Research on High Performance Intrusion Prevention System Based on Suricata

Jian Guo [1], Hua Guo [2], Zhong Zhang [3]

[1] Dept. of Information management and information system, Tianjin University of Technology, Tianjin, China

[2] Dept. of Communication Engineering, Tianjin Polytechnic University, Tianjin, China

[3] Div. of Science & Technology, Tianjin University, Tianjin, China

Abstract. Suricata is an open source, high-performance network IDS, IPS and network security monitoring engine. Based on Suricata and AF-PACKET technology, this paper research on the Suricata IPS applied to Huawei Kunpeng 920 CPU and Galaxy Kylin operating system, designs defense rules for common network threats at present, and tuning the performance of Suricata IPS in a high-traffic network environment. Using Ixia network tester, the results show that the design scheme can fully adapt to the relevant hardware system and software environment, the network throughput can reach 20Gbps.

Keywords: Suricata, IPS, Network Security, High-traffic Network.

1. Introduction

Suricata is a high-performance network IDS (Intrusion Detection System), IPS (Intrusion Prevention System) and network security monitoring engine [1]. Suricata is open source, developed and owned by a community-run non-profit foundation, OISF (Open Information Security Foundation). The Suricata community is quite active. Since 2015, Suricata conferences (SuriCon) have been held in different cities around the world every year.

At present, there are many open sources and mature IDS/IPS systems, such as Snort [2], Suricata, Bro (zeek) [3], etc. Suricata stands out due to the following characteristics:

1). free, open source, high performance, stable

2). rely on powerful extensible rules to filter network traffic, and support LUA scripting language

3). supports multiple operating systems and network protocols

4). supports multi-threading, multiple packet capture modes and high-speed regular expression matching engine hyperscan

At the same time, Suricata has been integrated with the current new technologies. Through data mining, machine learning, deep learning and big data analysis, IDS has opened up new fields in the two directions of intelligence and distribution. Therefore, there are many large-scale businesses are using Suricata.

This paper will research on a high-performance intrusion prevention system [4] based on Suricata for the actual network environment of a large enterprise. First, adapt Suricata on the server using Huawei Kunpeng 920 CPU and Galaxy Kylin operating system; then, research precise defense rules for more than two hundred of common network attacks; finally, 20Gbps of HTTP network traffic mixed with attacks are used for the performance tuning of Suricata.

2. System Hardware and Software Environment

In the actual network environment of the enterprise, the server we use is Tsinghua TongFang K620-M3. The CPU of this server is Huawei Kunpeng 920, with a total of 64 cores, a single core running frequency of 2.6 GHZ, and supports LUMA architecture. The server has 384 GB memory, 480G SSD hard disk, two Huawei SP333 network cards each with two ports. The operating system is Galaxy Kylin Advanced Server V10. Suricata is developed based on the version of 5.0.6.

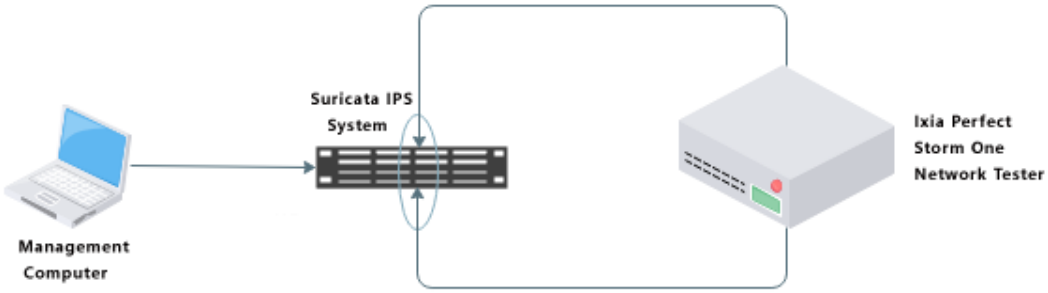

Figure 1. System Architecture Diagram

The network tester we use is IXIA PerfetStorm One from Keysight Technologies Corporation. The tester has 8 optical fiber network ports, and the maximum network test traffic can reach 40 Gbps.

Figure 2. IXIA PerfetStorm One network tester

3. R&D Process of Suricata IDS System

3.1 Environment Adaptation

Before developing Suricata's defense rules, we adapted suricata to the hardware and software environment of the system.

