# Improvement of Hardware Firewall's Data Rates by Optimizing Suricata Performances

Kire Jakimoski, *Faculty of Informatics, FON University – Skopje, Republic of North Macedonia,*
Nidhi V Singhai, *Red Piranha Limited – Australia*

*Abstract* — **Suricata is a high-performance, multi-threaded Network IDS, IPS and Network Security Monitoring engine that can monitor networks in real time to protect against attacks. With its diverse features, Suricata is the choice of modern Unified Threat Management (UTM) systems to help networks secure their boundaries. As the network capacity increases to 40Gbps and beyond, it becomes important to tune Suricata to provide a lossless detection to the network. This paper describes the different tunings that were done to Red Piranha's Crystal Eye appliances to achieve 60Gbps Suricata throughput. Suricata throughput, as described in this paper, defines the amount of data that can be handled by Suricata engine without any drops. We describe the hardware configurations as well as the Suricata configurations that can help achieve high detection rates. We have also added the test results for single NIC and dual NIC systems and discussed the impact of hardware on Suricata performance.**

*Keywords* — **Data Rates, Intrusion Detection System, multithreading, Suricata, Traffic.**

## I. Introduction

IMPROVEMENT of the data processing rates of the modern hardware firewalls is more than needed nowadays. Network Intrusion Detections Systems (NIDs) are important part of the hardware firewalls for detection of malicious activities within the networks. NIDs are facing great challenges with rapid increasement of the network speeds when monitoring large and diverse traffic volumes. Suricata is a popular open-source, multi-threaded, high-performance NIDS that is researched in many scientific papers. In [1] authors compare Suricata with Snort and Bro as most popular open source NIDSs. Comparison in [1] is done in packet processing speed, system resource usage and packet drop rate. Snort and Suricata are evaluated in [2] using hardware network testing setup to ensure a realistic environment where focus is put on the accuracy of the detection especially dependent on bandwidth.

In [3] authors installed Suricata and Snort on two different but identical computers and it was noted that Suricata could process higher data rates of network traffic than Snort with lower packet drop rate with consuming higher computational resources.

Results in [4] showed that Suricata drops fewer packets compared to Bro and Snort successively when a DDoS attack is happening and detect more malicious packets.

Authors in [5] are studying the three most popular NIDS tools, Suricata, Snort and Bro. Processing and detection rate of Suricata and Snort are analyzed and compared in [6] in order to decide which is better in single threading or multi-threading environment.

Authors in [7] found out that the deep packet inspection based on many-core platform can process network traffic of which the bandwidth reaches up to 4 Gbps.

Red Piranha's Crystal Eye UTM appliances [8] are multi-core systems that enable multi-threaded applications to use the underlying hardware for high performance. Multi-threading scales the system by adding more threads for running different applications that inspect the incoming traffic before transmitting it to/from the protected network.

Crystal Eye uses Suricata as its Intrusion Detection and Protection Engine. The IDPS solution of Crystal Eye can be used in IDS, IPS or NSM mode. As the range of UTM products increase in their capacity to handle higher traffic speeds, it becomes important to tune Suricata to provide a lossless detection to the network.

Crystal Eye Series-80 is a high-end appliance that is suitable for telecoms or large IT needs. This paper provides details about Suricata tuning efforts for this appliance.

The remainder of this paper is organized as follows: Section 2 explains test setup details. Important considerations are mentioned in Section 3. Section 4 describes traffic tests and traffic profile. Test phase 1 is described in Section 5. Test results of the test phase 1 are presented in the Section 6. Section 7 is describing test phase 2 and Section 8 present test results of the test phase 2. Finally, Section 9 concludes the paper.

## II. Test Setup

The technical specifications of the Crystal Eye appliance used for the tests have taken [10] Septun Mark I as reference. Hardware setup is described below:

The work on which this paper reports is supported by the team of Red Piranha Limited, Australia. We thank Adam Bennett for permission to publish this material.

K. Jakimoski is with the Faculty of Informatics, FON University, Bul. Kiro Gligorov 5, 1000 Skopje, Republic of North Macedonia; (e-mail: kire.jakimoski@fon.edu.mk).

