Suricata Performance White Paper

Jonathan H. Stammler

Fall 2011

Contents

INTRO

Suricata is an Open Source Intrusion Detection System developed and released its first stable version in July 2010.

Suricata is a unique Intrusion Detection System (IDS) because of the features it offers. Some of these features are multi-threading, HTP parser and CPU handling capabilities to name just a few. This experiment demonstrates Suricata's capability on a high-bandwidth network with different configurations under a fixed hardware set. These tests show the effects different configurations and a growing rule set. Included in the appendix section of this document is up-to-date Getting Started with Suricata installation guides for Ubuntu, Debian, Centos, and FreeBSD.

Note: No personally identifiable information was collected from the network traffic during this experiment. Only packet, memory, and CPU statistics were collected.

TEST ENVIORMENT

This experiment was tested on a network that transmitted roughly 25 mb/second line. Suricata was run on a Debian 6 squeeze OS. The physical hardware specifications are the following: a quad core processer Intel i7 2.67 with hyper-threading, 12 gigs of ram, and an Intel Corporation 82576 Gigabit Network Connection network interface card (NIC).

It is important to note that the quad core processor with hyper-threading enabled allows for 2 threads per core. This means CPU utilization of 100% is really as high as 800% theoretically and while 800% can't be fully reached, this perspective helps give insight to how hard the CPU is working in the following CPU load and utilization graphs.

The different configurations were run for roughly 4-7 days; each monitoring the CPU load, memory usage, and network traffic load. The version of Suricata used was the Current Version SVN checkout in October 2011.

EXPERIMENT

The experiment consisted of 6 different tests each with a different configuration of Suricata. The first five tests used the free emerging threats rule sets. The last test included the pro emerging rule sets. All changes were made to Suricata's configuration file labeled suricata.yaml. The same Suricata instance was stopped , reconfigured and then restarted for each test.

Data was collected from Suricata itself to provide data on actual attacks and statistics of the traffic being analyzed. Each test also recorded the bits in/out of the network traffic Suricata was receiving as well as the CPU load and memory usage of the machine the tests were running on.

TEST RESULTS

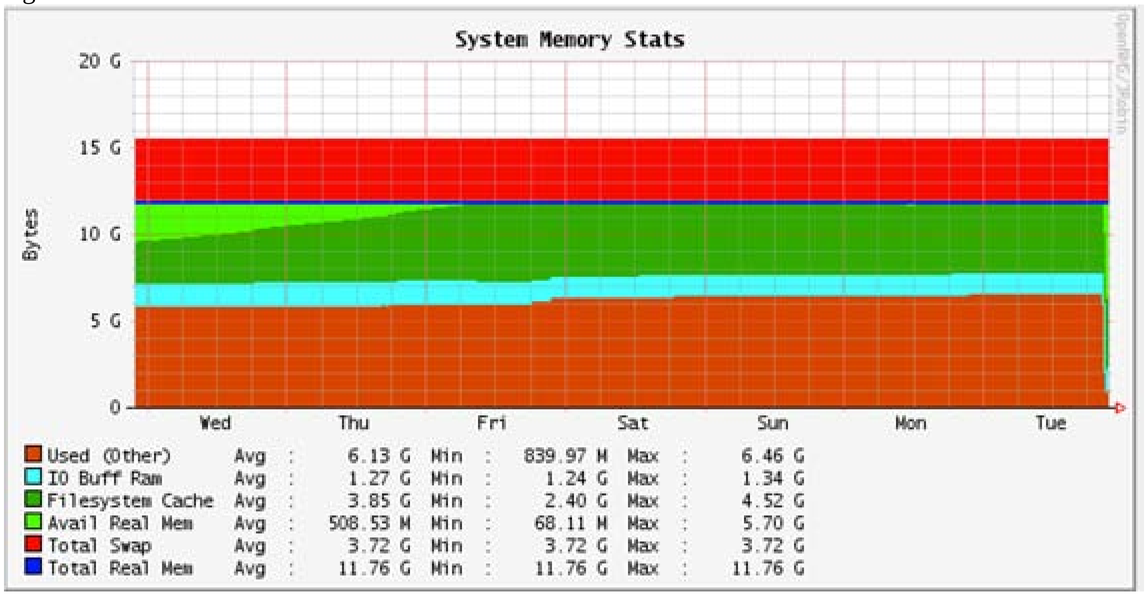
Test1

The first test of Suricata was the basic install and setup through the installation guide that can be found at http://www.openinfosecfoundation.org . The rules used for this test were the free emerging threats rule sets. No changes were made to the suricata.yaml. This test was to create a baseline of Suricata without any features or configuration changes. The suricata.yaml configuration file has default settings have the CPU affinity was set to 'no' and detect_thread_ratio was set at 1.5.

In figure 1.1 Suricata slowly starts using up available memory but never gets close to taking out the I/O buffer or file-system cache. Comparing figure 1.2 the CPU utilization and figure 1.3 the network throughput shows that the CPU usage is correlated with the increased amount of data traffic Suricata is inspecting. The increase from Friday to Saturday shows the network at around 600 megabytes where the CPU utilization 300% of its capability.

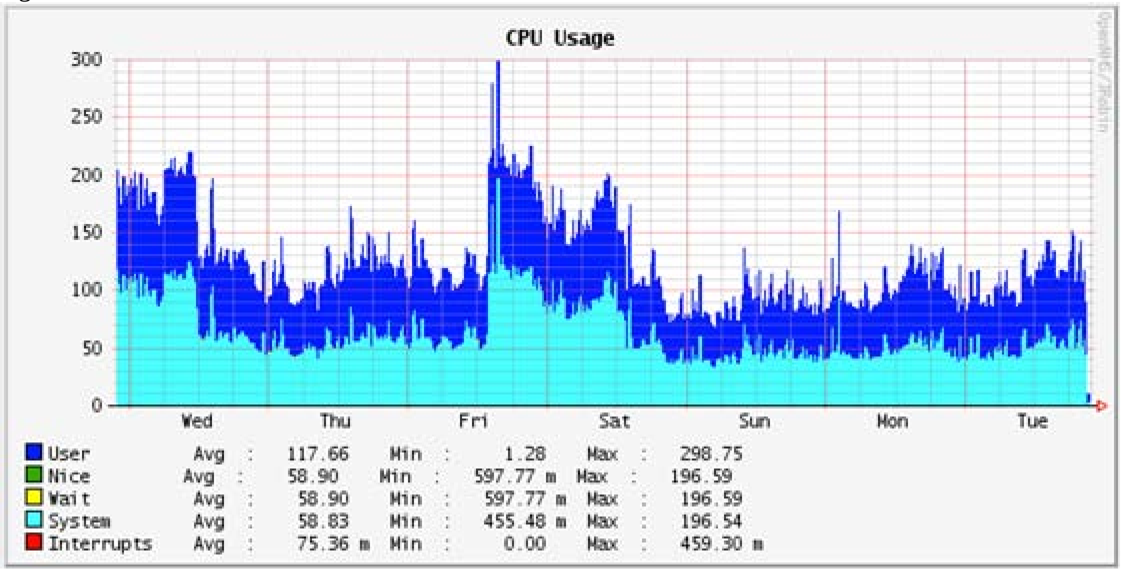
According to the Suricata's stats.log no packets were dropped in test1.

Figure 1.1



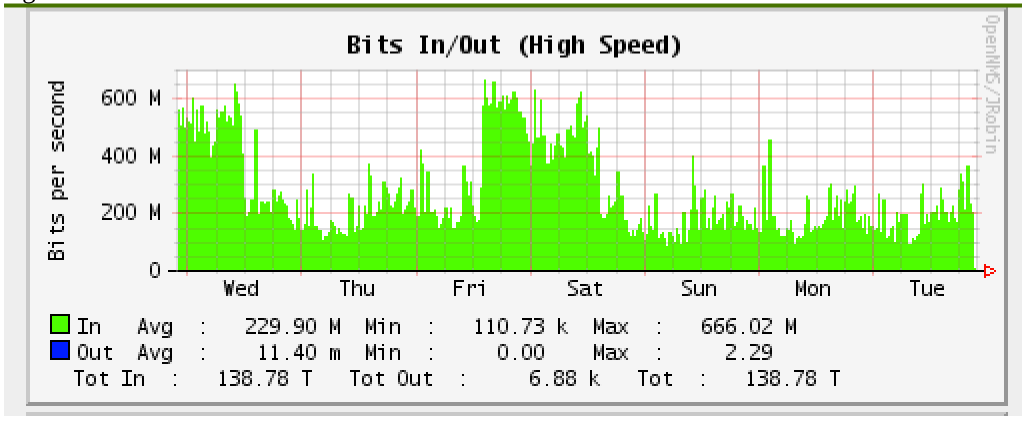
🟧	Used (Other)	Avg	:	6.13 G	Min	:	839.97 M	Max	: 6.46 G
🟦	IO Buff Ram	Avg	:	1.27 G	Min	:	1.24 G	Max	: 1.34 G
🟩	Filesystem Cache	Avg	:	3.85 G	Min	:	2.40 G	Max	: 4.52 G
🟩	Avail Real Mem	Avg	:	508.53 M	Min	:	68.11 M	Max	: 5.70 G
🟥	Total Swap	Avg	:	3.72 G	Min	:	3.72 G	Max	: 3.72 G
🟦	Total Real Mem	Avg	:	11.76 G	Min	:	11.76 G	Max	: 11.76 G

