

PAPER • OPEN ACCESS

Optimization of traditional Snort intrusion detection system

To cite this article: Dongyan Zhang and Shuo Wang 2019 *IOP Conf. Ser.: Mater. Sci. Eng.* **569** 042041

View the [article online](#) for updates and enhancements.

Optimization of traditional Snort intrusion detection system

Dongyan Zhang¹, Shuo Wang^{2*}

¹School of Computer & Communication Engineering, University of Science & Technology Beijing, Beijing, 100083, China

²School of Computer & Communication Engineering, University of Science & Technology Beijing, Beijing, 100083, China

*Corresponding author : m13120375758@163.com

Abstract. With the rapid development of the Internet, the following network security issues are increasingly prominent and the increasing number of network attacks has also attracted the attention of more professionals. Network attacks are generally divided into operation attack, spoofing attack, flooding attack, redirection and so on. In order to ensure the security of computer system, intrusion detection system is designed, and people pay more and more attention to it. Firewall as the first security gate to maintain network security, intrusion detection system is undoubtedly the second security gate after the firewall. Snort intrusion detection system is a typical application of intrusion detection system. In addition, Snort is a real-time traffic analysis system that can capture and analyze packets on the network according to defined rules. However, with the continuous increase of data volume and the emergence of big data, the pattern library of Snort intrusion detection system also expands correspondingly, leading to the decrease of detection efficiency. DPDK(Data Plane Development Kit) adopts polling method to realize data packet processing, which saves CPU interrupt time, memory copy time, and provides a simple and efficient data packet processing method to the application layer, making the development of network applications more convenient. How to improve the efficiency of Snort intrusion detection system with the advantage of DPDK's high-performance packet processing is the research focus of this paper.

1. Introduction

Intrusion detection has always been one of the important topics in the field of network security. In recent years, cyber attacks are emerging one after another and there are more and more cyber attacks happening. In February 2018, more than 4,000 websites including the UK government, the US and Australia all experienced the same security issues at the same time due to their use of vulnerable third-party plug-ins. A huge number of website visitors have fallen victim to crypto robberies. In May 2018, the Russian hacking campaign has affected more than 500,000 routers worldwide. The attack spreads malware called "VPNFilter" that can be used to coordinate infected devices to create large-scale botnets. But it can also directly monitor and manipulate Web activity on infected routers. These functions can be used for a variety of purposes, including starting network operations or spam activities, stealing data, and developing targeted localization attacks. Details of the work of 30,000 Victorian civil servants were stolen in January 2019 after unknown parties downloaded parts of the Victorian government directory. The directory for government employees includes work emails, job titles and work phone Numbers. In addition, the number of customers affected by the big data breach at marriott international, the parent company of hotel giant starwood, fell from 500 million to 383



million, with more than 5 million unencrypted passport numbers and about 8.6 million encrypted credit card Numbers stolen. The breach is one of the biggest breaches of personal data in history. Marriott hotels said starwood had been hacked since 2014.

1.1. Traditional snort intrusion detection system

Snort consists of several major software modules, including packet sniffer, preprocessor plug-in, detection module and alarm output module. The pre-processing and detection module enables snort's detection engine to detect the contents of packets more effectively, and the alarm output plug-in can output alarm information in a variety of ways[1].

