Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 183 (2021) 837–843

Procedia Computer Science

www.elsevier.com/locate/procedia

10th International Conference of Information and Communication Technology (ICICT-2020)

Performance optimization of Snort based on DPDK and Hyperscan

Longwen Shuai[a,b], Suo Li[a,b,c,*]

[a]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100000, China
[b]Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang, 110000, China
[c]School of Mechanical Engineering, Shenyang Ligong University, 110000, China

Abstract

Snort is an open source, lightweight and widely used intrusion detection system. The detection rules are the core of Snort's detection capabilities. Snort captures and checks in real time whether the data packets meet the traffic characteristics described by a certain detection rule and triggers an alarm if it matches. Due to the insufficient packet capture capability and the performance defects of the detection engine module of Snort. It is difficult to process all arriving data packets in real time when Snort uses a large number of detection rules to process high-speed network traffic. And then it results in a high false negative rate. In this paper, we first analyzed the architecture of Snort and proposes that the key to reducing the false negative rate under high-speed network traffic is to improve Snort's packet capture capability and the performance of the detection engine module. In order to improve the performance of packet capture module of Snort, we design and implement the Snort DAQ module based on the high-performance packet processing framework DPDK. The high-performance regular engine Hyperscan is integrated into Snort in order to optimize detection engine module. Experiments show that Snort's packet capture capability and the detection rate of malicious traffic under high-speed network traffic have been greatly improved after optimization.

© 2021 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of the scientific committee of the 10th International Conference of Information and Communication Technology.

Keywords: Intrusion Detection System; Snort; DPDK; Hyperscan;

1. Introduction

With the rapid development of Internet technology, the network has penetrated into all sectors of society. However, due to the openness of the network, various security incidents occur frequently and the methods of attacks

* Corresponding author. Tel.:+86-13840002708
*E-mail address:*lisuook@qq.com

1877-0509 © 2021 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of the scientific committee of the 10th International Conference of Information and Communication Technology.
10.1016/j.procs.2021.03.007

are diverse and concealed. To solve this problem, intrusion detection system (IDS) can be used to monitor network traffic. If it detects the presence of attack behavior in network traffic, it will generate alarm or trigger other active response measures. Snort is a widely used open source intrusion detection system[1][12].

Due to the diversification of attack traffic, Snort detection rule set base becomes larger and larger. At the same time, the development of communication technology leads to the continuous growth of network bandwidth. These reasons cause Snort to face huge challenges and Snort needs more computing resources to do intrusion detection on network traffic. If the traffic bandwidth exceeds the scope of Snort's detection capability, it will result in packet loss and serious underreporting[2]. Therefore, it is of great significance to improve the performance of Snort intrusion detection system to meet the requirements of online real-time detection of high-speed network traffic.

2. Snort architecture and performance analysis

2.1. Architecture of the Snort intrusion detection system

Snort is mainly composed of five modules, including Packet capture module (Snort DAQ), parsing module, pre-processing module, detection engine and alarm output module [3-4]. Its architecture is shown in Figure 1.a.

1. packet capture module(Snort DAQ): It is responsible for capturing packets from the network card. On the underlying implementation, it uses the Libpcap library to capture packets [4].
2. parsing module: When Snort captures packets, it decodes them according to the network protocol stack. Based on link-layer protocol, packets are layered and decoded into packet data structures defined by Snort and then prepared for subsequent detection processes.
3. preprocessing module: It further processes the packet structures generated by the packet decoder so that the packets data can be processed by the detection engine.
4. detection engine module: Snort does intrusion traffic detection according to a series of detection rules. Each detection rule is used to represent the characteristics of an intrusion traffic and the behavior triggered when a packet matches the characteristics. It indicates that the packet is intruding when it matches a rule, and then triggers Snort's alarm mechanism. The core of the detection engine is feature matching, while the core of feature matching is pattern matching. Its underlying implementation is based on AC-BM algorithm and PCRE regular expression engine[5].
5. alarm output module: It outputs the results of packet inspection by the detection engine to the log file, socket or database in the form of log or alarm.