N.V. Singhai is with the Red Piranha Limited, Australia (e-mail: nidhi@redpiranha.net).

978-1-7281-4790-1/19/$31.00 ©2019 IEEE

- *Dual Intel® Xeon® CPU E5-2697 v4. This is an 18 core HT system. Total number of threads in the system with Dual CPU was 72.*
- *2 x Dual port XL710 40 GbE NICs. Only 1 port from each NIC was used.*
- *128 GB RAM, 8 DIMMS, 4 per socket*
- *OS: Ubuntu 18.04.2 LTS (Bionic Beaver), Kernel: 4.18.0-20-generic*
- *Latest Suricata (5.0.0 dev) from git*
- *Latest i40e driver from Intel (v2.9.21).*

## III. Tuning Considerations

Suricata's performance in our setup was dependent upon below factors:

- **HW capability and capacity**
  This is the most important factor in Suricata's performance. We used the cards that supported multi-queue feature to hash the incoming traffic to multiple receive queues. Hardware considerations have been discussed in detail below.

- **Suricata version**
  Latest Suricata version includes performance improvements and security fixes. Thus, we picked the latest from git.

- **Rules**
  Suricata scans the rules in-order to perform inspections. Hence the number of rules plays an important part in the performance. In our case, we have used 14350 signatures from Emerging Threats ruleset.

- **BIOS Settings**
  While most of the tests with default BIOS settings gave us expected results, we noticed that for speeds higher than 50Gbps, BIOS tunings were needed. We followed the suggestions in [10] Septun Mark I for our motherboard and made the settings as below
  - Disable ASPM
  - Disable VT-d
  - Disable SR-IOV (already disabled in our case)
  - Disable hardware prefetcher
  - Disable adjacent sector prefetcher
  - Disable DCU stream prefetcher

- **Traffic type**
  While traffic is not a tunable parameter, we have added it here since we saw higher throughput rates with simple http flows. We experimented with different traffic profiles in our tests and have mentioned the profile that simulated the enterprise traffic fully in this paper.

Most of the tunings in the tests were done to make sure that the packet is read from the L3 cache in order to minimize the time that is required to fetch the data.

Several important considerations in this regard are:

- Single port of each XL710 card was used. These cards come with dual ports where the second port has been reserved for redundancy.
- NICs on different NUMA (Non-Uniform Memory Access) nodes were used. The Crystal Eye appliance used in the tests had 2 NUMA nodes. Interfaces were identified on the different NUMA nodes and used for testing. In our setup, each NUMA node was attached to 36 cores.
- Interface receive and transmit queues were pinned to the cores that were local to the NUMA node for the interface. Symmetric hashing was used to evenly distribute the traffic among these queues.
- Suricata was run in worker mode and worker threads for each interface were pinned to the cores local to the NUMA node for the interface. This ensured that the packet was read from the local L3 cache and avoided NUMA interleaving while reading packets by Suricata.
- Memcap values in the Suricata config file were adjusted to handle high traffic load. These values were adjusted keeping in mind the available memory and the memory consumption of Suricata as described by Peter Manev in his blog [9].
- Interface buffers and interrupts were adjusted in order to minimize losses at the NIC and Suricata.

NIC offloading was disabled and Network balancing was enabled for the interfaces.

## IV. Traffic Tests and Traffic Profile

Traffic testing for the setup was performed in two phases. In the first phase, single NIC card was used for testing. A maximum of 34.5 Gbps Suricata throughput was obtained in this case. In the second phase, we tested with two NIC cards located on different NUMA nodes. With proper tuning, we were able to achieve a maximum of 60 Gbps Suricata throughput in this case.