First, we install the dependent packages related to the Galaxy Kylin operating system and the Kunpeng 920 CPU for Suricata. These packages include libpcap, pcre, libyaml, libnetfilter, numactl, etc., as well as language environments such as Rust, Lua, and Python [5]. Then, the high-speed regular expression matching engine Hyperscan [6] is needed, as well as basic defense rules. During the adaptation process of Suricata, we encountered a lot of incompatibility and version matching problems. After many code modifications, compilations and tests, we successfully installed and ran the main program of Suricata on the server.

3.2 Developing Defense Rules

The team first wrote relevant defense rules based on the actual attack packets collected by the enterprise and the vulnerability of CVE2020. First, use Wireshark software to analyze the TCP/IP protocol of the attack packets, find out the characteristic values of the attack data packets, and then write the corresponding defense rules according to the characteristic values. We also use Hyperscan to perform multi-pattern scanning to find attack packets appearing in different streams. Defense rules are written based on Suricata's syntax and are compatible with Snort IDS. Below are the defense rules against two worms attack, CVE 2004-0362 and MSB MS02-093.

drop udp any any -> any 1434 (msg:"180 All-2 Microsoft SQL Server Slammer/Saphire Worm MSB MS02-093"; content: "|04|"; depth: 1; content: "sock"; content: "send"; distance: 6; within:4; sid:180; rev:1;)

drop http any any -> any any (msg:"181 All-3 Code Red Worm CVE 2004-0362"; flow: established; content:"|85 4c fe ff ff 8b 8d 64 fe ff ff 0f be 11 85 d2 74 02|"; content: "GET"; http_ method; content:"/default.ida"; http_uri; depth:12; sid:181; rev:1;)

Since the enterprise network environment requires suricata to run in IPS mode, the keyword of the rules should be drop, and if it is the IDS mode, the keyword should be alert. After the attack test of Ixia PerfectStorm One, the defense rules we wrote can achieve an interception rate of 98.5%, and the false interception rate is close to 0.

3.3 Selecting Runmode of Suricata

Suricata has multiple runmodes, including single, workers, autofp, etc. In these three modes, Suricata has the same functions such as packet capture, intrusion detection, and logging from the network card queues. But the workers mode has the better performance, which is also our choice.

In workers mode, each working thread handles all tasks such as packet capture, packet decoding, flow table tracking, session tracking, virus detection, intrusion detection, logging, and output. On the contrary, in autofp mode, there are multiple threads that capture packets, and after the packets are decoded, they are sent to the stream processing thread. This will increase the CPU's workload for stream processing thread balancing, thereby affecting system performance. The two runmodes are shown in the figure below.

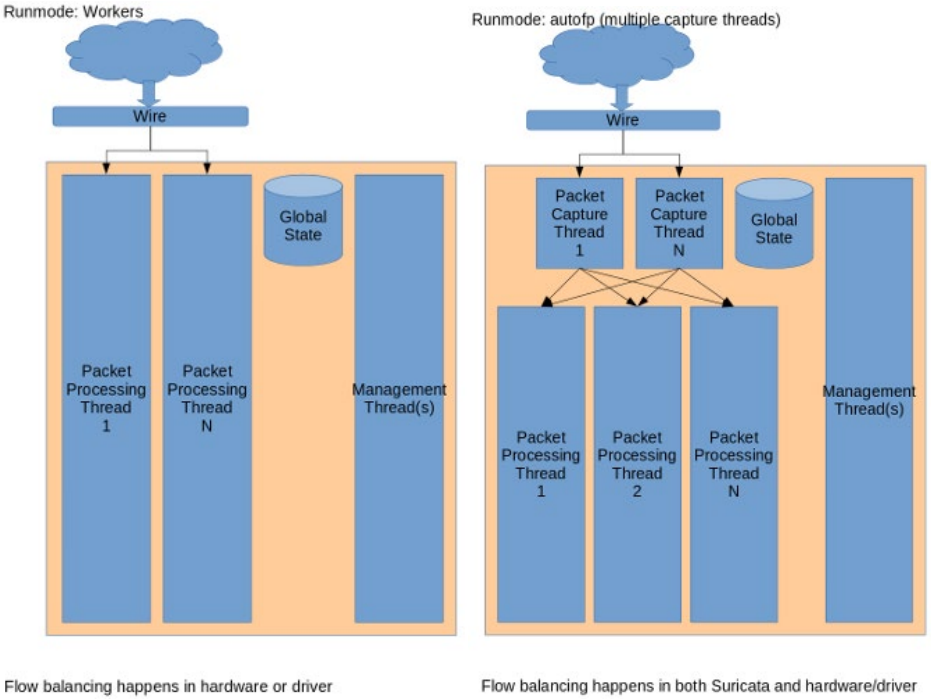

Figure 3. Schematic of Workers and Autofp runmodes

3.4 Selecting Packet Capture Method

Suricata has a variety of network packet capture methods, including AF_PACKET, PF_RING [7] DPDK [8], Netmap [9], etc. During the development process, we chose the AF_PACKET mode.

