatically. So, rules need to be changed when the rules of Suricata that snort can't detect automatically application layer protocols are used in snort. For example, if there is 'http' in the position of the protocol in snort rule, it should be changed to the 'tcp' [21]. However, just changing the protocol doesn't mean that the snort can use all changed rules.

III. IMPLEMENTATION ENVIRONMENT

Figure 1 is flowchart of proposed mechanism.

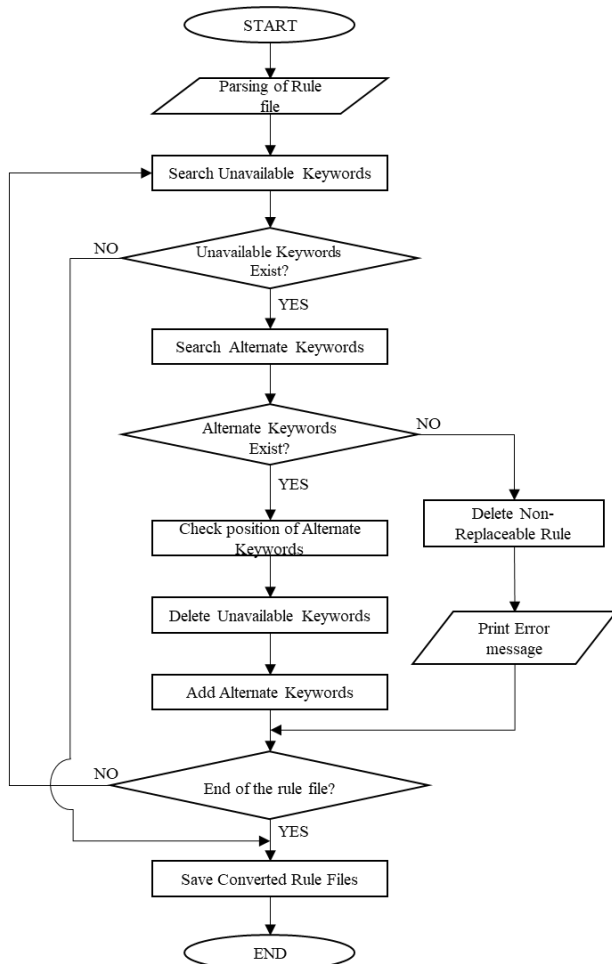


Figure 1. Flowchart of proposed mechanism

Virtual Machines (VM) are created to install the suricata and snort on the host pc. Program for creating VM is Oracle VM VirtualBox. Network of VMs is set 'Bridged Adapter'. 'Bridged Adapter' places VM on the same private network as the host pc. Therefore, the VM must be given an IP address of the private network.

Ubuntu is installed on the VM before installing the NIDPS engine. Ubuntu is Operating System (OS) on Linux-based and version of ubuntu is 18.04.3. After the OS installation, VMs are assigned IP address by DHCP. Since then suricata is installed on VM1 and snort is installed on VM2. The version of suricata is 4.1.5 and snort is 2.9.14.1.

Network is configured for implementation as shown in Figure 2.

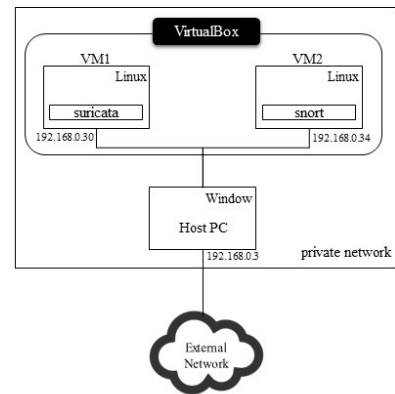


Figure 2. Network configuration for implementation

After installing suricata, you must set 'HOME_NET' in 'suricata.yaml'. 'HOME_NET' is IP address of host that user want to protect. we set 'HOME_NET' to IP address of VM1. Figure 3 is shown IP address of VM1. Figure 4, Figure 5 are shown how to set 'HOME_NET' in 'suricata.yaml'. Figure 6 is shown path of suricata rule files in 'suricata.yaml'.

```

suricata@suricata-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
    inet 192.168.0.30 netmask 255.255.255.0
  
```

Figure 3. IP address of VM1

```

suricata@suricata-VirtualBox:~$ sudo vim /etc/suricata/suricata.yaml
  
