Red Piranha
unified threat management

Introduction

Red Piranha’s Crystal Eye UTM appliances 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 imperative 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 document provides details about Suricata tuning efforts for this appliance. This document has been inspired by Septun Mark I and Septun Mark II from the OISF team.

Our special thanks to Peter Manev(@pevma) who was as keen with our testing as we were and guided us throughout the course of this exercise. Special thanks also to Dylan Leahy, Victor Julien and the team @ OISF & Red Piranha.

Test Setup

Our test setup closely matches the one used in Septun Mark I/II to replicate the test conditions and the new hardware configuration used in these tests is detailed below.

- 2 x dual port XL710 40GbE cards – 1 port of each card used
- 2 x Intel(R) Xeon(R) CPU E5-2697 v4 – 36 cores total, 72 threads with HT enabled
- 128GB 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)
- 14350 signatures from Emerging Threats ruleset

Important Considerations

As per the Septun document, Suricata's performance is dependent upon 4 major variables

- Suricata version
- Traffic type
- Rules used
- HW capability

This document details these variables along with the tuned configuration settings that we used in our setup.

Most of the tunings in Septun were done to make sure that the packet is read from the L3 cache to minimize the time required to fetch the data. Some of the important considerations in this regard have been listed here

- 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 nodes were used. This is a dual socket system, with 2 NUMA nodes. Interfaces were identified on the different NUMA nodes and used for testing.

  `lstopo` is the linux utility which was used to fetch the information about the interface association with the NUMA nodes.

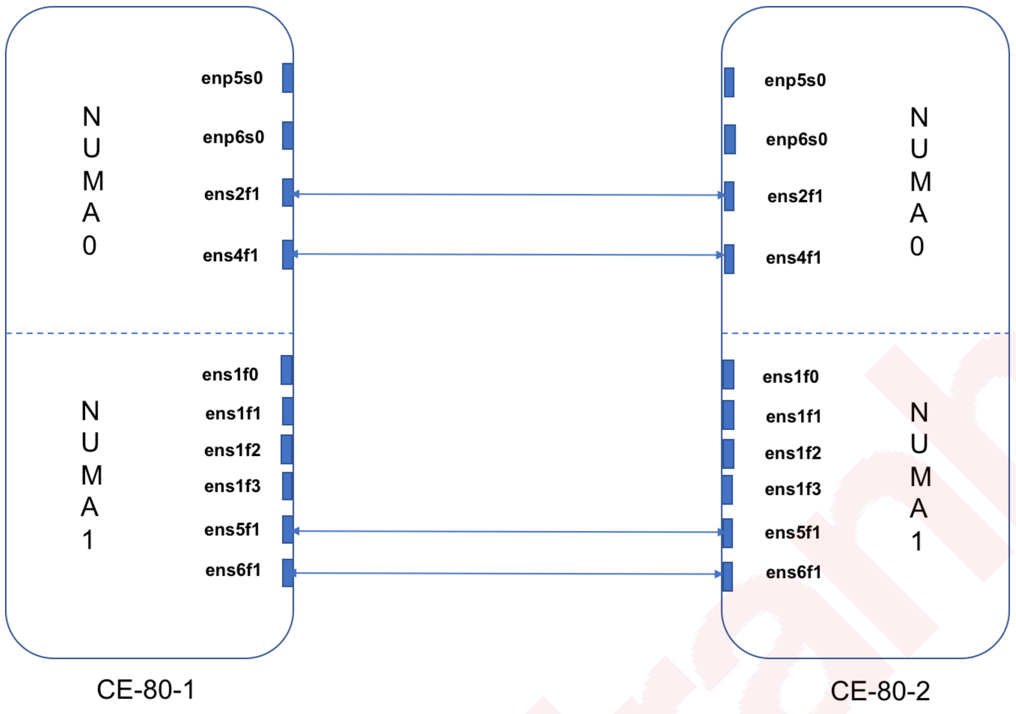
  From our setup:

  ```
  # lstopo
  Machine (126GB total)
    NUMANode L#0 (P#0 63GB)
  …
  HostBridge L#0
          PCIBridge
            PCI 8086:1583
              Net L#0 "ens2f0"
            PCI 8086:1583
  ```

```
          Net L#1 "ens2f1"
      PCIBridge
        PCI 8086:1583
          Net L#2 "ens4f0"
        PCI 8086:1583
          Net L#3 "ens4f1"
      …
      PCIBridge
        PCI 8086:1533
          Net L#6 "enp5s0"
      PCIBridge
        PCI 8086:1533
          Net L#7 "enp6s0"

NUMANode L#1 (P#1 63GB)
    HostBridge L#7
      PCIBridge
        PCI 8086:1589
          Net L#11 "ens1f0"
        PCI 8086:1589
          Net L#12 "ens1f1"
        PCI 8086:1589
          Net L#13 "ens1f2"
        PCI 8086:1589
          Net L#14 "ens1f3"
      PCIBridge
        PCI 8086:1583
          Net L#15 "ens5f0"
        PCI 8086:1583
          Net L#16 "ens5f1"
      PCIBridge
        PCI 8086:1583
          Net L#17 "ens6f0"
        PCI 8086:1583
          Net L#18 "ens6f1"
```

Thus, interfaces on NUMA0 are ens2f0, ens2f1, ens4f0, ens4f1, enp5s0, enp6s0. Similarly, interfaces on NUMA1 are ens1f0, ens1f1, ens1f2, ens1f3, ens5f0, ens5f1, ens6f0, ens6f1.