Figure: 1.2



```
🟦 User        Avg  :  117.66   Min  :    1.28     Max  :  298.75
🟩 Nice       Avg  :   58.90   Min  :  597.77 m   Max  :  196.59
🟨 Wait        Avg  :   58.90   Min  :  597.77 m   Max  :  196.59
🟦 System      Avg  :   58.83   Min  :  455.48 m   Max  :  196.54
🟥 Interrupts  Avg  :   75.36 m Min  :    0.00     Max  :  459.30 m
```

Figure: 1.3



```
🟩 In   Avg  :   229.90 M  Min  :   110.73 k   Max  :   666.02 M
🟦 Out  Avg  :    11.40 m  Min  :     0.00     Max  :     2.29
  Tot In  :   138.78 T   Tot Out  :     6.88 k    Tot  :   138.78 T
```

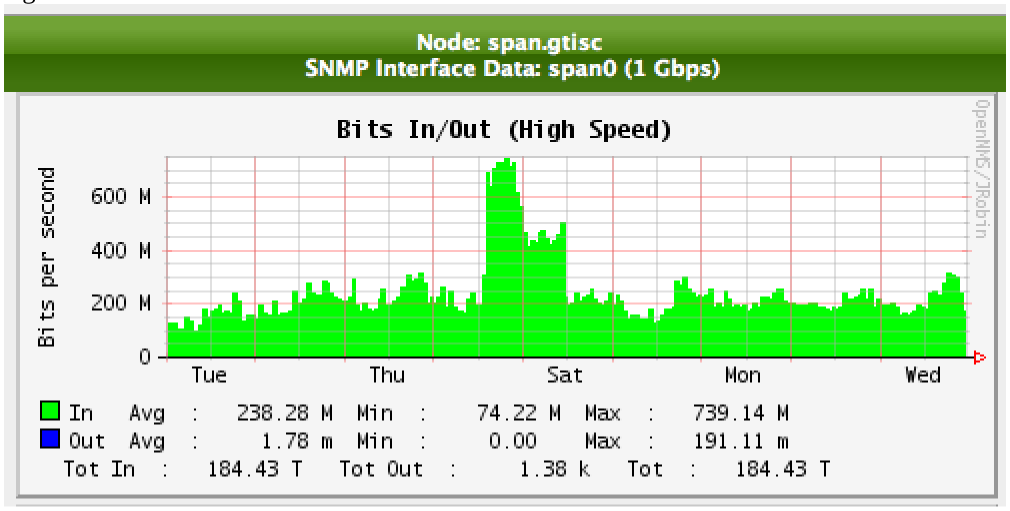
TEST2

TEST2 was used to see the effect of Suricata configuration by enabling Suricata’s multithreading capability. The changes made to the Suricata.yaml file were the following: changing the detect_thread_ratio from 1.5 threads to 2 threads per core.

Figure2.1 depicts the amount of real network traffic Suricata inspected during test2. The total size of data inspected by Suricata was 184.43 terabytes and the network was saturated with over 700 mbs at one point. Figure 2.3 and Figure 2.4 depict clearing that Suricata was able to handle this spike in network traffic with a small increase in CPU Utilization and little to no increase in CPU load. According to the stats.log no packets were lost during Test2.

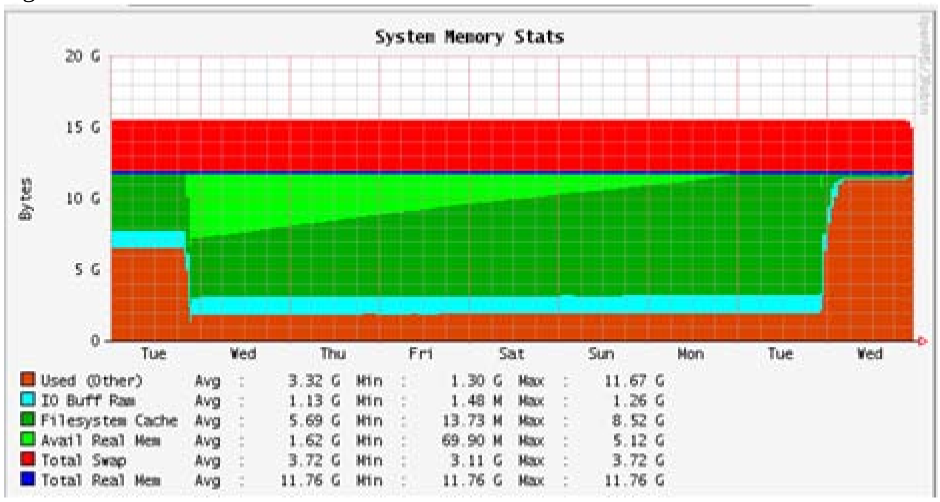
The increase in memory and CPU utilization towards the end of test2 on Wednesday seen in figure 2.2 and 2.3 was a result of another user transferring large amounts of data from the system Suricata was running on.

Figure2.1



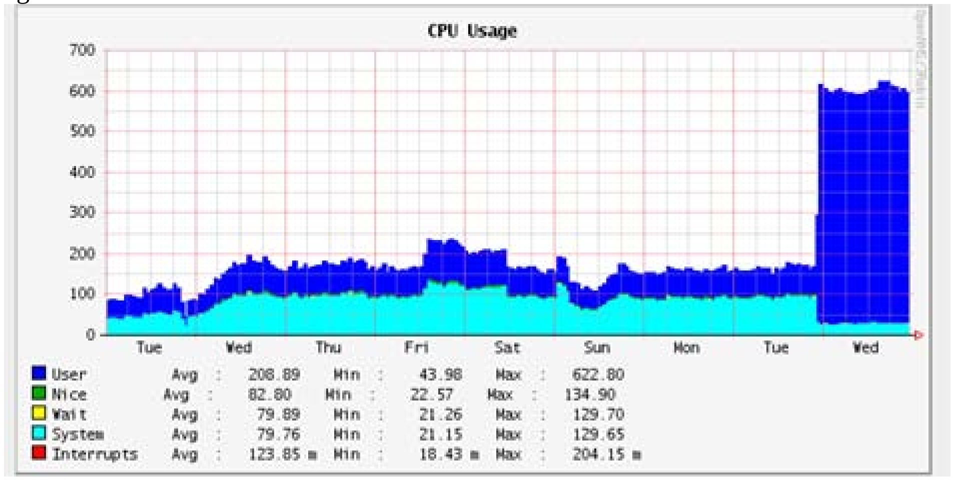
```
🟩 In   Avg  :   238.28 M  Min  :    74.22 M  Max  :   739.14 M
🟦 Out  Avg  :     1.78 m  Min  :     0.00    Max  :   191.11 m
  Tot In  :   184.43 T   Tot Out  :     1.38 k   Tot  :   184.43 T
```