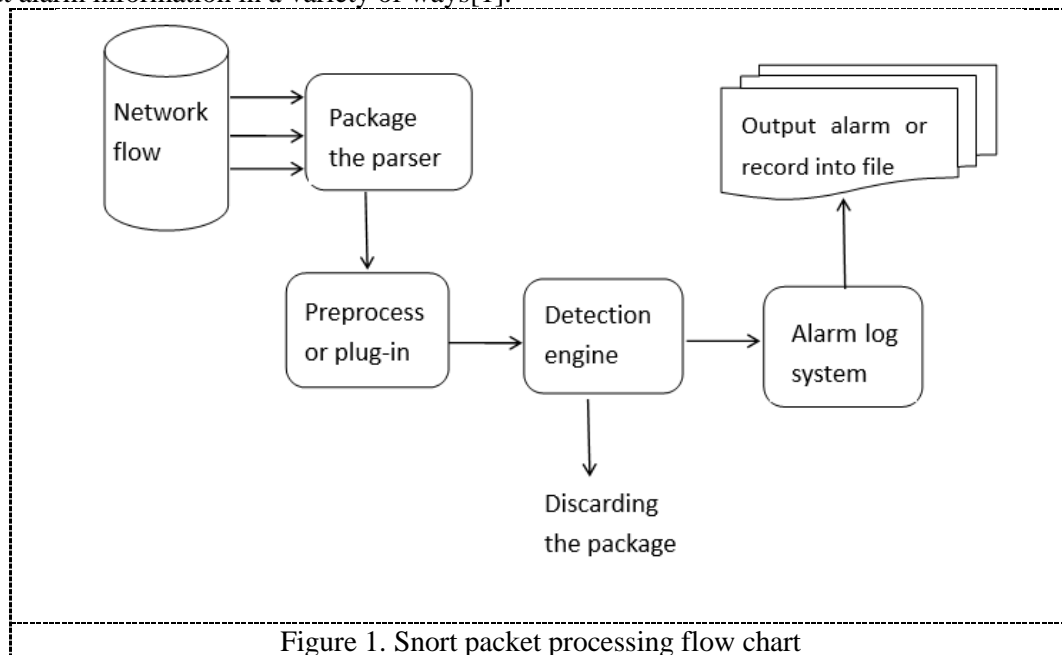


Figure 1. Snort packet processing flow chart

The most basic function of Snort is a packet sniffer. But a packet sniffer is just the beginning of what snort is doing. Snort as packet sniffer capture packets from the network card, and then the packets after a packet parser fills in the data link layer protocol package structure in the body, then the packets are sent to the preprocessor plug-in[2]. Then, the packet structure passes through all the rule chains in the detection engine for one-time detection. If there is a data chain that conforms to the rules, the packet will be detected. Finally, the detection result is sent to the output module, and the data packet is either discarded or a log alarm is generated.

1.1.1. Traffic capture module. Before snort2.9.0, the traffic capture module used the Libpcap library to capture packets. The Libpcap library is a network driver for Linux /UNIX. Libpcap was originally written for TCPdump to run at the data link layer and read packets directly from the network card, regardless of the card or driver. After snort2.9.0 version introduces DAQ (Data Acquisition) library, the module is actually an abstraction layer for message processing services. DAQ replaces a direct call to the libpcap function with an abstraction layer that allows snort to operate on a variety of hardware and software interfaces. Snort calls the underlying function library through DAQ and gets packets from the network[2].

1.1.2. Package parser. After snort captures packets using DAQ, they are sent to the packet parser in their original state and decoded. Packet resolver resolves network packets into the packet structure defined by snort to prepare for subsequent preprocessing and detection engine analysis.

1.1.3. Preprocessor. The work of preprocessor is very important for the detection engine of any intrusion detection system to analyze data according to rules. When snort receives packets, the main

probe engine can't process them or apply rules to them. A preprocessor is a component or plug-in that performs such operations on packets from a packet parser, with the goal that the data can be processed by a probe engine. In addition, some preprocessors can do other things, such as some obvious errors in the probe packet header.

1.1.4. Rule parsing and probing engine. A probe engine is the most important part of snort. The detection engine can be divided into two parts: rule creation/translation and rule-based detection engine.

1.1.5. Alarm output module. The snort data checked by the detection engine needs to be output in some way. Depending on what the detection engine finds in the package, the alarm output system is used to generate either record behavior or generate an alarm. The output module can perform different actions according to the specified way of saving log and alarm system to generate output information.

1.2. DPDK(Data Plane Development Kit)

DPDK consists of environment abstraction layer, ring buffer management, memory pool management, network message buffer management, timer management and other core components, providing a clean and complete framework[7].