We used interfaces ens2f1 and ens6f1 for our testing.



- Isolate all cores for Suricata from any tasks and only use dedicated threads for housekeeping task. This is done to prevent any user tasks from being scheduled on these threads.

  In our setup, with 2 NUMA nodes, the CPU cores were distributed as below

  ```
  # lscpu
  …
  NUMA node0 CPU(s):   0-17,36-53
  NUMA node1 CPU(s):   18-35,54-71
  ```

  We identified 0,18,36 and 54 for housekeeping and isolated the remaining cores for Suricata. This was done by modifying the grub configuration

  ```
  # cat /proc/cmdline
  ```

  ```
  BOOT_IMAGE=/boot/vmlinuz-4.18.0-20-gener                ic
  root=UUID=61a72786-ba4f-11e8-bf06-4cedfb911a90  ro  maybe-
  ubiquity          audit=0          processor.max_cstate=3
  intel_idle.max_cstate=3 selinux=0 apparmor=0 mce=ignore_ce
  isolcpus=1-17,19-35,37-53,55-71
  ```

- Symmetric hashing of the receive traffic (RSS) was used to evenly distribute the traffic between all the queues. Suricata worker threads were pinned to the same cores.

  We used the i40e driver script, `set_irq_affinity`, to define the receive and transmit queues for the traffic. The configuration details are mentioned below.

- Suricata was run in workers mode and the worker threads were pinned to the cores depending upon the NUMA locality as described in Suricata configs for different tests.

- Memcap values in the Suricata config file were adjusted to handled high traffic load. We adjusted these values keeping in mind the available memory and the memory consumption of Suricata as described by Peter Manev in his blog [5].

- Buffers and interrupts were adjusted to minimize losses at the NIC and Suricata

Preparations

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 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

Given our motherboard constraints with this test set up, we were not able to specifically disable Node Interleaving and Enable IOAT. But statistics from our box suggested that node-interleaving was not being used in our tests. We verified this by observing the `interleave_hit` value in `numastat` output. Between multiple runs in the setup, this value remained constant.

Also, DDIO was enabled by default in our system.

Packages

Common libraries installed for our setup

```
apt-get -y install git build-essential autoconf automake \
libtool pkg-config  libpcre3 libpcre3-dbg libpcre3-dev \
libpcap-dev libnet1-dev  libyaml-0-2 libyaml-dev zlib1g \
zlib1g-dev  libmagic-dev libcap-ng-dev libjansson-dev \
libjansson4 libnss3-dev libnspr4-dev libgeoip-dev libluajit-5.1-dev \
rustc cargo clang libelf-dev clang-6.0 libncurses5-dev gcc make git bc
libssl-dev build-essential autoconf automake clang-tools-6.0 \
libllvm6.0  llvm-6.0-tools  bison  flex  tmux  lshw  python3-distutils  \
python3-yaml libhyperscan-dev libmaxminddb-dev
```

- Hyperscan

Latest hyperscan (v5.1.1, as of writing this document) was installed from github as per the instructions
https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Hyperscan

- BPF

```
cd /opt && \
git clone https://github.com/libbpf/libbpf && cd libbpf/src/ && \
make clean && make && \
make install && make install_headers && \
echo "/usr/lib64" > /etc/ld.so.conf.d/local.conf && \
ldconfig
```

- Latest Intel Drivers

```
mkdir -p /opt/i40e && \
cd /opt/i40e/ && \
wget
https://sourceforge.net/projects/e1000/files/i40e%20stable/2.7.29/i40e-
2.7.29.tar.gz && \
tar -zxf i40e-2.7.29.tar.gz && \
cd /opt/i40e/i40e-2.7.29/src && \
make clean && make && make install && \
cd ../../ && \
ls -lh i40e-2.7.29/scripts/set_irq_affinity

rmmod i40e && modprobe i40e
```

- Latest Ethtool

```
cd /opt/ && \
wget
https://mirrors.edge.kernel.org/pub/software/network/ethtool/ethtool-
4.19.tar.xz && \
tar -xf ethtool-4.19.tar.xz && \
cd ethtool-4.19 && \
./configure && make clean && make && make install && \
ls -lh /usr/local/sbin/ethtool
```

- Suricata

Latest Suricata from git (5.0.0-dev) was installed on the box for testing.

```
cd /opt && \
git clone https://github.com/OISF/suricata.git && cd suricata \
&& git clone https://github.com/OISF/libhtp.git -b 0.5.x && \
./autogen.sh    &&    CC=clang-6.0    ./configure    --prefix=/usr/    --
sysconfdir=/etc/ --localstatedir=/var/  \
--enable-geoip --enable-rust-strict \
--enable-luajit --enable-ebpf --enable-ebpf-build \
--enable-hyperscan && \
sudo make clean && sudo make -j && \
sudo make install-full && sudo ldconfig

cp ebpf/xdp_filter.bpf /etc/suricata/
```