Trex traffic generator was used to generate stateful traffic for our setup. Trex uses traffic profiles to replay traffic in the system. Traffic profiles use pcaps to generate the connections at the defined rates in the profiles. In our case, client was running on the Trex and the Crystal Eye appliance acted as the receiver of the traffic. Such a profile is mostly used in IDS mode

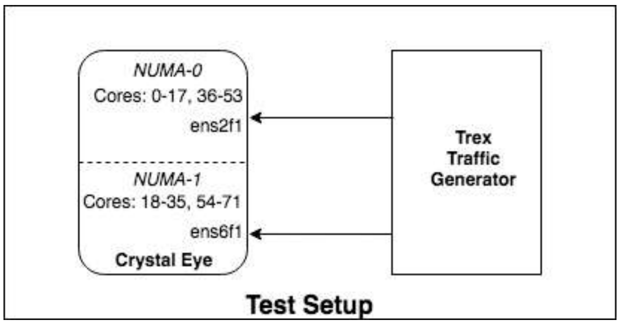

Fig. 1. Test Setup

Our traffic profile can be roughly described as:

```
HTTP/HTTPS traffic: 77.49%
RTP: 13.23%
Exchange: 4.31%
Citrix: 2.15%
SMTP: 1.08%
DNS: 1.08%
Oracle: 0.65%
-------------------------------------------
Total: 100%
```

## V. TEST PHASE 1 (SINGLE NIC)

In this test, we passed traffic from Trex to only one interface in Crystal Eye appliance. Interrupts for this interface were pinned to 34 cores belonging to the NUMA node that was local to the interface, leaving the 2 cores for housekeepin. We used a low entropy key to enable Receive Side Scaling (RSS) hashing to these cores.

We used the linux ethtool utility to set the interface parameters.

The number of receive queues on the interface were set to 34

```
  ethtool -K ens2f1 rxhash on
  ethtool -K ens2f1 ntuple on
  ethtool -L ens2f1 combined 34
```

We used the i40e driver's script, set_irq_affinity, to assign the interrupts to the cores local to the NUMA node for the interface

```
set_irq_affinity 1-17,37-53 ens2f1
```

We enabled symmetric RSS with a low entropy key for our interface

```
ethtool -X ens2f1 hkey
6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6
D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D
:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:
5A:
6D:5A equal 34
```

Suricata was started in AF-PACKET mode and Suricata configuration was modified to enable cpu-affinity for the threads. The worker threads were pinned to the same cores as the receive queues for the interface. This enabled Suricata to read the packet from the L3 cache as much as possible and improve performance. Also, cluster-type for the interfaces was set to cluster_qm to bind Suricata to the RSS queues.

```
af-packet:
  - interface: ens2f1
    threads: 17
    cluster-id: 99
    cluster-type: cluster_qm
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
     ring-size: 300000
    block-size: 2097152

  - interface: ens2f1
    threads: 17
    cluster-id: 99
    cluster-type: cluster_qm
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
  ring-size: 300000
    block-size: 2097152

threading:
  set-cpu-affinity: yes
  cpu-affinity:
    - management-cpu-set:
        cpu: [ "0", "36" ]
    - worker-cpu-set:
        cpu: [ "1-17", "37-53" ]
        mode: "exclusive"
```

## VI. TEST RESULTS OF TEST PHASE 1

With the above settings, we were able to process input rates of 34 Gbps on single port of 40GbE card without any drops at the NIC or Suricata. CPU utilization on the Crystal Eye appliance was of the order of 80-95% across all the 34 cores. Suricata statistics logs showed that all the packets captured by Suricata were picked up by the engine for analysis and there were no kernel drops observed in the system.

## VII. TEST PHASE 2 (DUAL NICS)

In the second phase of the testing, 60 Gbps traffic was passed from Trex to two interfaces on separate NUMA nodes on Crystal Eye. The interface configs, entropy keys and receive queues were kept same as in Test Phase 1.

```
ethtool -L ens2f1 combined 34
ethtool -L ens6f1 combined 34

set_irq_affinity 1-17,37-53 ens2f1
set_irq_affinity 19-35,55-71 ens6f1
```