Figure 2.2



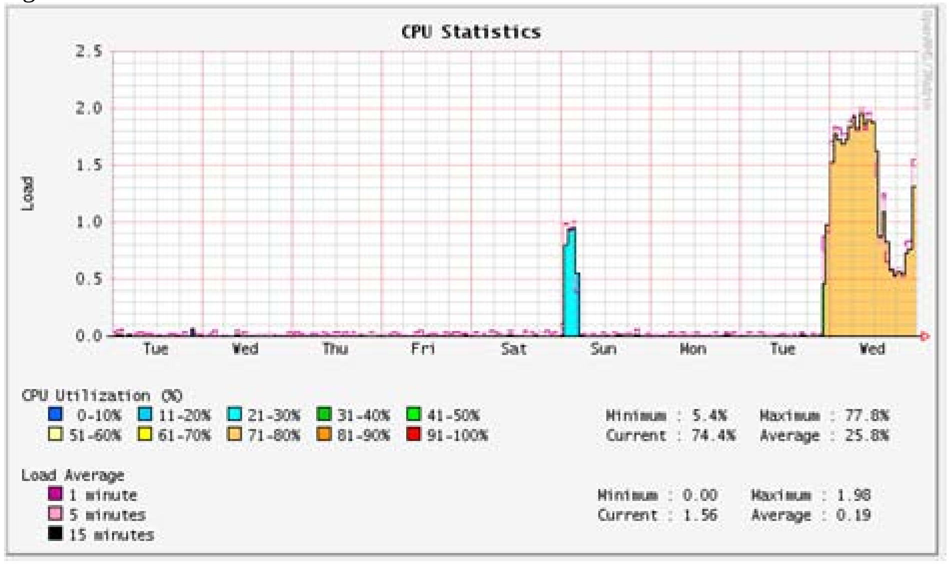
```
🟧 Used (Other)      Avg  :   3.32 G  Min  :    1.30 G  Max  :   11.67 G
🟦 IO Buff Ram       Avg  :   1.13 G  Min  :    1.48 M  Max  :    1.26 G
🟩 Filesystem Cache  Avg  :   5.69 G  Min  :   13.73 M  Max  :    8.52 G
🟩 Avail Real Mem    Avg  :   1.62 G  Min  :   69.90 M  Max  :    5.12 G
🟥 Total Swap        Avg  :   3.72 G  Min  :    3.11 G  Max  :    3.72 G
🟦 Total Real Mem    Avg  :  11.76 G  Min  :   11.76 G  Max  :   11.76 G
```

figure2.3



```
🟦 User        Avg  :   208.89    Min  :    43.98    Max  :   622.80
🟩 Nice       Avg  :   82.80     Min  :   22.57     Max  :  134.90
🟨 Wait        Avg  :    79.89    Min  :    21.26    Max  :   129.70
🟦 System      Avg  :    79.76    Min  :    21.15    Max  :   129.65
🟥 Interrupts  Avg  :   123.85 m  Min  :    18.43 m  Max  :   204.15 m
```

Figure2.4



TEST3

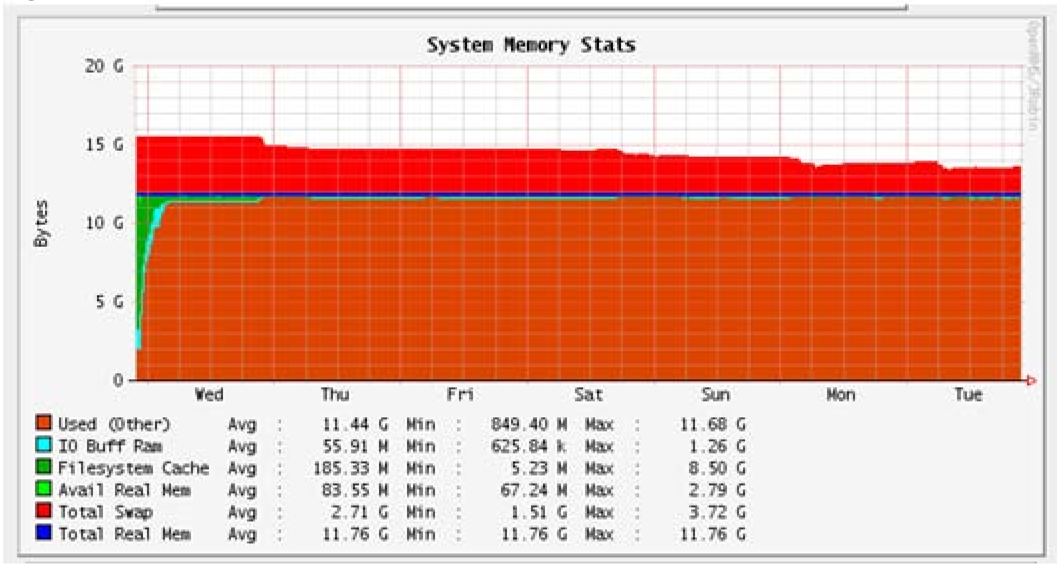
Test3 was the first test to rebuild Suricata to enable the use of PF_RING capture accelerator. PF_RING would likely be used by any IDS required analyze a large bandwidth line without packet loss. PF_RING polls packets from the Network interface card into a ring buffer in memory and allows the Suricata to read the packets from the ring buffer at its own speed without dropping packets and using less CPU to handle the packets.

The installation of PF_RING with Suricata can be found in the appendix section for Debian, Ubuntu, and CentOS. The changes made to the Suricata.yaml were the same as Test2. The detect_thread_ratio was set to 2 threads. According to the Suricata's stats.log no memcap session drops during Test3.

In Figure 3.1 the memory stats are very different from what was seen in Test2 figure 2.1. The start of test3, Suricata quickly took over available memory at a faster rate and in totality double the memory to run Suricata with PF_RING enabled.

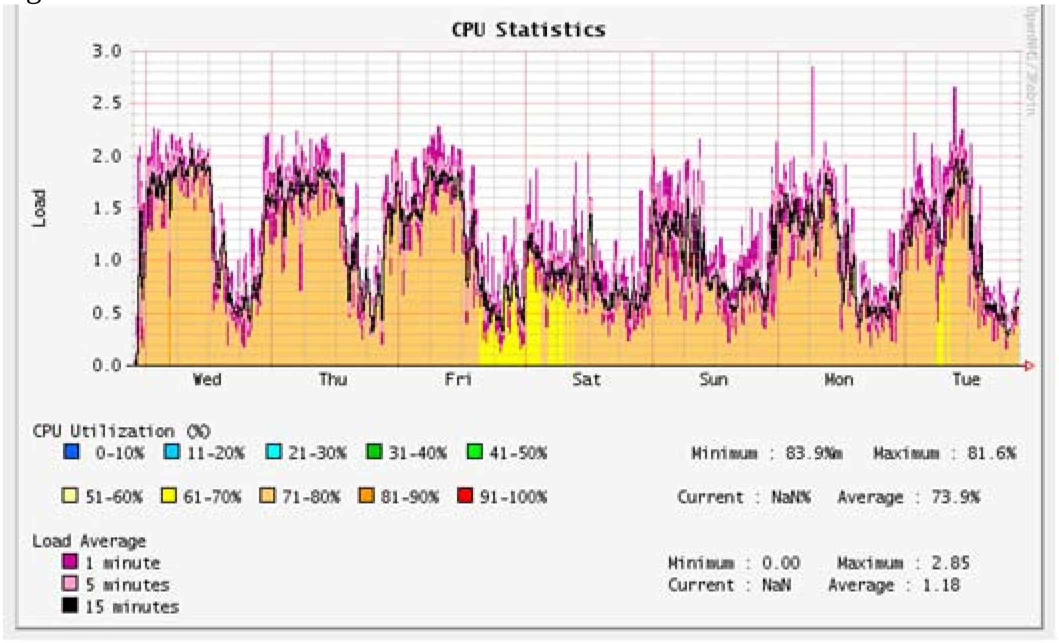
Looking at the results in figure 3.2 and figure 3.3 we can see as the network traffic increased the CPU utilization decreased. This is a result of PF_RING using memory to buffer the packets and leaving more CPU load to other tasks. In figure 3.3 the highest amount of network traffic was between Friday and Saturday well over 700 mbs. The average CPU load utilization was around 73%. However, the CPU load of a quad-core system is 4.0 and the load average was well below around 2.0 CPU load. This test shows that this configure of Suricata with PF_RING with the hardware configuration is inconclusive. This is because looking again at figure 3.1 The use of swap space is increasing in use. This configuration would have a decrease in Suricata analysis performance with a larger load of packets per second.