- Trex

```
# mkdir -p /opt/trex
# cd /opt/trex
# wget --no-cache http://trex-tgn.cisco.com/trex/release/latest
# tar -xzvf latest
```

Enable IP Forwarding

Make sure IP forwarding is enabled on the system

```
#sysctl net.ipv4.ip_forward

net.ipv4.ip_forward = 1
```

Disable IRQ Balance

```
systemctl stop irqbalance
systemctl disable irqbalance
```

Disable NIC offloading

Offloading was disabled for all the interfaces

```
ip link set ens2f1 promisc on arp off up
echo 1 > /proc/sys/net/ipv6/conf/ens2f1/disable_ipv6

for i in rx tx tso ufo gso gro lro tx nocache copy sg txvlan rxvlan; do
      /sbin/ethtool -K <intf> $i off 2>&1 > /dev/null;
done
```

Enable Network Balancing

This was done for each test for all the interface participating in the traffic.

```
for proto in tcp4 udp4 tcp6 udp6; do
       /usr/local/sbin/ethtool -N <intf> rx-flow-hash $proto sdfn
done
```

Adjust NIC interrupts

Interrupt rates need to be properly adjusted so that they do not increase CPU utilization or prevent the packets being dropped due to slow CPU. For our setup, the below configs worked well

```
/usr/local/sbin/ethtool -C ens2f1 adaptive-rx off adaptive-tx off rx-usecs
150
/usr/local/sbin/ethtool -C ens6f1 adaptive-rx off adaptive-tx off rx-usecs
150
```

Adjust NIC ring descriptor size

Ring descriptor sizes have impact on the L3 cache miss ratio. Higher values can increase the L3 cache miss, which would result in slower packet access. Lower values can lead to packet drops, if CPU is not reading fast enough. For our setup and the interrupt values mentioned above, below values worked for us.

```
/usr/local/sbin/ethtool -G ens2f1 rx 1024
```

```
/usr/local/sbin/ethtool -G ens6f1 rx 1024
```

In one of our runs, L3 cache miss was in the range of 3% for 60Gbps traffic.

On NUMA0:

```
perf stat -e LLC-loads,LLC-load-misses,LLC-stores,LLC-prefetches -C 2 sleep 60

 Performance counter stats for 'CPU(s) 2':

     1,343,332,368      LLC-loads                                                   (66.67%)
        39,568,833      LLC-load-misses    #    2.95% of all LL-cache hits (66.67%)
       167,477,300      LLC-stores                                                  (66.67%)
   <not supported>      LLC-prefetches

      60.001256221 seconds time elapsed
```

On NUMA1:

```
perf stat -e LLC-loads,LLC-load-misses,LLC-stores,LLC-prefetches -C 20 sleep 60

 Performance counter stats for 'CPU(s) 20':

     1,319,813,342      LLC-loads                                                   (66.67%)
        38,714,606      LLC-load-misses     #   2.93% of all LL-cache hits (66.67%)
       170,313,548      LLC-stores                                                  (66.66%)
   <not supported>      LLC-prefetches

      60.001568549 seconds time elapsed
```

Disable pause frames

As per the recommendations in Septun Mark I, we disabled the pause frames on our interfaces

```
/usr/local/sbin/ethtool -A ens2f1 rx off tx off
/usr/local/sbin/ethtool -A ens6f1 rx off tx off
```

Suricata Config

Common changes for all the tests made to the default suricata.yaml have been listed below. Af-packet and cpu-affinity configs have been mentioned in the tests section.

- Disable line based logs
  We should only enable the logs that are needed for our setup to minimize load on the CPU for logging.

  ```
  outputs:
    # a line based alerts log similar to Snort's fast.log
    - fast:
        enabled: no
  ```

- Set eve log filetype to 'syslog'

  ```
  - eve-log:
        enabled: yes
        filetype: syslog
  ```

- Disable all event types except alerts

We were interested only in alerts for these tests, so we disabled all the other events.

- Application Layer configs

```
app-layer:
  protocols:
    tls:
      ja3-fingerprints: yes
     encrypt-handling: bypass

   dns:
      # memcaps. Globally and per flow/state.
      global-memcap: 4gb
      state-memcap: 1mb

   libhtp:
         default-config:
           personality: IDS

           # Can be specified in kb, mb, gb.  Just a number indicates
           # it's in bytes.
           request-body-limit: 1mb #100kb
           response-body-limit: 1mb #100kb
```

- Increase max-pending-packets

  To deal with high traffic, we set the max-pending-packets to maximum value.

```
max-pending-packets: 65500
```

- Change runmode to workers

```
runmode: workers
```

- Set default-packet-size

```
default-packet-size: 1574
```

- Disable unix socket

  We did not need it for this testing. So we disabled it.

```
unix-command:
  enabled: no
```