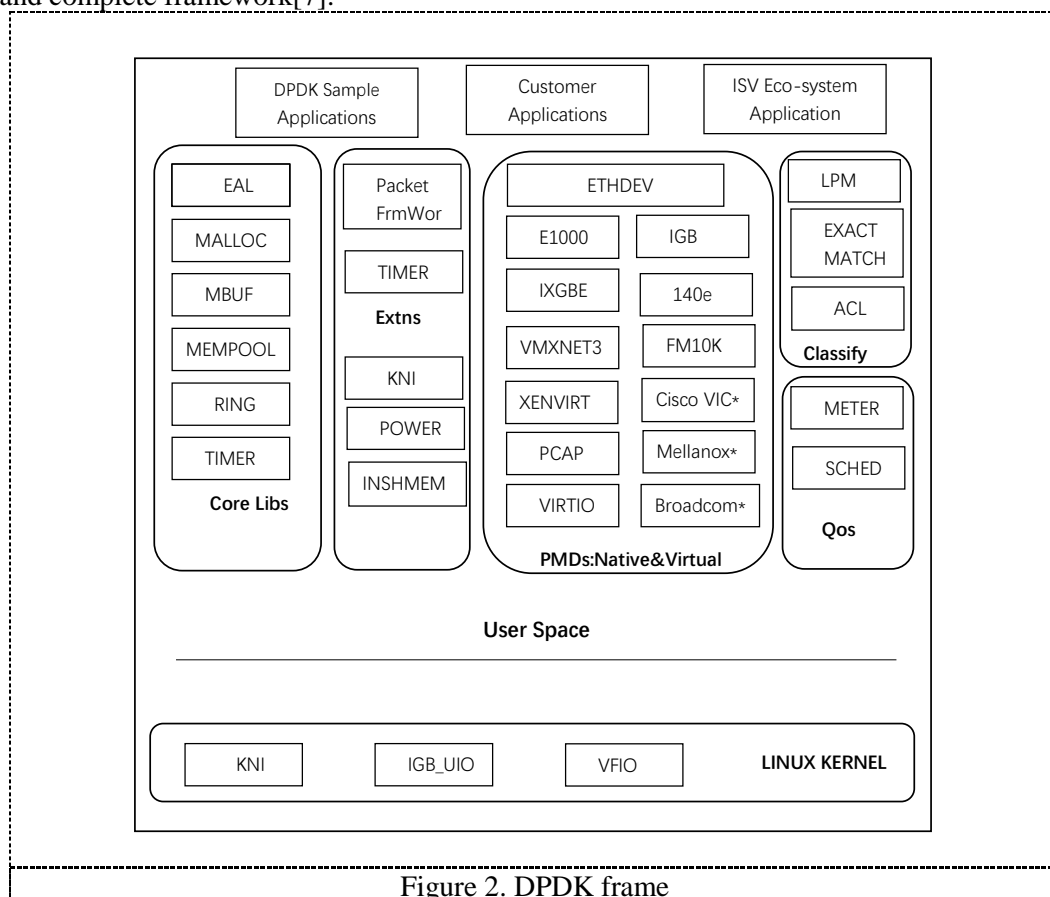


Figure 2. DPDK frame

Environment abstraction layer (EAL): it provides DPDK components and application interface, abstracts the underlying framework and hides the shape of hardware. The environment abstraction layer has major functions: DPDK loading and starting, support for multi-threading and multi-process execution, CPU affinity Settings, system memory allocation and release, atomic operations, timer references, PCI bus access, trace and debug functions, memory management, warning operations, interrupt handling, and so on. Ring buffer management (librte_ring): Ring data structure provides a lockless, multi-producer and multi-consumer FIFO table processing interface. Its advantage is that it is

easier and faster to deploy and suitable for a large number of operations. Memory POOL management (librte_mempool): the primary responsibility of memory POOL management is to allocate a POOL of a specified number of objects in memory. Each POOL is uniquely identified by name and USES a ring to store idle object nodes. It also provides a number of other services such as per-core backup caching of object nodes and automatic alignment to ensure that elements are balanced on per-core memory channels[9]. Network message buffer management (librte_mbuf): the message cache manager provides the ability to create and release message caches that DPDK applications might use to store messages. Messages are typically created and stored from the DPDK MEMPOOL library at the start of the program. The BUFF library provides an API for packet release applications. In general, message BUFF is used to cache common messages and packet BUFF is used to cache network packets. Timer management (librte_timer): this library location DPDK execution unit provides timing services to support asynchronous execution of functions. Timers can be set to cycle calls or called only once. EAL provides an interface for high-precision clock acquisition and initialization on demand at each core[7].

2. The background

Under the traditional data system packet capture mode, there are three main factors affecting the packet capture performance. Frequent resource allocation and release: the system allocates a buffer and a packet descriptor for each packet that arrives, and does not release the data until it is delivered to the user-mode space[5]. Copy of data between user and kernel states: the common packet capture platform reads the data packets through system calls, while the copy of the data packets between user and kernel states consumes a lot of CPU. Network card interrupt: each packet will trigger an interrupt after it reaches the network card cache. Such frequent interruption will make other processes unable to obtain CPU resources, and the received packets in the cache area will not be timely processed, which will result in a large number of packets lost[10].

2.1. Libpcap packet processing

Libpcap is a typical packet capture platform, which creates a RawSocket to capture all incoming packets from the network card and store them in msgbuff. Libpcap works by first setting the network card to layer promiscuous mode, so that the network card can receive all the packets flowing through the host network and then copy them out for preliminary processing. BPF filters the copied packets to reduce system overhead. Finally, the filtered data packets are delivered to the LINUX kernel buffer, and the user layer application program takes the data packets out of the kernel buffer and places them in the user space memory area for further processing. The Libpcap package process has multiple copy operations and interrupt responses, which consumes a lot of CPU time slices and prevents other processes from obtaining CPU resources.