```

Figure 4. Command of modifying configuration file in suricata

```

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.0.30]"
    #HOME_NET: "any"
    EXTERNAL_NET: "!$HOME_NET"
    #EXTERNAL_NET: "any"
  
```

Figure 5. Modifying 'HOME_NET' in suricata

```

default-rule-path: /var/lib/suricata/rules
  
```

Figure 6. Position of the rule file in suricata

Likewise, snort is similar to suricata. 'HOME_NET' can be set in 'snort.conf'. We set 'HOME_NET' on IP address of VM2. Figure 7 is shown IP address of VM2. Figure 8, Figure9 are shown how to set 'HOME_NET' in 'snort.conf'. Figure10 is shown path of snort rule files in 'snort.conf'.

```

snort@snort-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
    inet 192.168.0.34 netmask 255.255.255.0
  
```

Figure 7. IP address of VM2

```

snort@snort-VirtualBox:~$ sudo vim /etc/snort/snort.conf
  
```

Figure 8. Command of modifying configuration file in snort

```

# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.0.34/24

# Set up the external network addresses.
ipvar EXTERNAL_NET any
  
```

Figure 9. Modifying 'HOME_NET' in snort

```
# Path to your rules files
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
var WHITE_LIST_PATH /etc/snort/rules/iplists
```

Figure 10. Position of the rule file in snort

IV. IMPLEMENTATION AND RESULT

Figure 11 is packet which want to detect by NIDPS engines. The packet is http response packet when user tries to access 'inform.cbnu.ac.kr'. we assume that if there is 'PHP/4' in http header, it is a threatening packet

Time	Source	Destination	Protocol
51 1.167682	203.255.89.46	192.168.0.34	HTTP

```
Frame 51: 1320 bytes on wire (10560 bits), 1320 bytes captured (10560) on interface 0
Ethernet II, Src: EfmNetwo_7a:48:5c (88:36:6c:7a:48:5c), Dst: PcsCompu_08:00:27:00:00:00
Internet Protocol Version 4, Src: 203.255.89.46, Dst: 192.168.0.34
Transmission Control Protocol, Src Port: 80, Dst Port: 59562, Seq: 351
[25 Reassembled TCP Segments (35057 bytes): #21(1448), #22(1448), #23(1448)]
Hypertext Transfer Protocol
> HTTP/1.1 200 OK\r\n
Date: Tue, 08 Oct 2019 12:53:43 GMT\r\n
Server: Apache/1.3.41 (Unix) PHP/4.4.9\r\n
X-Powered-By: PHP/4.4.9\r\n
```

Figure 11. Detection packet

A. Http Detection Rule in Suricata

```
suricata@suricata-VirtualBox:~$ sudo vim /var/lib/suricata/rules/protocol-http.rules
```

Figure 12. Command creating new rule

```
alert http $EXTERNAL_NET 80 -> $HOME_NET any (msg:"Warning protocol HTTP";content:"PHP/4";http_header;)
```

Figure 13. protocol-http.rules

```
rule-files:
- protocol-http.rules
#- suricata.rules
```

Figure 14. add protocol-http.rules in suricata

Figure 12, Figure 13, Figure 14 are shown that create and add new rule('protocol-http.rules') in order to detect packet of Figure 11. Figure 13 is rule of suricata that detects automatically http response packet and tries to match on 'PHP/4' in http header buffer.

When user try to access to 'inform.cbnu.ac.kr' while suricata is run, warning occurs as shown in Figure 15. Because 'PHP/4' is matched on the new rule in http message of http response packet.

```
10/09/2019-01:01:25.943297 [**] [1:0:0] Warning protocol HTTP [**] [Classification: (null)] [Priority: 3] {TCP} 203.255.89.46:80 -> 192.168.0.30:33772
```

Figure 15. Log of suricata

B. Http Detection Rule in Snort

In this section, snort try to detect packet such as Figure 11 by new rule of suricata (that is Figure 13). So, we make sure that snort can use rule of suricata without any rule conversion. Snort incur error when using rule of suricata and trying access to 'inform.cbnu.ac.kr' while snort is run as Figure 16.

```
ERROR: /etc/snort/rules/protocol-http.rules(1) Bad protocol: http. Fatal Error, Quitting..
```

Figure 16. Bad prorocol error

Figure 16 is shown that cause of error is that the snort can't detect automatically http. Rule conversion is essential.