- Adjust memcaps and timeout values (Note: Only modified parameters are mentioned below)
  Memcap values were adjusted taking into consideration the system memory size.

```
defrag:
  memcap: 1gb
  timeout: 10

flow:
  memcap: 18gb
```

```
    hash-size: 256072 #16388608
    prealloc: 300000
    managers: 1 # default to one flow manager
    recyclers: 1 # default to one flow recycler thread

  flow-timeouts:

    default:
      established: 60

    tcp:
      new: 30
      emergency-new: 1

    udp:
      established: 60

    icmp:
      established: 60

  stream:
    memcap: 12gb
    checksum-validation: no      # reject wrong csums
    prealloc-sessions: 200000
    bypass: yes
    reassembly:
      memcap: 24gb
      segment-prealloc: 200000
```

- Change performance settings

```
  detect:
     profile: high
```

- Enable auto mode for prefilter to use hyperscan for pattern matching.

```
  prefilter:
      default: auto
```

Traffic Tests

Traffic testing for the setup was performed in two phases.

In the first phase, we tested with a single NIC card. We were able to achieve a maximum of 34.5 Gbps Suricata throughput (decode packets = kernel packets) 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 60Gbps Suricata throughput in this case.

For both the tests, `rp_filter` was enabled in our setup.

```
root@ce-80-1:~# sysctl net.ipv4.conf.default.rp_filter
net.ipv4.conf.default.rp_filter = 1
root@ce-80-1:~# sysctl net.ipv4.conf.all.rp_filter
net.ipv4.conf.all.rp_filter = 1
```

Traffic profile:

One of the most important variables of Suricata performance is the type of traffic. While our observation was that http traffic gives wire speeds for Suricata, this does not simulate an enterprise traffic correctly. Thus, we chose our profile carefully to simulate the enterprise network as much as possible.

We used Trex traffic generator to generate stateful traffic for our setup. Trex was running on a system with similar configuration.
Latest Trex version from git (v2.56, as of writing this document) was used for these tests.

Trex uses profiles to replay pcap traffic in the system. Profiles contain information regarding the client and server addresses to be used and connections per second (cps) to be initiated for each pcap. Each pcap file should correspond to a single session. 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%
```

Test Phase 1: (Single NIC)

In this test, traffic was received on only one interface in the DUT.
Since this was the only active interface in the NUMA, we assigned all the rx queues to this interface to enable hashing of the receive traffic.

```
ifconfig ens2f1 down
/usr/local/sbin/ethtool -L ens2f1 combined 34
/usr/local/sbin/ethtool -K ens2f1 rxhash on
/usr/local/sbin/ethtool -K ens2f1 ntuple on
ifconfig ens2f1 up
/opt/i40e/i40e-2.9.21/scripts/set_irq_affinity 1-17,37-53 ens2f1

/usr/local/sbin/ethtool -X ens2f1 hkey
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:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:
6D:5A equal 34
ethtool -x ens2f1
ethtool -n ens2f1
```

Since we enabled receive hashing, we added cluster_qm for the cluster-type in our af-packet interface configs. Also, we needed two separate configurations for the interface to map the receive queues with the CPU threads mentioned in the affinity section.

Note: same interfaces will belong to the same cluster ie. cluster-id will be the same.

```
af-packet:
  - interface: ens2f1
    threads: 17
    cluster-id: 99
    cluster-type: cluster_qm
    defrag: yes
    # xdp-cpu-redirect: [ "2-18" ]
    xdp-mode: driver
    xdp-filter-file:  /etc/suricata/xdp_filter.bpf
    bypass: yes
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
    ring-size: 300000
    block-size: 2097152 #1048576

  - interface: ens2f1
    threads: 17
    cluster-id: 99
    cluster-type: cluster_qm
    defrag: yes
    # xdp-cpu-redirect: [ "2-18" ]
    xdp-mode: driver
    xdp-filter-file:  /etc/suricata/xdp_filter.bpf
    bypass: yes
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
    ring-size: 300000
    block-size: 2097152 #1048576
```

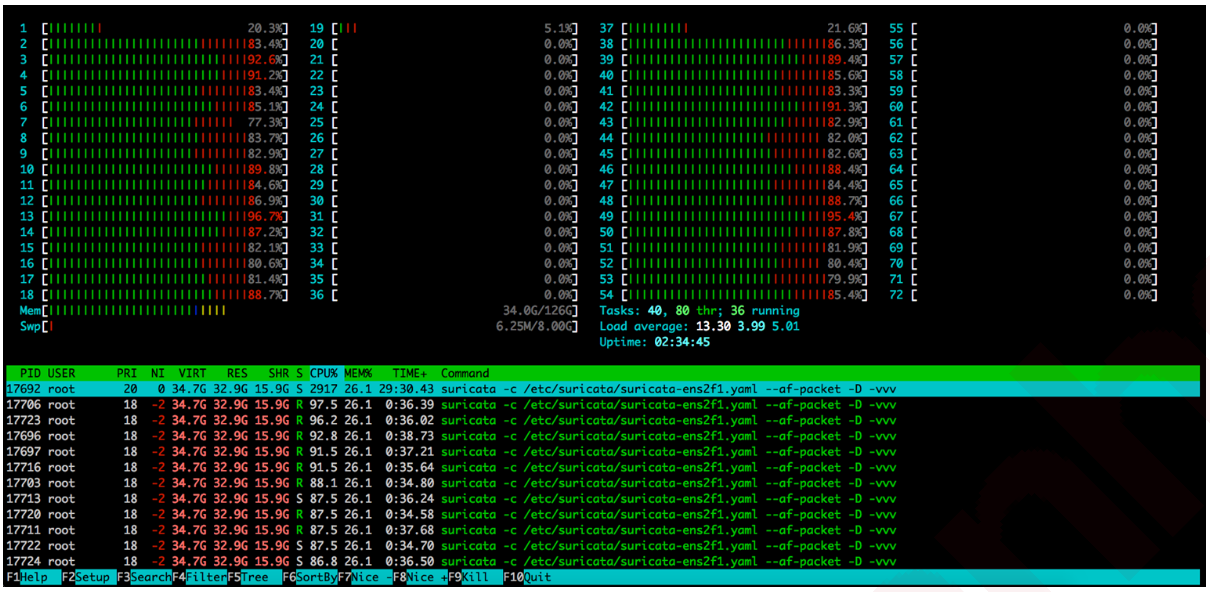
Setting CPU affinity in Suricata