Figure: 3.1



```
🟥 Used (Other)        Avg  :   11.44 G  Min  :  849.40 M  Max  :  11.68 G
🟦 IO Buff Ram         Avg  :   55.91 M  Min  :  625.84 k  Max  :   1.26 G
🟩 Filesystem Cache    Avg  :  185.33 M  Min  :    5.23 M  Max  :   8.50 G
🟩 Avail Real Mem      Avg  :   83.55 M  Min  :   67.24 M  Max  :   2.79 G
🟥 Total Swap          Avg  :    2.71 G  Min  :    1.51 G  Max  :   3.72 G
🟦 Total Real Mem      Avg  :   11.76 G  Min  :   11.76 G  Max  :  11.76 G
```

Figure: 3.2

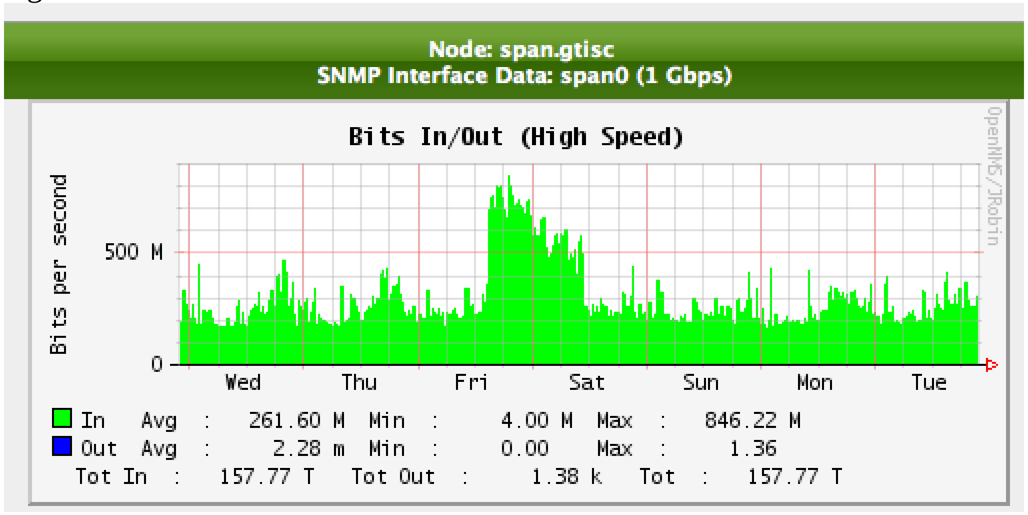


```
CPU Utilization (%)
  🟦 0-10%  🟦 11-20%  🟦 21-30%  🟩 31-40%  🟩 41-50%        Minimum : 83.9%m   Maximum : 81.6%

  🟨 51-60%  🟨 61-70%  🟧 71-80%  🟧 81-90%  🟥 91-100%      Current : NaN%   Average : 73.9%

Load Average
  🟪 1 minute                                               Minimum : 0.00    Maximum : 2.85
  🟪 5 minutes                                              Current : NaN    Average : 1.18
  ⬛ 15 minutes
```

Fig3.3



```
🟩 In    Avg  :   261.60 M  Min  :    4.00 M  Max  :   846.22 M
🟦 Out   Avg  :     2.28 m  Min  :    0.00    Max  :     1.36
  Tot In  :   157.77 T   Tot Out  :    1.38 k   Tot  :   157.77 T
```

TEST4

Test4 was another test of Suricata with PF_RING installed. The changes made to the Suricata.yaml file double the number of threads. The detect_thread_ratio was changed from 2 threads to 4 threads.

The CPU statistics came in as expected where in test3 the CPU load was around 2.0 and in figure 4.1 of test4 the CPU utilization was upper bounds of 2.0 and over 3.0 during the test4.

In comparison of the CPU usage to figure 4.3 the network traffic was significantly less than previous tests with 56.94 terabytes. The spike towards the end of test4 in figure4.1 shows the inverse relationship of CPU load and utilization decreasing while the network traffic increase to over 600 mbs. This was similar to the results seen in test3 and opposite to the results seen in Test1 and Test2.

Figure 4.2 shows similar results of memory usage being quickly used up. However, the swap is less affected then in test3. The stats.log registered a memcap drop during test4. Memcap_drop in the Suricata means that there was not enough memory to sufficiently complete an application and flow layer analysis for a number of packets. Please note that each individual packet was individually inspected by Suricata. The results of test4 show that while the CPU load never reaches 5.0 load and as mentioned before the system can handle almost 8.0 CPU load. However, this end of this test shows evidence that the memory in this hardware configuration is not sufficient to effectively run Suricata at a consistently higher network bandwidth rate.

Figure 4.1

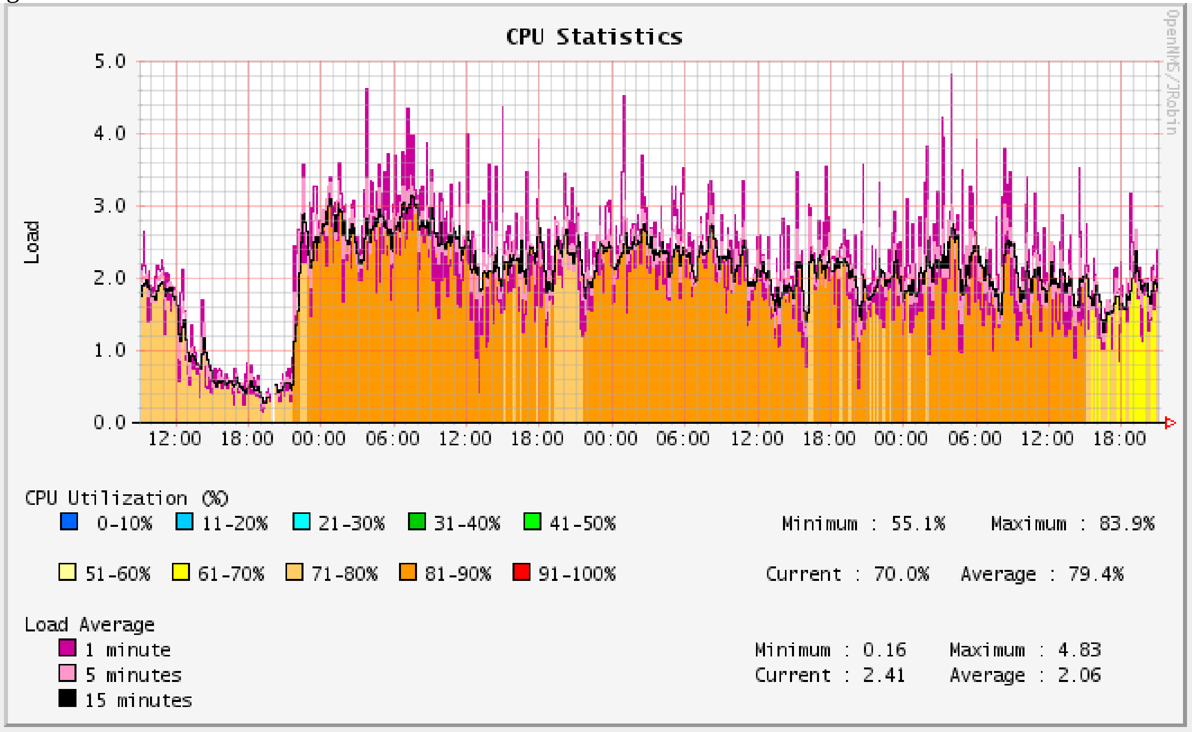
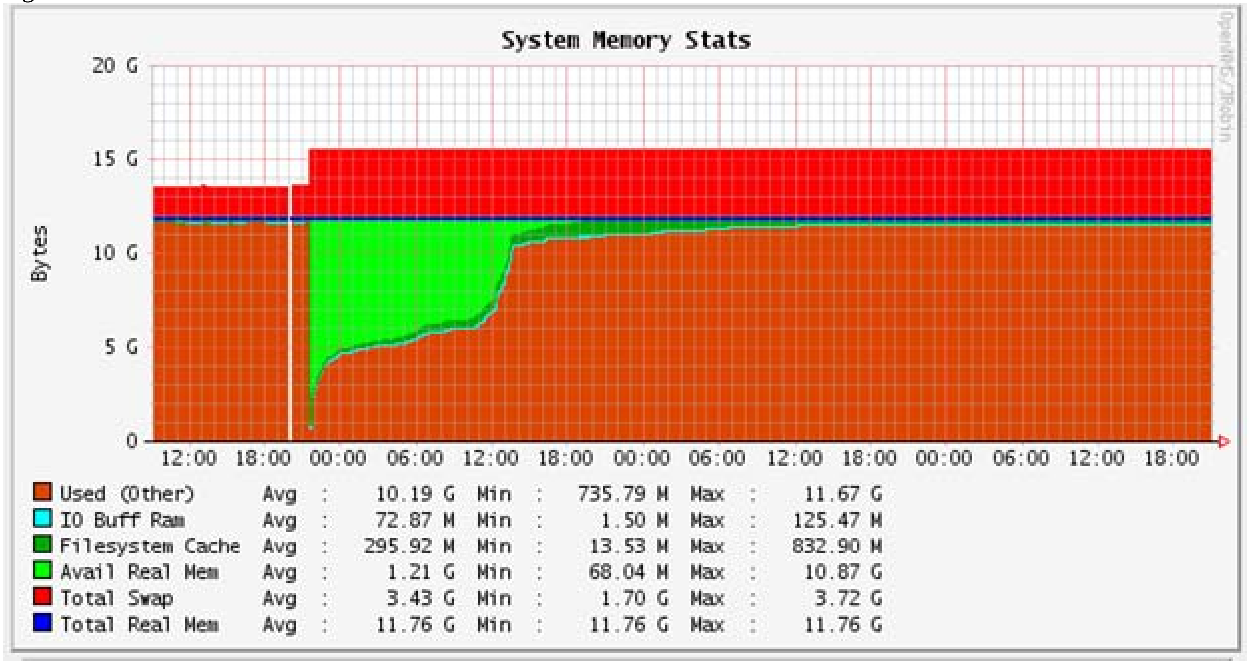
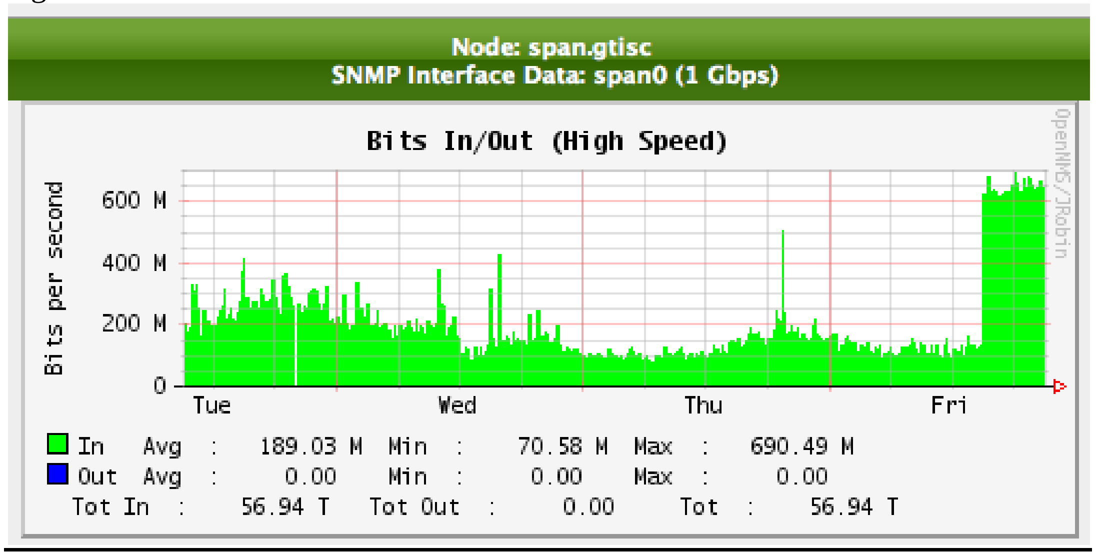