AF_PACKET establishes a software bridge between two interfaces by copying packet from one interface to another (and reverse). The specific implementation modes of AF-PACKET include Socket and PACKET_MMAP. The default is PACKET_MMAP and the speed is faster. The principle of PACKET_MMAP is to use the shared memory method to allocate a kernel buffer in the kernel space, and then the user space program calls mmap to map to the user space. Copy the received packets data to the kernel buffer, so that the user space program can directly read the captured packets data. PACKET_MMAP reduces system calls, and the captured message can be read without using Socket's recvmsg. Compared with the Socket method, it reduces one copy and one system call.

Although the current DPDK packet capture method can bypass the operating system kernel and has an advantage over AF-PACKET in terms of speed, AF_PACKET is simple to configure, does not require DPDK network card drivers, adapts to most server network environments and network cards, and, in the PACKET_MMAP mode of AF_PACKET, the performance of supporting large network traffic is no less than that of DPDK. So we choose AF-PACKET capture method.

4. Suricata IDS Performance Tuning

In the current large-scale enterprise network environment, IPS and IDS often face normal HTTP traffic of more than 10Gbps, which is mixed with various worm and Trojan attacks. Based on this actual demand, our team used the Ixia PS One network traffic tester to test the previously designed Suricata IPS system with high traffic. However, before the performance of the system was optimized, Suricata began to experience packet loss and instability when faced with network traffic of around 1Gbps. Based on this, the R&D team optimized the performance by adjusting the network card queue, kernel parameters, NUMA and CPU affinity settings, and stream engine parameters, finally made the system throughput approaching 20Gbps.

4.1 RSS Tuning

RSS (Receive Side Scaling) is a technique used by network cards to distribute incoming traffic over various queues on the NIC. We used RSS technology to adjust the number of queues of the network card, the number of queues of the network card is very important for the performance of Suricata.

Many network cards use the ethtool tool to adjust the number of RSS queues of the network card [10], Huawei SP333 NIC also has this tool. First we should make sure irqbalance is off and not running, then the number of RSS queues is set to 32, the number is equal to the cores of Kunpeng 920 CPU in one NUMA node. In this configuration, the best packet capture performance on the NIC side can be obtained. The commands are as follows.

```
service irqbalance stop
ifconfig eth1 down
/usr/local/sbin/ethtool -L eth1 combined 32
/usr/local/sbin/ethtool -K eth1 rxhash on
/usr/local/sbin/ethtool -K eth1 ntuple on
ifconfig eth1 up
```

4.2 OS Kernel Parameter Tuning

The kernel parameters of the Kylin Linux OS also have a great impact on the performance of the system, which is mainly reflected in the size of the buffer allocated to the Socket and TCP buffers. We have made the following adjustments to the kernel parameters of the Galaxy Kylin operating system, so that it can fully utilize the capabilities of the system hardware [11].

```
sudo sysctl net. Core. rmem_default=73400320
sudo sysctl net.core.wmem_default=73400320
sudo sysctl net.core.rmem_max=134217728
sudo sysctl net.core.wmem_max=134217728
sudo sysctl net.ipv4.tcp_rmem="100000000 100000000 100000000"
sudo sysctl net.ipv4.tcp_wmem="100000000 100000000 100000000"
sudo sysctl net.ipv4.tcp_mem="100000000 100000000 100000000"
sudo sysctl net.core.netdev_max_backlog=300000
sudo sysctl net.ipv4.tcp_no_metrics_save=1
sudo sysctl net.ipv4.tcp_congestion_control=htcp
sudo sysctl net.ipv4.tcp_mtu_probing=1
sudo sysctl net.core.netdev_budget=3000
sudo sysctl -p
```

4.3 CPU Affinity and NUMA Settings

Non Uniform Memory Access (NUMA) is a computer memory design used in multiprocessing, where the memory access time depends on the memory location relative to the processor[12]. Under NUMA, a processor can access its own local memory faster than non-local memory (memory local

to another processor or memory shared between processors). The benefits of NUMA are limited to particular workloads, notably on servers where the data is often associated strongly with certain tasks or users. Schematic diagram of the structure of NUMA is shown as follows.