```
threading:
  set-cpu-affinity: yes

  cpu-affinity:
    - management-cpu-set:
        cpu: [ "0", "36" ]  # include only these CPUs in affinity settings
    - worker-cpu-set:
        cpu: [ "1-17", "37-53" ]
        mode: "exclusive"
        # Use explicitely 3 threads and don't compute number by using
        # detect-thread-ratio variable:
        # threads: 3
        prio:
          low: [ ]
          medium: [ "0","36"]
          high: [ "1-17","37-53" ]
          default: "high"
```

Test results

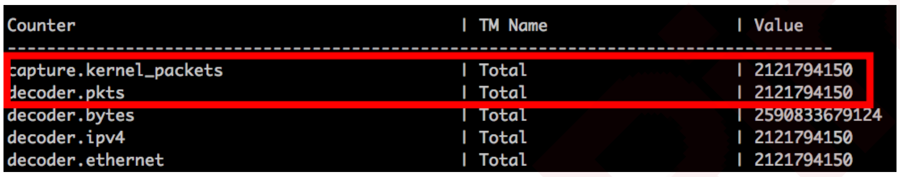
With the above settings, we were able to process input rates of 34Gbps on single port of 40GbE card without any drops at the NIC or Suricata.

<u>CPU utilization on DUT</u>



```
  1  [||||||||                  20.3%]   19 [|||                         5.1%]   37 [|||||||||                 21.6%]   55 [                            0.0%]
  2  [|||||||||||||||||||||||||||83.4%]   20 [                            0.0%]   38 [|||||||||||||||||||||||||||86.3%]   56 [                            0.0%]
  3  [|||||||||||||||||||||||||||92.6%]   21 [                            0.0%]   39 [|||||||||||||||||||||||||||89.4%]   57 [                            0.0%]
  4  [|||||||||||||||||||||||||||91.2%]   22 [                            0.0%]   40 [|||||||||||||||||||||||||||85.6%]   58 [                            0.0%]
  5  [|||||||||||||||||||||||||||83.4%]   23 [                            0.0%]   41 [|||||||||||||||||||||||||||83.3%]   59 [                            0.0%]
  6  [|||||||||||||||||||||||||||85.1%]   24 [                            0.0%]   42 [|||||||||||||||||||||||||||91.3%]   60 [                            0.0%]
  7  [|||||||||||||||||||||||||   77.3%]   25 [                            0.0%]   43 [|||||||||||||||||||||||||||82.9%]   61 [                            0.0%]
  8  [|||||||||||||||||||||||||||83.7%]   26 [                            0.0%]   44 [||||||||||||||||||||||||||  82.0%]   62 [                            0.0%]
  9  [|||||||||||||||||||||||||||82.9%]   27 [                            0.0%]   45 [|||||||||||||||||||||||||||82.6%]   63 [                            0.0%]
  10 [|||||||||||||||||||||||||||89.8%]   28 [                            0.0%]   46 [|||||||||||||||||||||||||||88.4%]   64 [                            0.0%]
  11 [|||||||||||||||||||||||||||84.6%]   29 [                            0.0%]   47 [|||||||||||||||||||||||||||84.4%]   65 [                            0.0%]
  12 [|||||||||||||||||||||||||||86.9%]   30 [                            0.0%]   48 [|||||||||||||||||||||||||||88.7%]   66 [                            0.0%]
  13 [|||||||||||||||||||||||||||96.7%]   31 [                            0.0%]   49 [|||||||||||||||||||||||||||95.4%]   67 [                            0.0%]
  14 [|||||||||||||||||||||||||||87.2%]   32 [                            0.0%]   50 [|||||||||||||||||||||||||||87.8%]   68 [                            0.0%]
  15 [|||||||||||||||||||||||||||82.1%]   33 [                            0.0%]   51 [|||||||||||||||||||||||||||81.9%]   69 [                            0.0%]
  16 [|||||||||||||||||||||||||||80.6%]   34 [                            0.0%]   52 [||||||||||||||||||||||||||  80.4%]   70 [                            0.0%]
  17 [|||||||||||||||||||||||||||81.4%]   35 [                            0.0%]   53 [|||||||||||||||||||||||||||79.9%]   71 [                            0.0%]
  18 [|||||||||||||||||||||||||||88.7%]   36 [                            0.0%]   54 [|||||||||||||||||||||||||||85.4%]   72 [                            0.0%]
  Mem[|||||||||||||||||||||||||||                                34.0G/126G]   Tasks: 40, 80 thr; 36 running
  Swp[|                                                          6.25M/8.00G]   Load average: 13.30 3.99 5.01
                                                                                Uptime: 02:34:45