Figure 4.2



```
🟧 Used (Other)        Avg  :   10.19 G  Min  :   735.79 M  Max  :    11.67 G
🟦 IO Buff Ram         Avg  :   72.87 M  Min  :     1.50 M  Max  :   125.47 M
🟩 Filesystem Cache    Avg  :  295.92 M  Min  :    13.53 M  Max  :   832.90 M
🟩 Avail Real Mem      Avg  :    1.21 G  Min  :    68.04 M  Max  :    10.87 G
🟥 Total Swap          Avg  :    3.43 G  Min  :     1.70 G  Max  :     3.72 G
🟦 Total Real Mem      Avg  :   11.76 G  Min  :    11.76 G  Max  :    11.76 G
```

Figure 4.3



```
🟩 In   Avg  :   189.03 M  Min  :    70.58 M  Max  :   690.49 M
🟦 Out  Avg  :     0.00    Min  :     0.00    Max  :     0.00
  Tot In  :    56.94 T  Tot Out  :     0.00     Tot  :    56.94 T
```

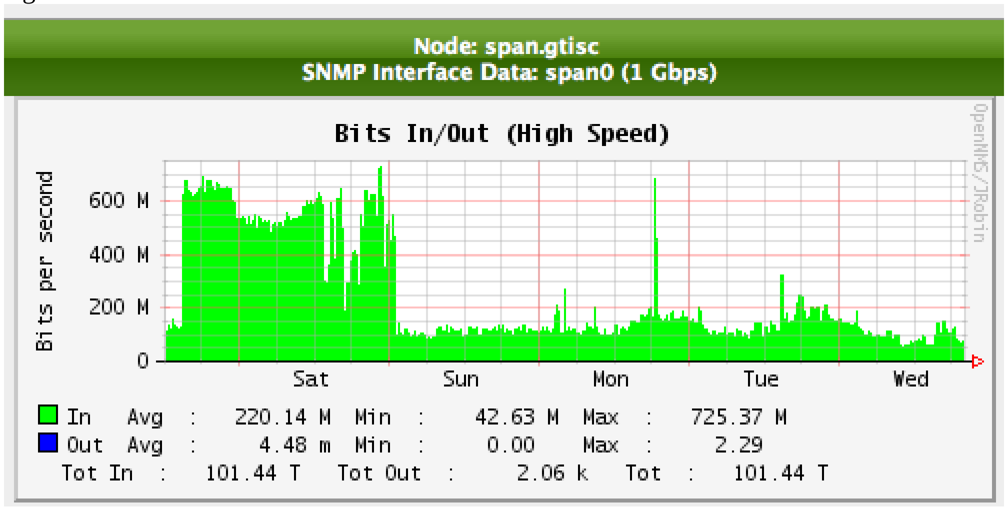
TEST5

Test5 was Suricata with PF_RING enabled with another increase in the detect_thread_ratio from 4 threads to 6 threads. Test5 shows similar results test3 and test4. Figure 5.1 shows as expected the increased load. Figure 5.2 also shows the inverse relationship of the CPU utilization and the network traffic.

In figure 5.3, it is notable after the available memory has been used up the swap space is used completely at the end of the experiment except for .5 gigs of memory. Figure 5.4 shows that the CPU utilization at over 600% for the duration of the test out of the theoretical total, 800%. According to the stats.log, test5 had a larger memcap session drop rate.

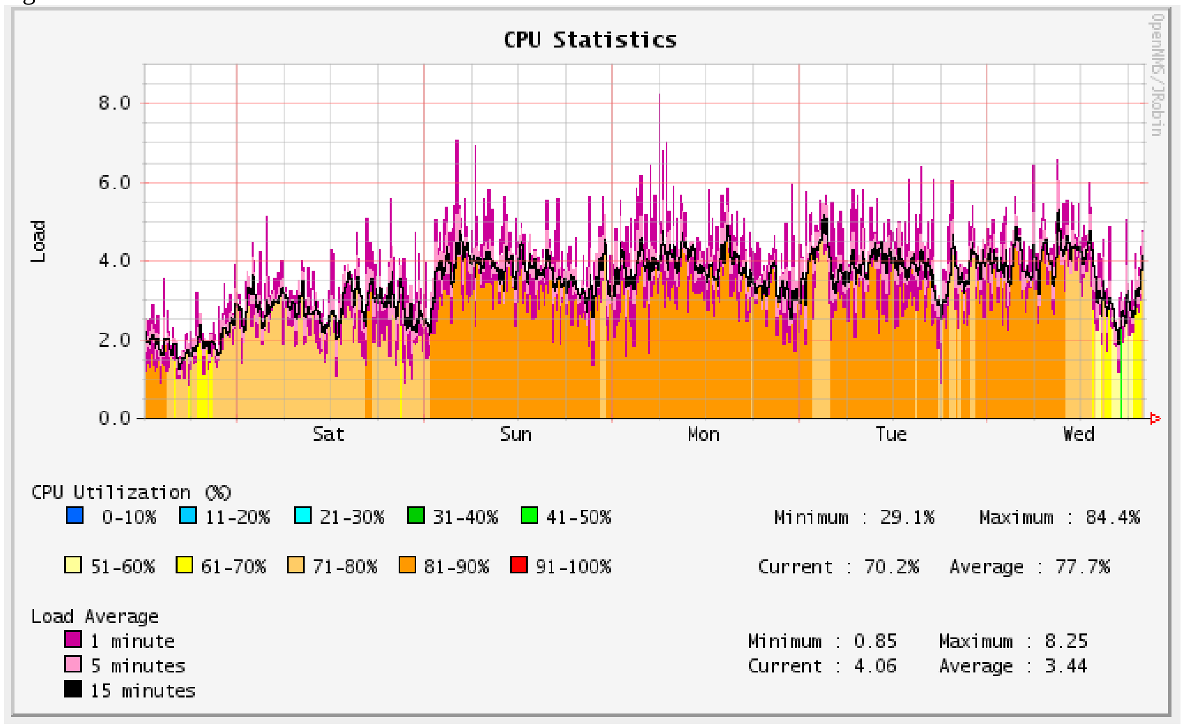
The results of the test5 show that Suricata was using almost all of the hardware's resources in terms of memory and the majority of the CPU usage. Test5 showed that Suricata's performance and analyzing capabilities were reduced because of the number of threads and the use of PF_RING putting more of the network packet load on the memory.