C. Converted Rule in Snort

In order to using the rule of suricata in snort, it must resolve error of Figure 16. First, VM2 parses protocol-http.rules and looks for protocols, keywords and rule options that unfit to use in snort. The result is shown that VM2 detect 'http' in protocol position. Secondly, VM2 converts protocol form 'http' to 'tcp'. If the converted rule is applied here, an error is incurred as shown in Figure 17.

```
ERROR: /etc/snort/rules/protocol-tcp.rules(1) Each rule must contain a rule sid.
```

Figure 17. Lack of 'sid' error

Cause of Figure 17 is that the converted rule doesn't contain 'sid'. 'sid' keyword is used to uniquely identify rules [22]. Using 'sid' keyword is optional in suricata, but snort must use 'sid' keyword. Therefore, the third task is that VM2 looks for 'sid' keyword in 'protocol-http.rules'. If there is no 'sid' keyword, VM2 adds the 'sid' keyword and allocates value more than 1,000,000. Because over 1,000,000 has been allocated for everyone's use [22]. To compare with the existing 'protocol-http.rules', the filename is changed to 'protocol-tcp.rules' and value of 'msg' keyword is also changed. Final converted rule is shown in Figure 18. Finally, VM2 add the converted rule in 'snort.conf' as Figure 19.

```
alert tcp $EXTERNAL_NET 80 -> $HOME_NET any (msg:"Warning protocol TCP";content:"PHP/4";http_header;sid:1000001;)
```

Figure 18. Converted rule

```
# site specific rules
include $RULE_PATH/protocol-tcp.rules
```

Figure 19. add protocol-http.rules in snort

```
10/09-01:27:27.632598 [**] [1:1000001:0] Warning protocol TCP [**] [Priority: 0] {TCP} 203.255.89.46:80 -> 192.168.0.34:40036
```

Figure 20. Log of snort

When user try to access on 'inform.cbnu.ac.kr' while snort is run, warning occurs as shown in Figure 20. Because 'PHP/4' is matched to the converted rule in tcp payload of http response packet.

Both suricata and snort generate warning by matching 'PHP/4' in the http header buffer with rule of each engine. As a result, snort can detect by using converted rule that converted existing rule which can detect automatically http in suricata.

V. CONCLUSIONS

In this paper, we propose rule conversion mechanism between suricata and snort. Previously, one NIDPS engine had to analyze the rules and manually created new rules to use the rules of the other. Therefore, rule conversion process is essential to use rule of other NIDPS engines. NIDPS engines for the implementation used suricata and snort. Rule conversions are performed in the following order:

Firstly, search protocols, keywords and rule options that snort can't be used in the existing rule file. Secondly, replace the first performance result with pre-defined protocols, keywords and rule options. Thirdly, if the existing rule file doesn't have 'sid' keyword, add 'sid' keyword and the value is allocated at least 1,000,000. Finally, save converted rule file and add filename of the rule file to configuration file. It was presented as implementation result that detection is possible after the rule conversion between NIDPS engines.

ACKNOWLEDGMENT

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2018-0-01396) supervised by the IITP(Institute for Information & communications Technology Promotion). This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2019R1A2C1006167).

* Corresponding Author : S.G. Choi(choisg@cbnu.ac.kr)