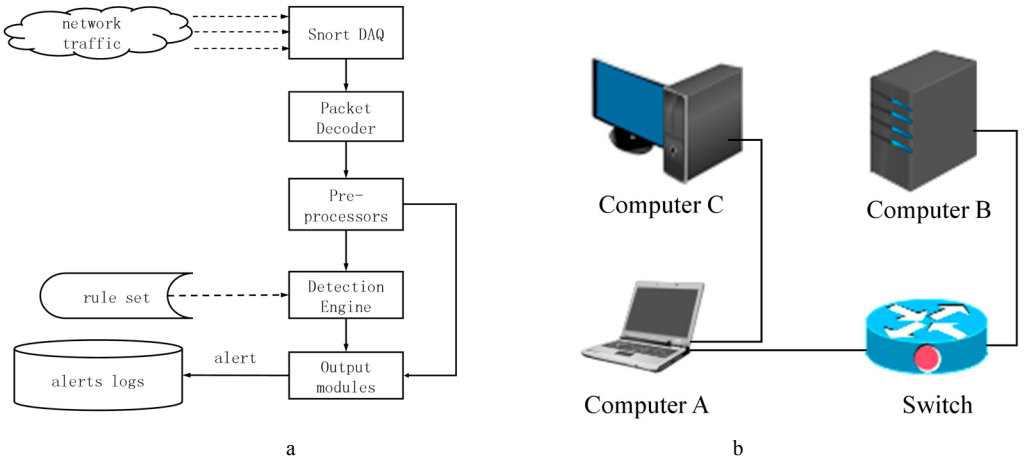

Fig. 1. (a) Snort's architecture; (b) network topology

2.2. Snort performance testing and analysis

To test Snort's alert rate against malicious traffic at different network speed. We designed the following experiment. The network topology diagram is shown in Figure 1.b.

Table 1 is the system configuration of computer A.

Table 1. System configuration of computer A.

configuration items	parameter
CPU	Intel® Core(M) I5-10210U, 2.1Ghz,12G memory, 8-core
NIC	Intel Corporation 82545EM Gigabit Controller
OS and Linux kernel	CentOS7.8, Linux kernel : 3.10.0-1127.19.1.el7.x86_64
Snort	Snort v2.9.8.2, Snort DAQ v2.0.6

Computer A has Snort installed. Computer B has TRex, an open source packet generation and delivery tool from Cisco that supports up to 200Gb/s of traffic. Computer C has packet replay software installed for sending malicious traffic. We choose including Eternalblue, Heartbleed dozens of typical malicious traffic. The packet capture module for Snort is sensitive to packet sizes. Packet capture rates of different sizes are different in high-speed network traffic environment. We chose the normal traffic including various packet sizes captured from real network environment as the background traffic in order to keep the test as close as possible to the actual working scenario of Snort. Background traffic transmission speed is adjusted by TRex and the sending speed of malicious traffic is 100kb/s. Finally, background traffic and malicious traffic packets are mixed to reach the network port Snort is listening to. The Snort alarm rate under different background traffic speed is shown in Figure 2.a.

Through experiments, it can be known that with the increase of the background traffic sending speed, Snort gradually produces false negatives. When the traffic rate reaches 480Mb/s, the false negative rate is as high as 50.63%.

Packet capture module and detection engine module is Snort performance bottlenecks. In high-speed network environment, Snort's data capture capability is insufficient and the detection engine cannot timely process captured packets, resulting in a high alarm rate of omission. Therefore, the key to Snort performance optimization is to improve the speed of Snort packet capture and pattern matching. Snort uses libpcap, a traditional packet capture platform, to capture packets by default. It adopts the Linux kernel-based bypass mechanism when capturing packets and only captures packets when the Linux kernel protocol stack processes the packets[6]. When a packet arrives, the network card transfers the packet to the receiving buffer of the network card driver through DMA. And then it triggers a hard interrupt to wake up the processor when the data transmission ends. Packets are also filtered according to filtering rules if a BPF filter is used. Then the network card driver sends the packet from the driver buffer to the kernel protocol stack buffer and then copies the packet to the upper application buffer[7].

It can be seen from the packet receiving process above that each received packet needs to trigger an interrupt. Packets need to go through many memory copies and a long processing process before they are handed over to an upper-level application. As network bandwidth moves from gigabits to gigabits, the system suffers from very high hardware and soft interrupts, system calls and context switches, and memory copy overhead, which hurts Snort's ability to capture packets. The key to improving packet capture capability is to minimize unnecessary memory copy operation, reduce or avoid system call, improve the working mode of network card and minimize the impact of interrupt on packet capture[8]. Snort also has several improved DAQ subtypes, such as Afpacket, NFQ, etc. But when working under high-speed network traffic, packet loss is still serious. To solve this problem, researchers have proposed a variety of high-performance packet processing frameworks, such as PF_ring[8], Intel DPDK[9-10], etc. Snort detection engine uses AC-BM multi-pattern matching algorithm and PCRE regular expression engine for pattern matching. This part is the core part of Snort, whose performance determines the performance of Snort. Researchers have also proposed many improvement methods for AC-BM algorithm and PCRE regular expression engine. Intel has developed a more efficient regular expression engine, Hyperscan, which can greatly improve the speed of pattern matching and is widely used in IDS/IPS, DPI and other products[11].