2.2. DPDK packet processing

DPDK packet processing uses polling to achieve packet processing, which saves CPU interruption time and memory replication time. DPDK provides a simple and efficient packet processing method for application layer, which makes the development of network applications more convenient [15].

The network card establishes a physical connection with the monitoring link and receives network packets. The DPDK PMD driver initializes the network card, configures the packet to capture queues, query intervals and allocates the packet reception buffer [18]. DPDK packet capture library calls the network card query function provided by PMD to check whether the package has been received. If a packet is received, it will be sent to the captured proxy server according to the data flow. Finally, the data packet is stored in the cache and sent to the flow feature analysis program for analysis and storage.

3. Snort intrusion detection system design

Based on the traditional snort intrusion detection system, our system introduces DPDK to optimize the system to make up for the deficiency in the performance of the traditional snort intrusion detection system[17]. DPDK-based snort intrusion detection system can run each network card in DPDK independently by binding the network card with thread id. The DPDK architecture is added as the efficiency of accepting package. The system creates the function library set for different working environment by creating the EAL(Environment Abstraction Layer), after which the developer can connect their application and the function library[19]. DPDK's environment abstraction layer hides the details of the underlying environment from the application and function libraries and can be extended to any processor. The Environment Abstraction Layer (EAL, Environment Abstraction Layer) of DPDK is mainly responsible for the access of the underlying computer resources (such as hardware and memory space), and the implementation of the implementation details of the interface provided to the user. Its initialization routines determine how these resources are allocated (PCI devices, timers, consoles, and so on)[18].

After the intrusion detection in snort, the filtered information is saved in the log file. After snort2.9.0, the log file cannot be directly stored in the MySQL database. Instead, the packet can be stored in the MySQL database table through barnyard2 conversion. Barnyard2 is an open source interpreter for Snort unified2 binary output files. Its main purpose is to allow Snort to write to disk in an efficient way, separating the tasks of parsing binary data into various formats. It doesn't cause Snort to miss out on network traffic. ACID Analysis Console for Intrusion Databases was used to display the detection data in the database in PHP pages[20,21].

4. The experiment

In this paper, a lot of experiments have been done on the traditional snort intrusion detection system and the improved intrusion detection system based on DPDK. Experiments show that packet size and transmission rate has a direct impact on the performance of the system. In this experiment, the sender is used to send quantitative data packets. Data packets are sent at rates of 100Mbps and 1000 Mbps. The packet sizes are 64 bytes, 128 bytes, 256 bytes, and 512 bytes. The number of packets is 100000. Finally, we get the packet receiving rate in the traditional snort intrusion detection system and the packet receiving rate in the DPDK intrusion detection system.

The receiving rate of the packet is expressed in P, the captured data packet is represented in TC and the transmitted data packet is represented in TS. The acceptance rate of data packets is expressed by the following formula.

$$P = \frac{TC}{TS} \quad (1)$$

4.1. 100Mbps speed under the contract experiment

At a speed of 100Mbps, 100000 packets are sent to view the packets detected by the traditional snort intrusion detection system and DPDK-based intrusion detection system respectively.

Table 1. At a speed of 100Mbps, percentage of packets detected by the system.

Packet size	number of packets sent	Snort IDS number of packets caught	DPDK IDS number of packets caught	Snort IDS packet receiving rate(%)	DPDK IDS packet receiving rate(%)
64	100000	100000	100000	100	100
128	100000	100000	100000	100	100
256	100000	100000	100000	100	100
512	100000	100000	100000	100	100

4.2. 1000Mbps speed under the contract experiment

At a speed of 1000Mbps, 100000 packets are sent to view the packets detected by the traditional snort intrusion detection system and DPDK-based intrusion detection system respectively.

Table 2. At a speed of 1000Mbps, percentage of packets detected by the system.

Packet size	number of packets sent	Snort IDS number of packets caught	DPDK IDS number of packets caught	<i>Snort IDS</i> <i>packet receiving</i> <i>rate(%)</i>	<i>DPDK IDS</i> <i>packet receiving</i> <i>rate(%)</i>
64	100000	8655	100000	8.6	100
128	100000	18447	100000	18.4	100
256	100000	69412	100000	69.4	100
512	100000	98896	100000	98.9	100

It can be seen from table 1 and table 2 that at the same transmission rate, the larger the packet is, the more packets are detected by the intrusion detection system. At the same packet size, the faster the transmission rate, the fewer packets detected by the intrusion detection system, and vice versa. At 100Mbps, both the traditional snort intrusion detection system and DPDK-based intrusion detection system can completely detect all packets sent. When the transmission rate is 1000Mbps, the performance of traditional snort intrusion detection system in detecting packets is far less than that of DPDK-based intrusion detection system. At 1000Mbps, the DPDK-based intrusion detection system can still detect all sent packets completely.