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
17692 root       20   0 34.7G 32.9G 15.9G S 2917 26.1 29:30.43 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
17706 root       18  -2 34.7G 32.9G 15.9G R 97.5 26.1  0:36.39 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
17723 root       18  -2 34.7G 32.9G 15.9G R 96.2 26.1  0:36.02 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
17696 root       18  -2 34.7G 32.9G 15.9G R 92.8 26.1  0:38.73 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
17697 root       18  -2 34.7G 32.9G 15.9G R 91.5 26.1  0:37.21 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
17716 root       18  -2 34.7G 32.9G 15.9G R 91.5 26.1  0:35.64 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
17703 root       18  -2 34.7G 32.9G 15.9G R 88.1 26.1  0:34.80 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
17713 root       18  -2 34.7G 32.9G 15.9G S 87.5 26.1  0:36.24 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
17720 root       18  -2 34.7G 32.9G 15.9G R 87.5 26.1  0:34.58 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
17711 root       18  -2 34.7G 32.9G 15.9G R 87.5 26.1  0:37.68 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
17722 root       18  -2 34.7G 32.9G 15.9G S 87.5 26.1  0:34.70 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
17724 root       18  -2 34.7G 32.9G 15.9G S 86.8 26.1  0:36.50 suricata -c /etc/suricata/suricata-ens2f1.yaml --af-packet -D -vvv
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Suricata Stats



```
Counter                                  | TM Name                  | Value
-------------------------------------------------------------------------------------
capture.kernel_packets                   | Total                    | 2121794150
decoder.pkts                             | Total                    | 2121794150
decoder.bytes                            | Total                    | 2590833679124
decoder.ipv4                             | Total                    | 2121794150
decoder.ethernet                         | Total                    | 2121794150
```

Test Phase 2: (dual NICs)

In this test, interfaces on separate NUMA nodes were picked up to test 60Gbps traffic. Both the interfaces used 34 of the 36 cores for their packet reception and processing. The remaining 4 cores across the two nodes were used for housekeeping.

```
#Interface 1
ifconfig ens2f1 down
/usr/local/sbin/ethtool -L ens2f1 combined 34
/usr/local/sbin/ethtool -K ens2f1 rxhash on
/usr/local/sbin/ethtool -K ens2f1 ntuple on
ifconfig ens2f1 up
/opt/i40e/i40e-2.9.21/scripts/set_irq_affinity 1-17,37-53 ens2f1
/usr/local/sbin/ethtool -X ens2f1 hkey
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:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:
6D:5A equal 34


#Interface 2
ifconfig ens6f1 down
/usr/local/sbin/ethtool -L ens6f1 combined 34
/usr/local/sbin/ethtool -K ens6f1 rxhash on
/usr/local/sbin/ethtool -K ens6f1 ntuple on
ifconfig ens6f1 up
/opt/i40e/i40e-2.9.21/scripts/set_irq_affinity 19-35,55-71 ens6f1
/usr/local/sbin/ethtool -X ens6f1 hkey
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:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:
6D:5A equal 34
```

Since we are using two different interfaces here, they both will belong to different clusters. In our case, we have assigned `cluster-id 99` to ens2f1 and `cluster-id 98` to ens6f1.