3. The optimization method of Snort

Through the analysis of the working principle of the Snort packet capture module and detection engine. We chose to use DPDK to replace the Libpcap library to optimize the Snort datagram capture module and use Hyperscan to optimize the Snort detection engine. The optimized Snort architecture is shown in Figure 2.b.

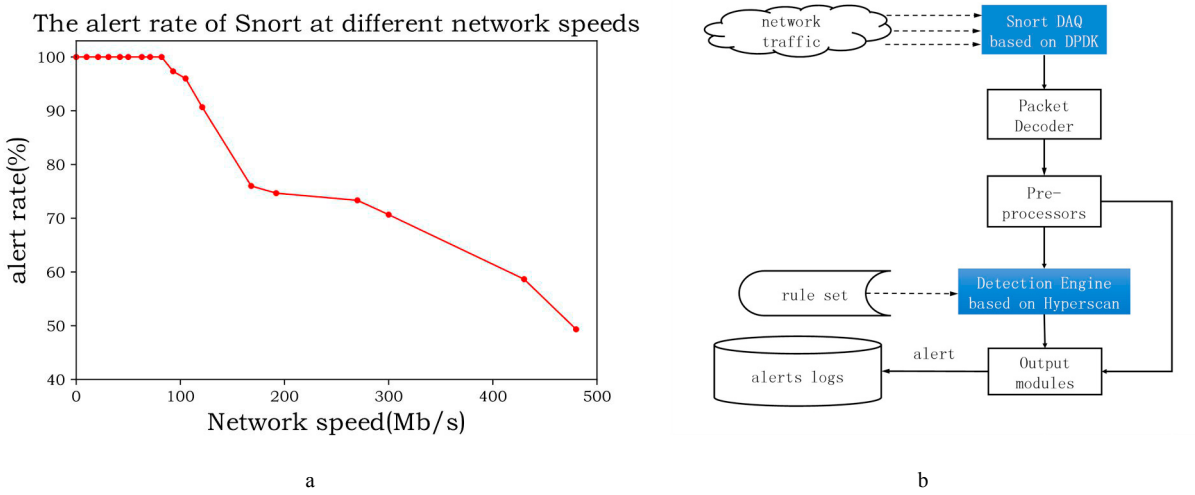

Fig. 2. (a) the alert rate of Snort at different network speed; (b) Snort architecture after optimized

4. Optimize Snort based on DPDK and Hyperscan

4.1. Optimize Snort packet capture module with DPDK

In order to solve the performance defects of the traditional Data packet capture platform and release the Data packet processing capability of multi-core IA processors. Intel and other companies developed the Data Plane Development Kit (DPDK) for Intel processors and network cards. It provides library functions and driver support for efficient packet processing in user space under IA processor architecture. DPDK focuses on high-performance packet processing in network applications[9]. It works at the user level and can replace network packet processing in traditional Linux systems, greatly improving packet processing power and throughput.

On the underlying implementation, Snort no longer calls the libpcap library function directly for packet capture since version 2.9, but calls the encapsulated DAQ subtype function interface instead. Different DAQ subtypes have different usage scenarios and environments. You can specify the DAQ type with the DAQ parameter when Snort is started. The Snort DAQ module is an abstraction layer independent of specific packet processing frameworks. It defines a common program interface that is platform independent of packet processing. This allows us to add a new DAQ subtype based on the DPDK implementation without changing Snort and other DAQ subtypes codes.

Based on the DPDK function interface and Snort DAQ interface specification, we have implemented dpdk_daq_initialize(), dpdk_daq_filter (), dpdk_daq_acquire () and other functions. The function level diagram is shown in Figure 3.

Finally, the dpdk_daq type was implemented and the Snort DAQ module was successfully modified through compilation.