Figure: 5.1



```
🟩 In    Avg  :   220.14 M  Min  :     42.63 M  Max  :    725.37 M
🟦 Out   Avg  :      4.48 m  Min  :      0.00     Max  :      2.29
  Tot In  :   101.44 T   Tot Out  :     2.06 k    Tot  :    101.44 T
```

Figure: 5.2



CPU Utilization (%)

🟦 0-10% 🟦 11-20% 🟦 21-30% 🟩 31-40% 🟩 41-50%   Minimum : 29.1%   Maximum : 84.4%

🟨 51-60% 🟨 61-70% 🟧 71-80% 🟧 81-90% 🟥 91-100%   Current : 70.2%   Average : 77.7%

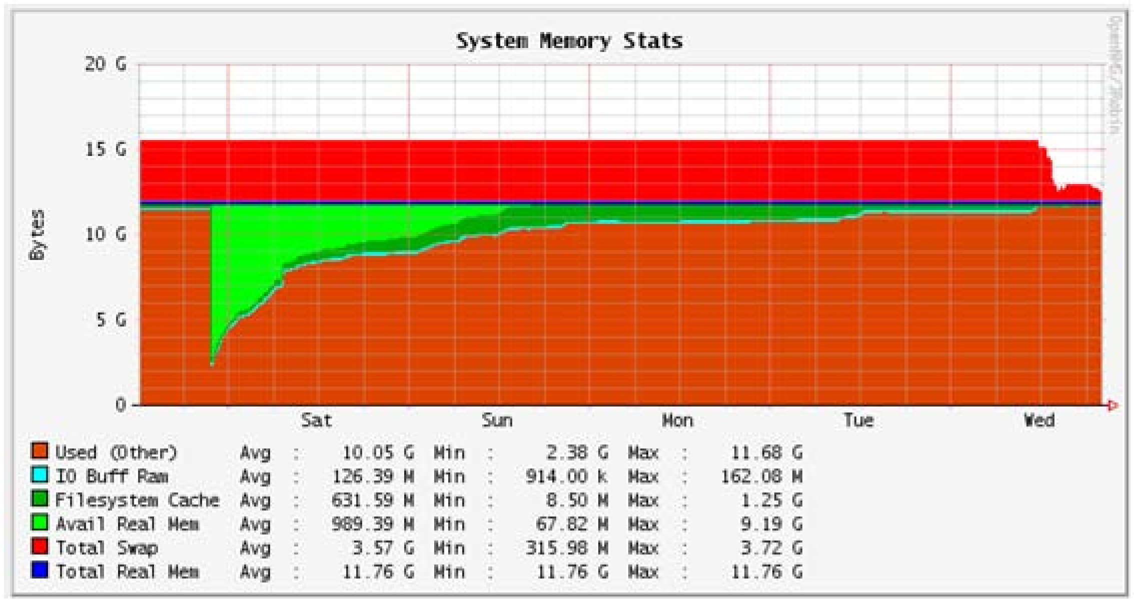
Load Average

🟪 1 minute   Minimum : 0.85   Maximum : 8.25

🟪 5 minutes   Current : 4.06   Average : 3.44

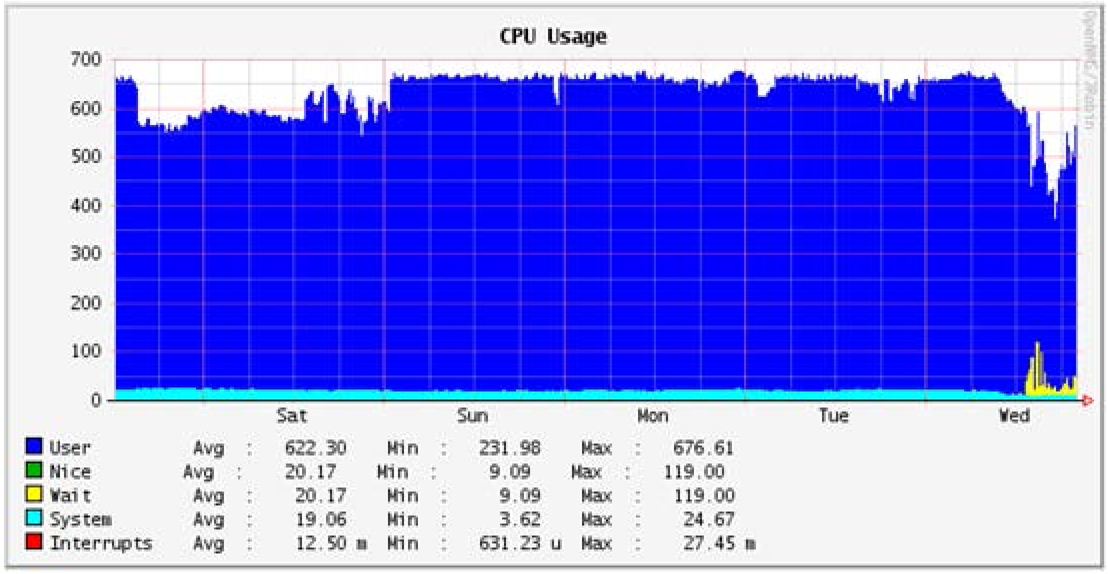
⬛ 15 minutes

Figure 5.3



🟧 Used (Other)	Avg	:	10.05 G	Min	:	2.38 G	Max	: 11.68 G
🟦 IO Buff Ram	Avg	:	126.39 M	Min	:	914.00 k	Max	: 162.08 M
🟩 Filesystem Cache	Avg	:	631.59 M	Min	:	8.50 M	Max	: 1.25 G
🟩 Avail Real Mem	Avg	:	989.39 M	Min	:	67.82 M	Max	: 9.19 G
🟥 Total Swap	Avg	:	3.57 G	Min	:	315.98 M	Max	: 3.72 G
🟦 Total Real Mem	Avg	:	11.76 G	Min	:	11.76 G	Max	: 11.76 G

Figure 5.4



```
🟦 User        Avg  :   622.30    Min  :   231.98    Max  :   676.61
🟩 Nice        Avg  :    20.17    Min  :     9.09    Max  :   119.00
🟨 Wait        Avg  :    20.17    Min  :     9.09    Max  :   119.00
🟦 System      Avg  :    19.06    Min  :     3.62    Max  :    24.67
🟥 Interrupts  Avg  :    12.50 m  Min  :   631.23 u  Max  :    27.45 m
```

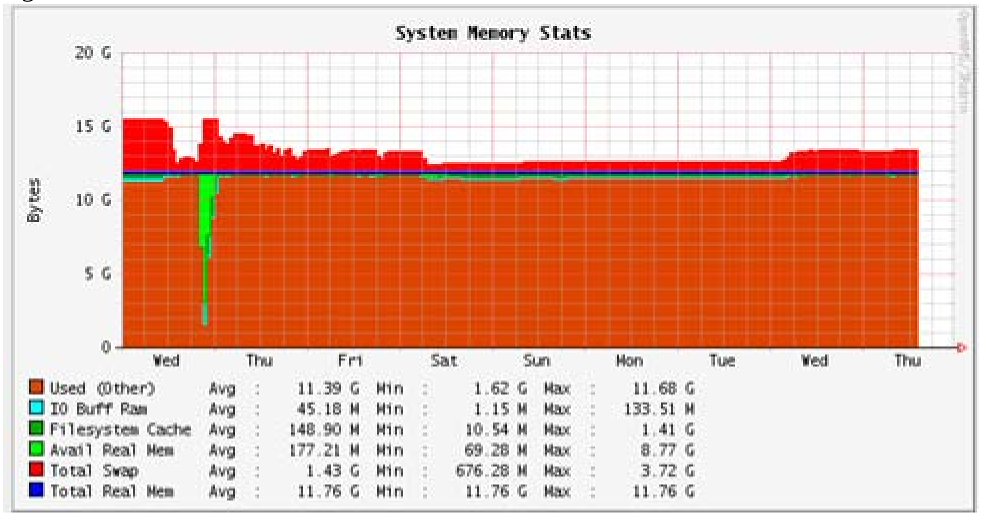
TEST6

Test6 was setup exactly as test4's configuration with PF_RING enabled and the detect_thread_ratio set to 4. Test6 was to determine the effect an increase in a larger rule set from adding the pro emerging threats rule set to Suricata to the existing free emerging threat rule sets.