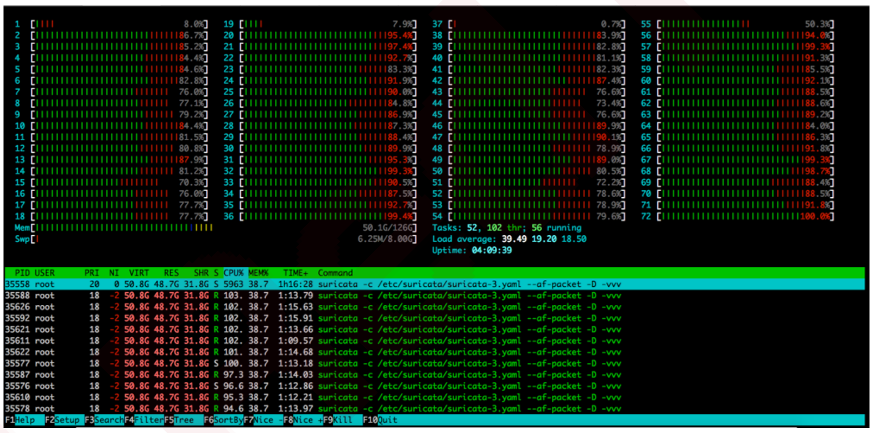
```
af-packet:
  - interface: ens2f1
    threads: 17
    cluster-id: 99
    cluster-type: cluster_qm
    defrag: yes
    # xdp-cpu-redirect: [ "2-18" ]
    xdp-mode: driver
    xdp-filter-file:  /etc/suricata/xdp_filter.bpf
    bypass: yes
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
    ring-size: 300000
    block-size: 2097152 #1048576

  - interface: ens6f1
    threads: 17
    cluster-id: 98
    cluster-type: cluster_qm
    defrag: yes
    # xdp-cpu-redirect: [ "2-18" ]
    xdp-mode: driver
    xdp-filter-file:  /etc/suricata/xdp_filter.bpf
    bypass: yes
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
    ring-size: 300000
    block-size: 2097152 #1048576

  - interface: ens2f1
    threads: 17
    cluster-id: 99
    cluster-type: cluster_qm
    defrag: yes
    # xdp-cpu-redirect: [ "2-18" ]
    xdp-mode: driver
    xdp-filter-file:  /etc/suricata/xdp_filter.bpf
    bypass: yes
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
    ring-size: 300000
    block-size: 2097152 #1048576

  - interface: ens6f1
    threads: 17
    cluster-id: 98
    cluster-type: cluster_qm
    defrag: yes
    # xdp-cpu-redirect: [ "2-18" ]
    xdp-mode: driver
    xdp-filter-file:  /etc/suricata/xdp_filter.bpf
    bypass: yes
    use-mmap: yes
    mmap-locked: yes
    tpacket-v3: yes
    ring-size: 300000
    block-size: 2097152 #1048576
```

Also, to maintain NUMA locality, worker threads were bound to cores on different NUMA nodes. This resulted in 4 different configurations for the interfaces in af-packet.

```
 cpu-affinity:
    - management-cpu-set:
        cpu: [ 0,18,36,54 ]  # include only these CPUs in affinity settings
    - worker-cpu-set:
        cpu: [ "1-17", "19-35", "37-53", "55-71" ]
        mode: "exclusive"
        # Use explicitely 3 threads and don't compute number by using
        # detect-thread-ratio variable:
        # threads: 3
        prio:
          low: [ ]
          medium: [ 0,18,36,54]
          high: [ "1-17","19-35","37-53","55-71" ]
          default: "high"
```