5. Conclusions

In this paper, through the comparison of experiments, we can see that the traditional snort intrusion detection system can basically meet the requirements at the speed of 100Mbps, but it shows obvious deficiencies at the speed of 1000Mbps. DPDK-based intrusion detection system optimizes the traditional snort intrusion detection system to solve the problem of insufficient snort performance. DPDK-based intrusion detection system realizes all-packet detection at 1000Mbps.

In addition, the research of intrusion detection system is still the focus of the field of network security. In the face of 10000Mbps or even higher speed network flow, how to use intrusion detection system to solve the current needs makes us to consider the problem and research direction.

References

- [1] Jianying Liang. (2015). Based on the snort network intrusion defense explored. *Cyberspace security*, 6 (2), 37, 38.
- [2] Peng Lu. (2013). A lightweight intrusion detection system snort. *Silicon valley* (4), 57-57.
- [3] Weifeng Huang. (2013). Methods and detection of network intrusion. *Electronic technology and software engineering* (21), 228-229.
- [4] Zaiyi, P. (2018). Network security situation analysis based on a dynamic bayesian network and phase space reconstruction. *The Journal of Supercomputing*(3), 1-16.
- [5] Wang, Q., & Zailin, P. (2010). Research on Network Attack and Detection Methods. *International Workshop on Education Technology & Computer Science*.
- [6] Zhen, L., Pan, H., Liu, W., Fei, X., Cao, Z., & Gang, X. (2016). A network attack forensic platform against http evasive behavior. *Journal of Supercomputing*, 73(7), 1-12.
- [7] Intel Corporation(2014), Packet Processing on IntelR Architecture, <http://intel.com/go/dpdk>.
- [8] Pongrácz, G., Molnár, L., & Kis, Z. L. (2013). Removing Roadblocks from SDN: OpenFlow Software Switch Performance on Intel DPDK. *Second European Workshop on Software Defined Networks*.
- [9] Haozhe Ren, & Mei Nian. (2018). Dpdk-based high-speed packet acquisition method. *Computer system applications*, 27(6), 242-245.
- [10] Park, W., & Ahn, S. (2017). Performance comparison and detection analysis in snort and suricata environment. *Wireless Personal Communications*, 94(2), 241-252.
- [11] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, & E. Vázquez. (2009). Anomaly-based network intrusion detection: techniques, systems and challenges. *Computers & Security*, 28(1-2), 18-28.

- [12] None. (2014). The practice of network security monitoring. *Network Security*, 2014(10), 4.
- [13] Chakrabarti, S. , Chakraborty, M. , & Mukhopadhyay, I. . (2010). [acm press the international conference and workshop - mumbai, maharashtra, india (2010.02.26-2010.02.27)] proceedings of the international conference and workshop on emerging trends in technology - icwet '10 - study of snort-based ids. 43.
- [14] Tjhai, G. C. , Papadaki, M. , Furnell, S. M. , & Clarke, N. L. . (2008). Investigating the problem of IDS false alarms: An experimental study using Snort. *Proceedings of The Ifip Tc 11 23 rd, International Information Security Conference*. Springer US.
- [15] Ning Zhao, & Shucui Xie. (2016). Analysis and application of dpdk-based efficient packet acquisition technology. *Computer engineering and science*, 38(11), 2209-2215.
- [16] Kuvaiskii, D., Chakrabarti, S., & Vij, M. (2018). Snort intrusion detection system with intel software guard extension (intel sgx).
- [17] LI Weicheng. (2016). Based on the research of DPDK message acquisition system and the implementation. (Doctoral dissertation).
- [18] Rajesh, R. , Ramia, K. B. , & Kulkarni, M. . (2015). Integration of LwIP Stack over Intel^(R) DPDK for High Throughput Packet Delivery to Applications. *Fifth International Symposium on Electronic System Design*. IEEE.
- [19] Kai, L. I., Lin, Y. E., Xiangzhan, Y. U., & Yang, H. U. (2017). Traffic dynamic load balancing method based on dpdk. *Intelligent Computer & Applications*.
- [20] Garg, A. , & Maheshwari, P. . (2016). Performance analysis of Snort-based Intrusion Detection System. *2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE.
- [21] Sagala, A. . (2016). Automatic SNORT IDS rule generation based on honeypot log. *International Conference on Information Technology & Electrical Engineering*. IEEE.