Figure 6.1 shows test6 for an almost half the test used up about 3.5 gigs of total swap. This is substantially more then what was used in test4 with the free rule set and was used faster than in test5 which had 6 threads. Figure 6.2 shows that the CPU utilization and load were around the levels found in test4 or much lower.

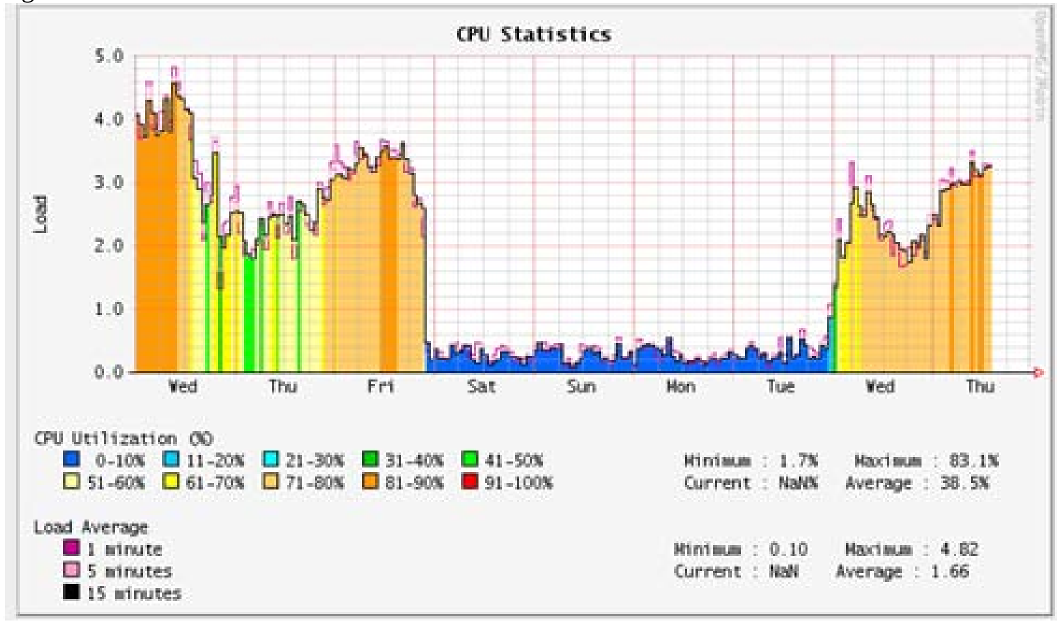
Figure 6.3 shows barely reached over 100 mbs until the end of the experiment. This is likely a result of thanksgiving break. At the end of the test the CPU load climbs back up to 3.0 load which matches the average load from test4. According to the stats.log, it took 2 days and 19 hours for Suricata to register a memcap session drop. This means Suricata was unable to analyze packets per flow or application. Suricata was still able to analyze the packet individually. Suricata needs more memory then available to handle Suricata and PF_RING. The use of swap space in with the memcap session demonstrates a loss in Suricata's analysis capabilities.

Figure6.1



```
🟥 Used (Other)       Avg  :   11.39 G  Min  :    1.62 G  Max  :   11.68 G
🟦 IO Buff Ram        Avg  :   45.18 M  Min  :    1.15 M  Max  :  133.51 M
🟩 Filesystem Cache   Avg  :  148.90 M  Min  :   10.54 M  Max  :    1.41 G
🟩 Avail Real Mem     Avg  :  177.21 M  Min  :   69.28 M  Max  :    8.77 G
🟥 Total Swap         Avg  :    1.43 G  Min  :  676.28 M  Max  :    3.72 G
🟦 Total Real Mem     Avg  :   11.76 G  Min  :   11.76 G  Max  :   11.76 G
```

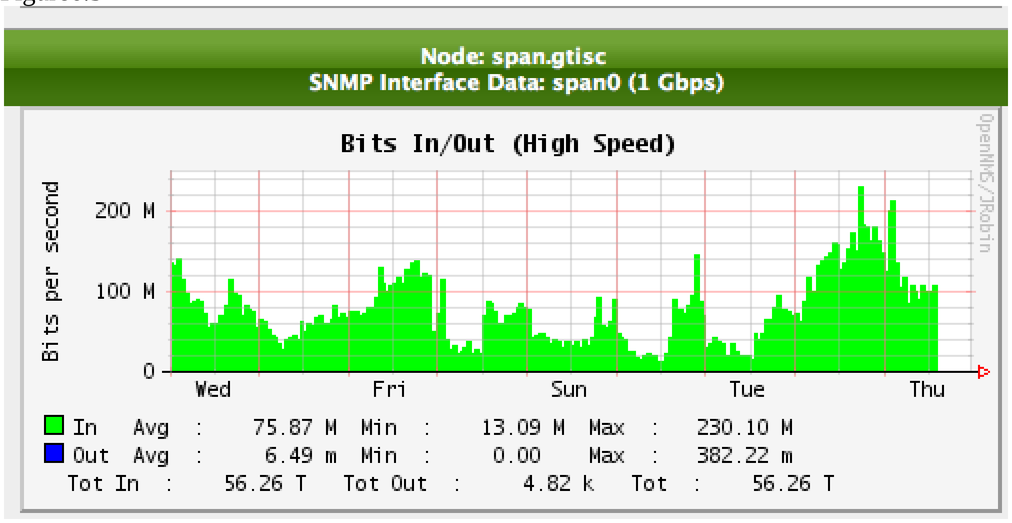
Figure6.2



```
CPU Utilization (%)
  🟦 0-10%   🟦 11-20%  🟦 21-30%  🟩 31-40%  🟩 41-50%        Minimum : 1.7%    Maximum : 83.1%
  🟨 51-60%  🟨 61-70%  🟧 71-80%  🟧 81-90%  🟥 91-100%       Current : NaN%    Average : 38.5%

Load Average
  🟪 1 minute                                                 Minimum : 0.10    Maximum : 4.82
  🟪 5 minutes                                                Current : NaN     Average : 1.66
  ⬛ 15 minutes
```

Figure6.3



Node: span.gtisc
SNMP Interface Data: span0 (1 Gbps)

```
🟩 In    Avg  :    75.87 M  Min  :   13.09 M  Max  :   230.10 M
🟦 Out   Avg  :     6.49 m  Min  :    0.00    Max  :   382.22 m
  Tot In  :    56.26 T   Tot Out  :    4.82 k   Tot  :    56.26 T
```

Conclusion

The tests 1-6 were discussed individually and closely to see changes within each test. The following overall results are all 6 tests compared side by side in network traffic, memory usage, and CPU utilization and usage.

Figure 7.1 shows over the length of the experiment Suricata inspected a total of 636 terabytes and the most traffic was inspected during week46 or test3. More interesting is the comparison in figure 7.2 of memory usage. In figure7.2 each dip in the available/file-system is the start of each test. Test3, the first test using Suricata with PF_RING shows that 12 gigs of memory. The memory on the hardware barely meets the required amount of memory. The swap space from test3 shows more swap space is used when more threads are created with the Suricata engine.

Test6 or the beginning of week49, figure7.2, the pro VT rule set is being utilized with suricata. The swap space being used is significantly more and is easy to see that to run suricata at this level. 12 gigs of memory is inadequate for a 1gig maximum traffic network. Figure 7.3 shows that throughout the tests the CPU utilization was around 70-90%. These tests suggest this hardware specification is sufficient for a network that operates up to 1 gigabyte threshold network and roughly 200 mbs on average without the use of PF_RING.

The results from these tests demonstrate the Suricata can efficiency and without packet loss handle a network of this size under the specified hardware constraints. For PF_Ring feature to be enabled with Suricata running and run Suricata at full analysis capabilities more memory will be required. PF_Ring off loads the CPU load to memory to create the ring of packets to be analyzed at the intrusion detection systems rate. During the last three tests, it was show that there was some memcap drop showed those packets were still being inspected individually. They were not dropped. The packets in the memcap drop due to memory shortage were not analyzed per flow or application. This reduced the analyzing capabilities of Suricata.