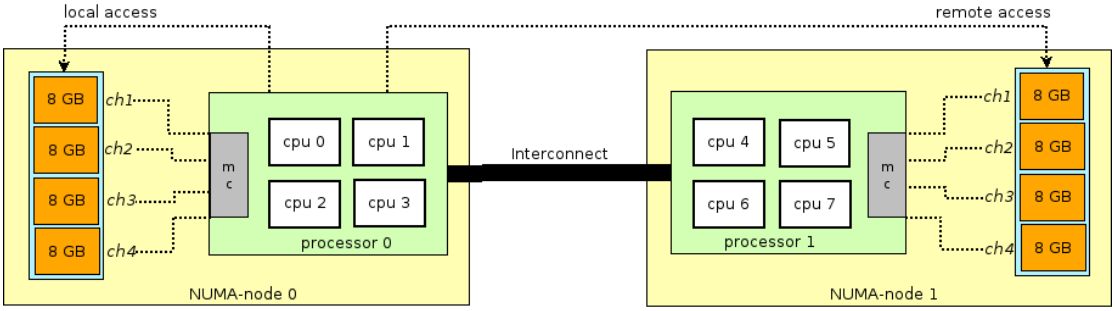

Figure 4. Structure of the NUMA nodes

The Huawei Kunpeng 920 CPU of the Tongfang server we use has a total of 64 cores, which are divided into 2 NUMA nodes, and each NUMA node has 32 cores.

First, all worker threads of suricata can be bound to the 32 cores in the first NUMA node via the numactrl or taskpc commands. The command is:

numactl –cpunodebind=0 /usr/bin/suricata -c /etc/suricata.yaml --af-packet –runmode=workers

or:

numactl –physcpubind=0-31 /usr/bin/suricata -c /etc/suricata.yaml --af-packet –runmode=workers

This ensures that all threads of Suricata run inside the First NUMA node.

Then, by setting the CPU affinity of the network card, you can bind the four network interfaces of the two Huawei SP333 network cards to the 32 CPU cores in the first NUMA node, which ensures that the work of the two network cards is also running inside the first NUMA node. The 4 network interfaces are enp1s0f0, enp1s0f1, enp4s0f0, enp4s0f1. The setting command is:

./set_irq_affinity_cpulist.sh 1-7 enp1s0f0
./set_irq_affinity_cpulist.sh 9-15 enp1s0f1
./set_irq_affinity_cpulist.sh 17-23 enp4s0f0
./set_irq_affinity_cpulist.sh 25-31 enp4s0f1

In this way, Suricata's worker threads, CPU cores, network cards, and memory can all be within one NUMA node, which can minimize cross-NUMA access and improve system performance of network throughput.

4.4 Suricata's Stream Engine Parameters Tuning

By adjusting Suricata's Flow and Stream engines parameter, the memory and buffer size used by Suricata can be increased, also the number of pre-allocated network connections can be increased, thereby improving Suricata's network packet processing performance. The specific settings are in Suricata's configuration file Suricata.yaml, which is shown as follows.

```
flow:
  memcap: 25gb
  hash-size: 500000
  prealloc: 300000
  emergency-recovery: 30
  managers: 5
  recyclers: 5
```

```
stream:
  memcap: 25gb
  checksum-validation: no
  inline: auto
  prealloc-sessions: 200000
  bypass: yes
  reassembly:
    memcap: 30gb
      depth: 128kb
```

In addition, reducing the timeout seconds for network connections in the suricata.yaml configuration file will also improve system performance.