In order to maintain NUMA locality of the received packets and minimize L3 cache misses, Suricata configuration enabled cpu-affinity for the threads. The worker threads were pinned to the cores matching the receive queues for the interfaces. Also, cluster_type for the interfaces was set to cluster_qm to bind Suricata to the RSS.

```
af-packet:
  - interface: ens2f1
    threads: 17
    cluster-id: 99
    cluster-type: cluster_qm
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
    ring-size: 300000
    block-size: 2097152

  - interface: ens6f1
    threads: 17
    cluster-id: 98
    cluster-type: cluster_qm
    defrag: yes
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
    ring-size: 300000
    block-size: 2097152

  - interface: ens2f1
    threads: 17
    cluster-id: 99
    cluster-type: cluster_qm
    defrag: yes
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
    ring-size: 300000
    block-size: 2097152

  - interface: ens6f1
    threads: 17
    cluster-id: 98
    cluster-type: cluster_qm
    defrag: yes
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
    ring-size: 300000
    block-size: 2097152

  cpu-affinity:
    - management-cpu-set:
        cpu: [ 0,18,36,54
    - worker-cpu-set:
        cpu: [ "1-17", "19-35", "37-53", "55-71"]
        mode: "exclusive"
        prio:
          low: [ ]
          medium: [ 0,18,36,54]
          high: [ "1-17","19-35","37-53","55-71"]
          default: "high"
```

## VIII. Test Results of Test Phase 2

With the above settings, we were able to process input rates of 60 Gbps on two 40GbE NICs in Crystal Eye without any drops at the NIC or Suricata. CPU utilization on the Crystal Eye appliance was of the order of 80-95% across all the 68 cores processing Suricata worker threads. Similar to test phase 1, Suricata statistics from the system showed that all the packets captured by Suricata were picked up by the engine for analysis and there were no kernel drops observed in the system.

## IX. Conclusion

The system and Suricata tunings described in this paper tested Suricata's capacity to process traffic at 60Gbps for a high-speed network. Furthermore, the forwarding capacity of the complete system with two receive and two transmit interfaces was bound to data speed of 30 Gbps.

Considering the above test results, it shows that future improvements in Suricata throughput can be achieved by

- Testing on a system with more sockets. This would increase the number of NUMA nodes and more NIC cards can be added to increase the overall capacity.
- Increasing the number of cores in the system. This would increase the number of Suricata worker threads and hence the capacity.

Also, recent Suricata releases have an extensive support for XDP. We did not use this feature in our test setup since our traffic replay consisted of small flows. In case of elephant flows, we believe that XDP will give us good results as well. We look forward to testing it on our setups.

## Acknowledgment

The work on which this paper reports is supported by the management of Red Piranha Limited, Australia. We thank Adam Bennett for permission to publish this material.

.

## References

[1] Hu, Qinwen, Muhammad Rizwan Asghar, and Nevil Brownlee. "Evaluating network intrusion detection systems for high-speed networks." *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 2017.

[2] Lukaseder T, Fiedler J, Kargl F. Performance Evaluation in High-Speed Networks by the Example of Intrusion Detection. arXiv preprint arXiv:1805.11407. 2018 May 29.

[3] Shah SA, Issac B. Performance comparison of intrusion detection systems and application of machine learning to Snort system. Future Generation Computer Systems. 2018 Mar 1;80:157-70.

[4] Cherkaoui R, Zbakh M, Braeken A, Touhafi A. Performance Analysis of Intrusion Detection Systems in Cloud-Based Systems. InInternational Symposium on Ubiquitous Networking 2017 May 9 (pp. 206-213). Springer, Cham.

[5] Cherkaoui R, Zbakh M, Braeken A, Touhafi A. Performance Analysis of Intrusion Detection Systems in Cloud-Based Systems. InInternational Symposium on Ubiquitous Networking 2017 May 9 (pp. 206-213). Springer, Cham.

[6] Park W, Ahn S. Performance comparison and detection analysis in snort and suricata environment. Wireless Personal Communications. 2017 May 1;94(2):241-52.

[7] Zhan YR, Wang ZS. Deep packet inspection based on many-core platform. journal of computer and communications. 2015 May 25;3(05):1.

[8] https://redpiranha.net/crystal-eye-utm-appliances.

[9] http://pevma.blogspot.com/2015/10/suricata-with-afpacket-memory-of-it-all.html.

[10] https://github.com/pevma/SEPTun.

[11] https://github.com/pevma/SEPTun-Mark-II.