4.2. Optimize the Snort detection engine using Hyperscan

Hyperscan is an open source high-performance regular expression engine from Intel that is compatible with PCRE regular expression syntax. Compared with PCRE, Hyperscan takes up more memory space while working, but has a lower CPU utilization [11]. With the PCRE regular engine, when the number of rules for Snort detection grows linearly, the performance of the detection engine decreases linearly. Increasing the traffic rate of Snort's listening network also worsens the performance degradation of the detection engine. This has led to Snort's high failure rate in the face of high-speed network traffic and a high number of detection rules. In addition, using Hyperscan for multi-string matching also provides a significant performance improvement over AC-BM[11]. The Hyperscan team also has an open source patch file for Snort. On top of that, we replaced the underlying AC-BM algorithm and PCRE regular engine of Snort with a Hyperscan patch[12], and recompiled Snort.

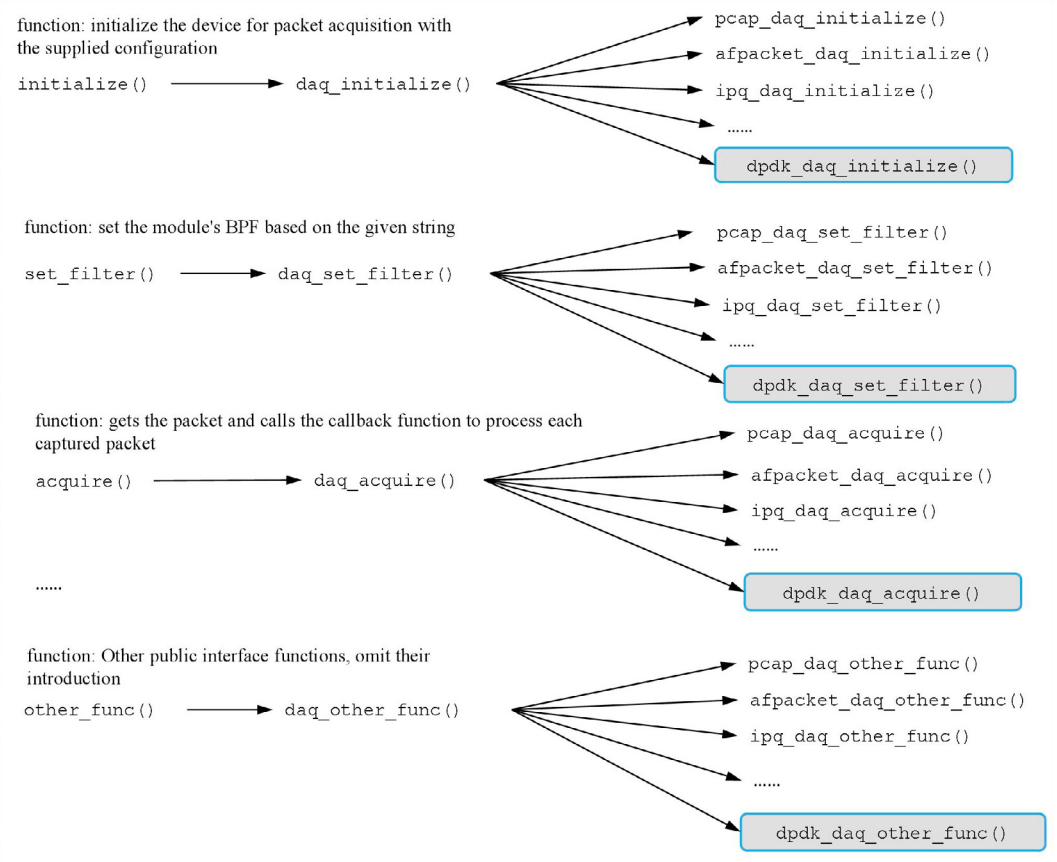

Fig. 3. Snort DAQ common interface function-level

As a result, we successfully optimized the Snort packet capture module and detection engine using DPDK and Hyperscan.

5. The experiment

Using the same experimental environment as the previous test of Snort performance, we designed two sets of experiments to test the packet capture capability of Snort before and after optimization and the omission rate of malicious traffic respectively. We used DPDK v16.07 and Hyperscan v4.5.2 during optimizing Snort.

According to the research on the traditional packet capture mechanism, packet size has a great influence on the packet capture module. In the experiment, UDP packets of 64, 128, 256 and 512 bytes were selected. TRex is used to adjust packet sending speed and number of packets to test Snort's packet capture capability. Packet capture rates of different sizes before and after Snort optimization at 100Mb/s and 1000Mb/s are shown in Figure 4.a.

Through experiments, it can be seen that Snort has a high packet loss rate for small packets in the case of high-speed traffic without taking optimization measures. As packet sizes get bigger, Snort's packet capture rate gradually increases. When the DPDK is used to optimize Snort packet capture module, 100% packets can be captured at 1000Mb/s. DPDK has significantly improved Snort's ability to capture packets.