REFERENCES

- [1] Bhavya Daya, *Network Security: History, Importance, and Future* Gainesville, FL, USA, 2013.
- [2] Akash Garg and Prachi Maheshwari, "A hybrid intrusion detection system: A review", 2016 10th International Conference on Intelligent Systems and Control (ISCO), Jan. 2016.
- [3] Alaa Hussein Al-Hamami, Ghossoon M. Waleed Al-Saadoon, "Development of a network-based: Intrusion Prevention System using a Data Mining approach", 2013 Science and Information Conference, pp. 641-644, Oct. 2013.
- [4] Kiho Nam, Keecheon Kim, "A Study on SDN security enhancement using open source IDS/IPS Suricata", 2018 International Conference on Information and Communication Technology Convergence (ICTC), pp. 1124-1126, Oct. 2018.
- [5] Satyendra Kumar Patel, Abhilash Sonker, "Internet Protocol Identification Number Based Ideal Stealth Port Scan Detection Using Snort", 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN), pp. 422-427, Dec. 2016.
- [6] Tomas Zitta, Marek Neruda, Lukas Vojtech, Martina Matejkova, Matej Jehlicka, Lukas Hach, Jan Moravec, "Penetration Testing of Intrusion Detection and Prevention System in Low-Performance Embedded IoT Device", 2018 18th International Conference on Mechatronics - Mechatronika (ME), Dec. 2018.
- [7] Xiangning Hou, Zhiping Jiang, Xinli Tian, "The detection and prevention for ARP Spoofing based on Snort", 2010 International Conference on Computer Application and System Modeling (ICCSM 2010), vol. 5, pp. 137-139, Oct. 2010.
- [8] Hussein Alnabulsi, Md Rafiqul Islam, Quazi Mamun, "Detecting SQL injection attacks using SNORT IDS", Asia-Pacific World Congress on Computer Science and Engineering, Nov. 2014.
- [9] Amtul Saboor, Monis Akhlaq, Baber Aslam "Experimental evaluation of Snort against DDoS attacks under different hardware configurations", 2013 2nd National Conference on Information Assurance (NCIA), pp. 31-37, Dec. 2013.
- [10] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In Lisa, volume 99, pages 229-238, 1999.
- [11] Zhimin Zhou, Chen Zhongwen, Zhou Tiecheng, Guan Xiaohui, "The study on network intrusion detection system of Snort", vol. 2, pp. 194-197, May 2010.
- [12] Wonhyung Park, Seongjin Ahn, "Performance Comparison and Detection Analysis in Snort and Suricata Environment", Wireless Personal Communications, vol. 94, issue. 2, pp. 241-252, May 2017.
- [13] G. C. Tjhai, M. Papadaki, S. M. Furnell and N. L. Clarke, "Investigating the problem of IDS false alarms: An experimental study using Snort", Proceeding of IFIP TC 11 23rd International Information Security Conference, 2008, pp. 253-267.
- [14] Rance J. DeLong, Los Gatos, "Structured exception-handling methods, apparatus, and computer program products", U.S. Patent US 6,247,169 B1, Jun. 12, 2001.
- [15] S. Chakrabarti, M. Chakraborty, and I. Mukhopadhyay, "Study of snort-based IDS," presented at the Proceedings of the International Conference and Workshop on Emerging Trends in Technology, Mumbai, Maharashtra, pp. 43-47, Feb. 2010.
- [16] Marc Norton, Daniel Roelker (2002), "Snort2.0: High Performance Multi-Rule Inspection Engine". [Online]. Available: <http://www.snort.org>, Sept. 2002.
- [17] D. Day, B. Burns, "A performance analysis of Snort and Suricata network intrusion detection and prevention engines", Proc. 5th Int. Conf. Digit. Society, pp. 187-192, Feb. 2011.
- [18] (2019) The Suricata website. [Online]. Available: <https://suricata-ids.org/>
- [19] J. Beale, A. Baker, J. Esler, T. Kohlenberg, S. Northcutt, *Snort: IDS and IPS Toolkit*, 1st ed., Jay Beale's open source security series. Syngress, 2007.
- [20] (2011) S. Damaye, "Suricata-Vs-Snort". [Online]. Available: www.aldeid.com/wiki/Suricata-vs-snort, Oct. 2. 2011.
- [21] (2019) Suricata User Guide Release 4.1.5. [Online]. Available: <https://suricata.readthedocs.io/en/suricata-4.1.5/>
- [22] The Snort Project, "SNORT Users Manual 2.9.13", Feb. 26. 2019



Jang Hyeon Jeong received B.S. degree in the College of Electrical & Computer Engineering, Chungbuk National University, Korea in 2019. He is currently a M.S. candidate in School of Electrical & Computer Engineering, Chungbuk National University. His research interests include network security, Smart Grid.



Si Young Lee received B.S. and M.S. and Ph.D degree in Computer Engineering from Kyeongbuk National University in 1995 and 1997 and 2000. He is currently CEO in Xabyss Inc. His research interest is network security.



Gyung Ho Park received B.S. degree in Computer Engineering from Kyeongbuk National University in 2002, and M.S degree in Digital entertainment from Korea Polytechnic University in 2013. He is currently CTO in Xabyss Inc. His research interest is network security.



Seung Gon Choi received B.S. degree in Electronics Engineering from Kyeongbuk National University in 1990, and M.S. and Ph.D. degree from Information Communications University, Korea in 1999 and 2004, respectively. He is currently an associate professor in College of Electrical & Computer Engineering, Chungbuk National University. His research interests include smart grid, IoT, mobile communication, high-speed network architecture and protocol.