4.5 Suricata's AF-PACKET Parameters Tuning

To achieve a system network throughput of 20Gbps, the AF-PACKET section needs to be set in the Suricata.yaml configuration file. First connect the two network interfaces enp1s0f0 and enp1s0f1, the throughput can reach 10Gbps, and then connect the two network interfaces enp1s0f0 and enp1s0f1, the throughput can also reach 10Gbps. The total throughput of the 4 network interfaces is 20Gbps.

```
af-packet:
  - interface: enp1s0f0
    threads: 16
    defrag: no
    cluster-type: cluster_flow
    cluster-id: 98
    copy-mode: ips
    copy-iface: enp1s0f1
    buffer-size: 64535
    use-mmap: yes
    ring-size: 200000
    block-size: 1048576
  - interface: enp1s0f1
    threads: 16
    defrag: no
    cluster-type: cluster_flow
    cluster-id: 95
    copy-mode: ips
    copy-iface: enp1s0f0
    buffer-size: 64535
    use-mmap: yes
    ring-size: 200000
    block-size: 2097152
  - interface: enp4s0f0
    threads: 16
    defrag: no
    cluster-type: cluster_flow
    cluster-id: 93
    copy-mode: ips
    copy-iface: enp4s0f1
    buffer-size: 64535
    use-mmap: yes
    ring-size: 200000
    block-size: 2097152
  - interface: enp4s0f1
    threads: 16
    defrag: no
    cluster-type: cluster_flow
    cluster-id: 92
    copy-mode: ips
    copy-iface: enp4s0f0
    buffer-size: 64535
    use-mmap: yes
    ring-size: 200000
    block-size: 2097152
```

In the above settings, Suricata is set to IPS mode, AF-PACKET uses mmap mode, set larger ring-size and buffer-size, and set worker threads to maintain a certain ratio with the network card queue. After testing, such settings are basically to achieve optimal performance.

4.6 Comparison of Performance Results

The figure below shows the network throughput of the system before optimization. It can be seen that when the network traffic is 1Gbps, the jitter is severe and the packet loss is serious.

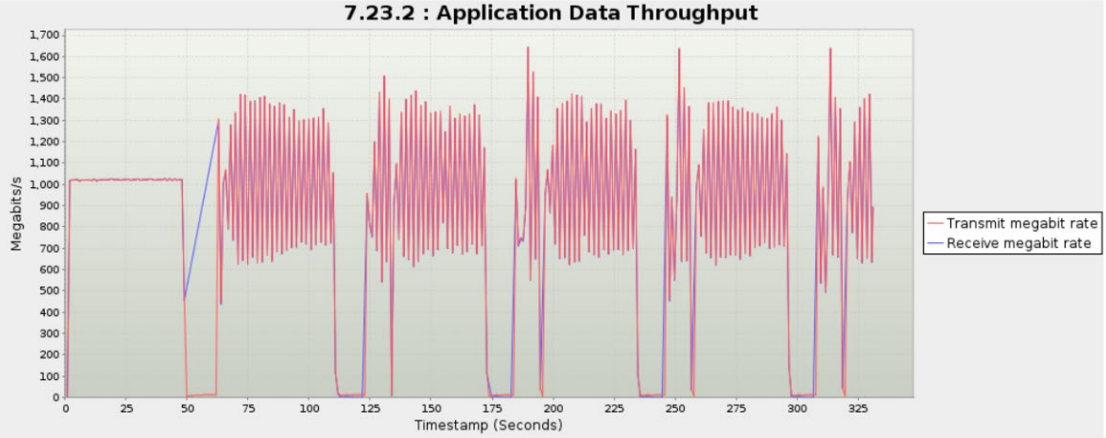

Figure 5. Network Throughput before Optimization

he figures below is the optimized network throughput of a single network card. It can be seen that it is close to 10G bps, and it reaches line speed without packet loss. The throughput of 2 network cards is already close to 20Gbps.