In experiment 2, we tested Snort's omission rate at different network speeds. The environment is the same as before. The experimental results are shown in Figure 4.b.

Experimental results show that Snort has a significant decrease in the false negative rate under the condition of high-speed traffic after optimization.

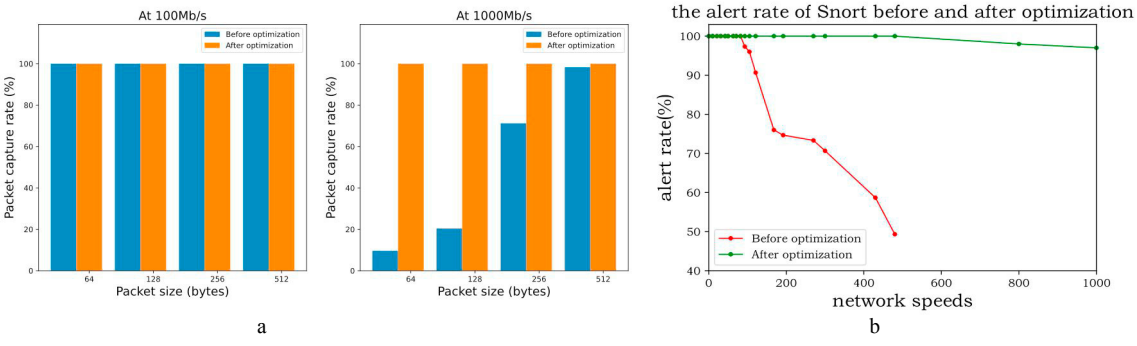

Fig. 4 (a) the packet capture rate at different network speeds before and after optimization; (b) the alert rate of Snort before and after optimization

6. Conclusion

This paper aims at the shortcomings of Snort in detecting high-speed network traffic online in real time. We first analyzed the overall Snort architecture, the performance defects of Snort packet capture module and detection engine module. DAQ subtype based on DPDK is designed and implemented to improve Snort packet capture capability. We optimized the Snort detection engine with the Hyperscan patch to speed up Snort feature matching. The experimental results show that after optimization, Snort's packet capture ability and detection rate of malicious traffic under the condition of high-speed traffic are greatly improved.

Acknowledgement

This work is supported by LiaoNing Revitalization Talents Program under Grant NO. XLYC1802112.

References

1. Day D, Burns B. A performance analysis of snort and suricata network intrusion detection and prevention engines[C]//Fifth international conference on digital society, Gosier, Guadeloupe. 2011;187-192.
2. Alka Gupta , Lalit Sen Sharma,Detecting attacks in high-speed networks: Issues and solutions, Information Security Journal: A Global Perspective 2020; 29:2,51-61
3. Chi R. Intrusion detection system based on snort[C]//Proceedings of the 9th International Symposium on Linear Drives for Industry Applications, Volume 3. Springer Berlin Heidelberg, 2014;657-664.
4. Li H, Liu D. Research on intelligent intrusion prevention system based on snort[C]//2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering. IEEE, 2010;1: 251-253.
5. Bispo J, Sourdis I, Cardoso J M P, et al. Regular expression matching for reconfigurable packet inspection[C]//2006 IEEE International Conference on Field Programmable Technology. IEEE, 2006; 119-126.

6. Jiande W F L. RESEARCH AND DESIGN OF LOW INTERACTIVE HONEYPOT TRAP BASED ON LIBPCAP[J]. Computer Applications and Software, 2009 (8): 89.
7. Li J, Wu C, Ye J, et al. The Comparison and Verification of Some Efficient Packet Capture and Processing Technologies[C]//2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). IEEE, 2019: 967-973.
8. GU R, TAN Y, JIA Y, et al. The high performance packet capture based on the PF_RING socket in Linux [J][J]. Journal of Inner Mongolia University of Science and Technology, 2007; 2.
9. Zhang D, Wang S. Optimization of traditional Snort intrusion detection system[C]//IOP Conference Series: Materials Science and Engineering. IOP Publishing, 2019, 569(4): 042041.
10. Iurman J. Speeding up Snort with DPDK for NFV cooperation[J].
11. Wang X, Hong Y, Chang H, et al. Hyperscan: a fast multi-pattern regex matcher for modern cpus[C]//16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19). 2019; 631-648.
12. https://www.hyperscan.io/2016/06/08/hyperscan-integration-snort-2-9-8-2-available