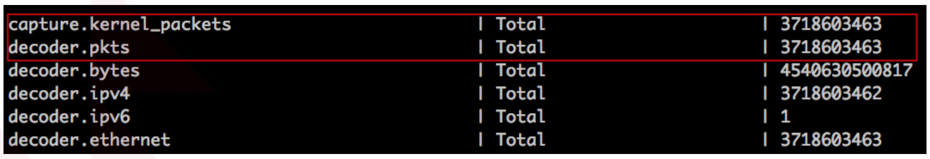
Test Results

DUT CPU Utilization



```
  1  [||||                  8.0%]   19 [||||                  7.9%]   37 [|                     0.7%]   55 [|||||||||||||||      50.3%]
  2  [|||||||||||||||||||||86.7%]   20 [|||||||||||||||||||||95.4%]   38 [|||||||||||||||||||||83.9%]   56 [|||||||||||||||||||||94.0%]
  3  [|||||||||||||||||||||85.2%]   21 [|||||||||||||||||||||97.4%]   39 [|||||||||||||||||||||82.8%]   57 [|||||||||||||||||||||99.3%]
  4  [|||||||||||||||||||||84.4%]   22 [|||||||||||||||||||||92.7%]   40 [|||||||||||||||||||||81.1%]   58 [|||||||||||||||||||||91.3%]
  5  [|||||||||||||||||||||84.6%]   23 [|||||||||||||||||||||83.3%]   41 [|||||||||||||||||||||82.3%]   59 [|||||||||||||||||||||85.5%]
  6  [|||||||||||||||||||||82.8%]   24 [|||||||||||||||||||||91.9%]   42 [|||||||||||||||||||||87.4%]   60 [|||||||||||||||||||||92.1%]
  7  [|||||||||||||||||||  76.0%]   25 [|||||||||||||||||||||90.0%]   43 [|||||||||||||||||||  76.6%]   61 [|||||||||||||||||||||88.5%]
  8  [|||||||||||||||||||  77.1%]   26 [|||||||||||||||||||||84.8%]   44 [||||||||||||||||||   73.4%]   62 [|||||||||||||||||||||88.6%]
  9  [||||||||||||||||||||  79.2%]  27 [|||||||||||||||||||||86.9%]   45 [|||||||||||||||||||  76.6%]   63 [|||||||||||||||||||||89.2%]
  10 [|||||||||||||||||||||84.4%]   28 [|||||||||||||||||||||87.3%]   46 [|||||||||||||||||||||89.9%]   64 [|||||||||||||||||||||84.0%]
  11 [|||||||||||||||||||||81.5%]   29 [|||||||||||||||||||||88.4%]   47 [|||||||||||||||||||||90.1%]   65 [|||||||||||||||||||||86.3%]
  12 [||||||||||||||||||||  80.8%]  30 [|||||||||||||||||||||89.9%]   48 [||||||||||||||||||||  78.9%]  66 [|||||||||||||||||||||91.8%]
  13 [|||||||||||||||||||||87.9%]   31 [|||||||||||||||||||||95.3%]   49 [|||||||||||||||||||||89.0%]   67 [|||||||||||||||||||||99.3%]
  14 [||||||||||||||||||||  81.2%]  32 [|||||||||||||||||||||99.3%]   50 [||||||||||||||||||||  80.5%]  68 [|||||||||||||||||||||98.7%]
  15 [|||||||||||||||||    70.3%]   33 [|||||||||||||||||||||90.5%]   51 [||||||||||||||||||   72.2%]   69 [|||||||||||||||||||||88.4%]
  16 [|||||||||||||||||||  76.0%]   34 [|||||||||||||||||||||87.5%]   52 [||||||||||||||||||||  78.6%]  70 [|||||||||||||||||||||88.5%]
  17 [|||||||||||||||||||  77.7%]   35 [|||||||||||||||||||||92.7%]   53 [||||||||||||||||||||  78.9%]  71 [|||||||||||||||||||||91.8%]
  18 [|||||||||||||||||||  77.7%]   36 [|||||||||||||||||||||99.4%]   54 [||||||||||||||||||||  79.6%]  72 [||||||||||||||||||||100.0%]
  Mem[|||||||||||||||||||||||||||||||||||                50.1G/126G]   Tasks: 52, 102 thr; 56 running
  Swp[|                                                6.25M/8.00G]   Load average: 39.49 19.20 18.50
                                                                      Uptime: 04:09:39

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
35558 root       20   0 50.8G 48.7G 31.8G S 5963 38.7  1h16:28 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
35588 root       18  -2 50.8G 48.7G 31.8G R 103. 38.7  1:13.79 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
35626 root       18  -2 50.8G 48.7G 31.8G R 102. 38.7  1:15.63 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
35592 root       18  -2 50.8G 48.7G 31.8G R 102. 38.7  1:15.91 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
35621 root       18  -2 50.8G 48.7G 31.8G R 102. 38.7  1:13.66 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
35611 root       18  -2 50.8G 48.7G 31.8G R 102. 38.7  1:09.57 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
35622 root       18  -2 50.8G 48.7G 31.8G R 101. 38.7  1:14.68 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
35577 root       18  -2 50.8G 48.7G 31.8G S 100. 38.7  1:13.18 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
35587 root       18  -2 50.8G 48.7G 31.8G R 97.3 38.7  1:14.03 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
35576 root       18  -2 50.8G 48.7G 31.8G S 96.6 38.7  1:12.86 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
35610 root       18  -2 50.8G 48.7G 31.8G R 95.3 38.7  1:12.21 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
35578 root       18  -2 50.8G 48.7G 31.8G R 94.6 38.7  1:13.97 suricata -c /etc/suricata/suricata-3.yaml --af-packet -D -vvv
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Suricata Stats



```
capture.kernel_packets                        | Total                     | 3718603463
decoder.pkts                                  | Total                     | 3718603463
decoder.bytes                                 | Total                     | 4540630500817
decoder.ipv4                                  | Total                     | 3718603462
decoder.ipv6                                  | Total                     | 1
decoder.ethernet                              | Total                     | 3718603463
```

Enhancements and Future Work

While the tests have given good results with 2 receive interfaces on different NUMA nodes achieving over 60gbps, the forwarding capacity of the complete system with 2 WAN and 2 LAN interfaces was bound to 30Gbps. This can be enhanced by

- Increasing the number of sockets in the system (upgrading the motherboard). This would increase the number of NUMA nodes and in turn the capacity of the system.

- Increasing the number of cores in the system (upgrading the processor)

Also, XDP did not come into picture in our setup since our traffic replay was mostly small flows. In case of elephant flow, we believe that XDP will give us good results as well. We are looking forward to testing it on our setups.

Future work will involve performing the same tests on other Crystal Eye firmware series models. We also aim to do performance tests on AMD Ryzen hardware in the near future. We will are also planning to test cluster Crystal Eye deployments running with packet brokers to achieve optimal throughputs above 100Gbps and beyond.

Authored By
Nidhi V Singhai
Red Piranha Limited

COMPANY PROFILE

Red Piranha is an Australian enterprise that engineers and manufactures advanced security products for Managed Service Providers (MSP) and enterprises to give them an advantage in fighting off cyber-crime, intrusion attempts and in securing their data to meet modern compliance requirements. Our crown jewel, Crystal Eye, is a unified threat management platform designed to be easy enough for enterprises to use and powerful enough for MSPs to see as a major game-changer when compared against the top products in the industry.

For more information, visit www.redpiranha.net

References

1. Septun Mark I
2. SEPTun Mark II
3. https://access.redhat.com/sites/default/files/attachments/20150325_network_performance_tuning.pdf
4. https://www.intel.com/content/dam/www/public/us/en/documents/reference-guides/xl710-x710-performance-tuning-linux-guide.pdf
5. http://pevma.blogspot.com/2015/10/suricata-with-afpacket-memory-of-it-all.html
6. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/performance_tuning_guide/sect-

red_hat_enterprise_linux-performance_tuning_guide-memory-configuring-huge-pages