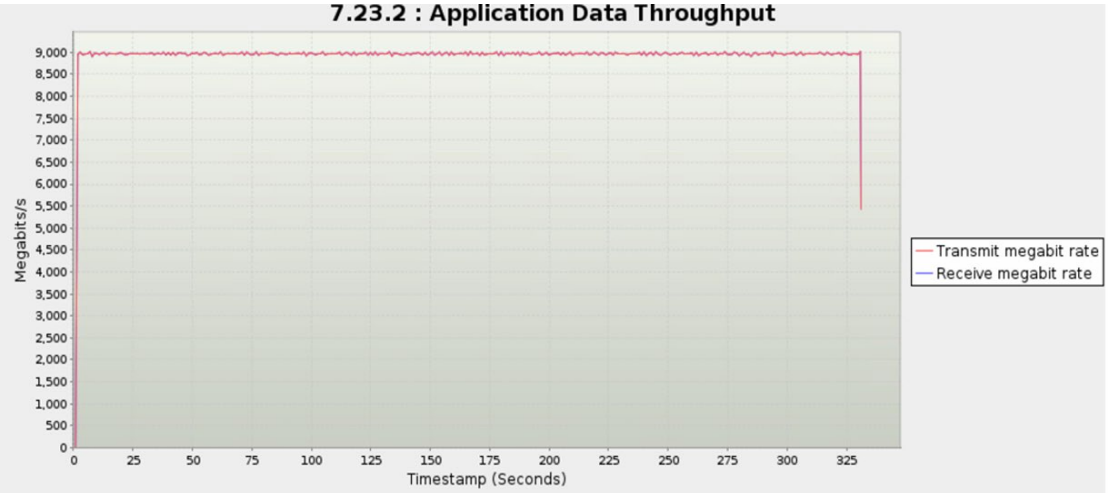

Figure 6. Network Throughput after Optimization

After the optimization process is completed, the maximum number of network connections and small packet throughput of the system have also been significantly improved comared to the original operating system.

5. Conclusion

In short, after research and development, our Suricata-based IPS system can perfectly adapt to Huawei Kunpeng 920 CPU and Galaxy Kylin operating system. The interception rate of our IPS reaches 98.5%, and the error rate is nearly zero. The network throughput of the system reaches 20Gbps of high network performance. Besides, the advantage is that our IPS system is based on AF-PACKET technology, with simple configuration, wide adaptability and high network throughput. However, there is still a certain gap with DPDK in terms of small packet throughput [13], and further research and development is needed in this area.

Our future research includes:

1) Develop a virtualized intrusion prevention system based on Suricata to facilitate the installation and operation of the system.

2) Optimize the program of kernel Malloc function, so that the system can support more traffic.

3) Continue optimization research for Suricata's DPDK and AF-PACKET technologies.

4) Consider how to improve the performance of the system when there are millions of defense rules.

The ultimate goal is to build an intrusion prevention system with precise defense, high performance, and easy configuration.

Acknowledgments

We thank all team members, as well as Ixia, Netitest, and Spirent for their equipment assistance.

References

[1] Suricata, Suricata, 2021, [online] Available at: https: //suricata.readthedocs.io/ [Accessed 2 2021].

[2] Snort.org, Snort - network intrusion detection & prevention system, 2021, [online] Available at: <https://www.snort.org/> [Accessed 2 2021].

[3] Zeek, the zeek network security monitor, 2021, [online] Available at: <https://zeek.org/> [Accessed 2 2021].

[4] W. Bulajoul, A. James, M. Pannu, Network intrusion detection systems in high-speed traffic in computer networks, in: 2013 IEEE 10th International Conference on E-Business Engineering, IEEE, 2013, pp. 168–175.

[5] G. Raj, M. Katoch, Security implementation through pcre signature over cloud network, Adv. Comput. 3 (3) (2012) 121.

[6] D. Day, B. Burns, A performance analysis of snort and suricata network intrusion detection and prevention engines, in: Fifth International Conference on Digital Society, Gosier, Guadeloupe, 2011, pp. 187–192.

[7] ntop. PF_RING ZC (Zero Copy). [Online]. Available: http://www.ntop.org/products/packet-capture/pf_ring/pf_ring-zc-zero-copy/.

[8] Harsh Patel, Hrishikesh Hiraskar, Mohit P. Tahiliani, extending network emu-lation support in ns-3 using dpdk, in: Workshop on ns-3 (WNS3), 2019, pp.17–24.

[9] Luigi Rizzo, Netmap: a novel framework for fast packet i/o, in: USENIX Conference on Annual Technical Conference (ATC), 2012, p. 9.

[10] Q. Hu, M.R. Asghar, N. Brownlee, Evaluating network intrusion detection systems for high-speed networks, in: 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), IEEE, 2017, pp. 1–6.

[11] A. Alhomoud, R. Munir, J.P. Disso, I. Awan, A. Al-Dhelaan, Performance evaluation study of intrusion detection systems, Procedia Comput. Sci. 5 (2011) 173–180.

[12] M. Dashti, A. Fedorova, J. Funston, F. Gaud, R. Lachaize, B. Lepers, V.Quema, M. Roth, Traffic management: A holistic approach to memoryplacement on NUMA systems, SIGPLAN Not. 48 (4) (2013) 381–394.