Figure7.1

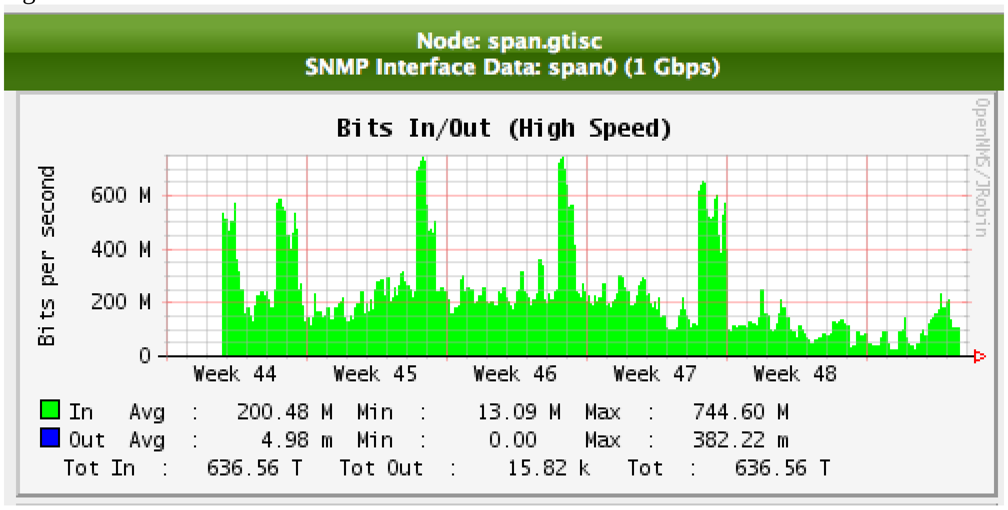
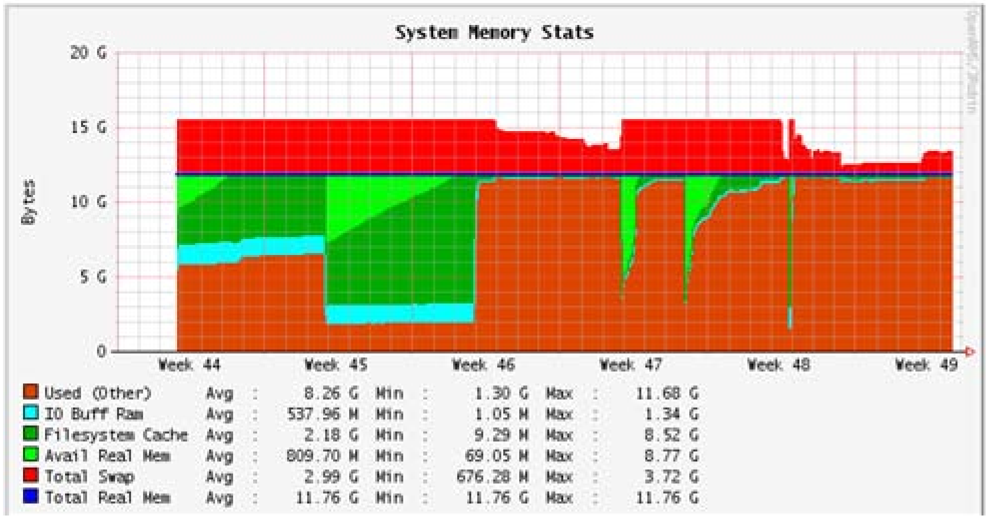
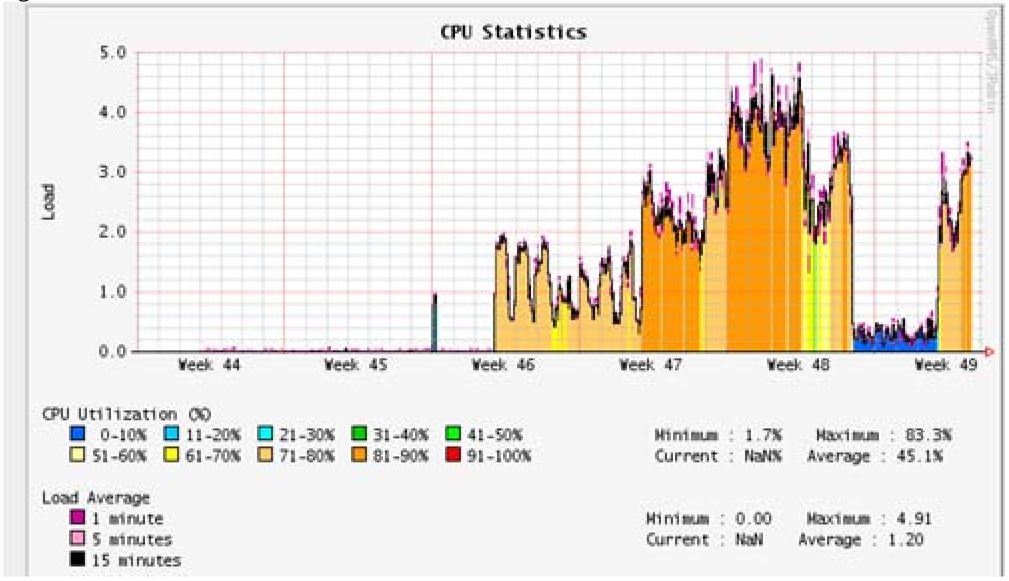


Figure7.2



```
🟫 Used (Other)       Avg  :    8.26 G  Min  :    1.30 G  Max  :   11.68 G
🟦 IO Buff Ram        Avg  :  537.96 M  Min  :    1.05 M  Max  :    1.34 G
🟩 Filesystem Cache   Avg  :    2.18 G  Min  :    9.29 M  Max  :    8.52 G
🟩 Avail Real Mem     Avg  :  809.70 M  Min  :   69.05 M  Max  :    8.77 G
🟥 Total Swap         Avg  :    2.99 G  Min  :  676.28 M  Max  :    3.72 G
🟦 Total Real Mem     Avg  :   11.76 G  Min  :   11.76 G  Max  :   11.76 G
```

Figure7.3



```
CPU Utilization (%)
  🟦 0-10%   🟦 11-20%  🟦 21-30%  🟩 31-40%  🟩 41-50%        Minimum : 1.7%    Maximum : 83.3%
  🟨 51-60%  🟨 61-70%  🟧 71-80%  🟧 81-90%  🟥 91-100%       Current : NaN%    Average : 45.1%

Load Average
  🟪 1 minute                                                 Minimum : 0.00    Maximum : 4.91
  🟪 5 minutes                                                Current : NaN     Average : 1.20
  ⬛ 15 minutes
```

APPENDIX

UBUNTU 11 Getting Started with Suricata

INTRO:

This is a guide to install Suricata with PF_RING. This installation was completed using Virtualbox version 4.0.6 and Ubuntu iso 11.04.

Sections:

1. Basic Suricata installation and how to enable some features
2. PF_RING (capture accelerator) Suricata installation with features from Section 1
3. Suricata Configuration

Required Packages To Build Suricata:

```
sudo apt-get -y install libpcre3 libpcre3-dbg libpcre3-dev

build-essential autoconf automake libtool libpcap-dev libnet1-dev

libyaml-0-2 libyaml-dev zlib1g zlib1g-dev libcap-ng-dev libcap-ng0
```

Install Suricata as and IDS and IPS system

```
#sudo apt-get -y install libnetfilter-queue-dev libnetfilter-queue1 libnfnetlink-dev
libnfnetlink


# cd /opt

# wget http://www.openinfosecfoundation.org/download/suricata-1.0.5.tar.gz

# tar -xvfz suricata-1.0.5.tar.gz

# cd suricata-1.0.5

#sudo ./configure  --enable-nfqueue

#sudo make

#sudo make install
```

To install enable additional features of Suricata , install the following packages and append the ./configure line as stated:

To enable HTP Library(HTML pre-processor):

```
#apt-get install htp

./configure --with-libhtp-libraries
```

To enable libcap_ng (dropping privileges) :

```
#apt-get install libcap-ng-dev

./configure  --with-libcap_ng-libraries=/usr/lib
```

2. PF_RING installation

This installation will install PF_RING as well as enable the above features.

2.1 Download the required packages:

```
#apt-get install build-essential libpcre3-dev libpcap-dev libnet1-dev libyamldev
libnetfilter-queue-dev zlib1g-dev htp subversion flex bison linux- headers-2.6.32-5-686
dkms libcap-ng-dev
```

Note This installation was done with linux-headers-3.0.0-12 and linux-headers-3.0.0-12-generic. Using linux-headers-2.6.32-5 caused issues while building the e1000e-pf_ring module

2.2 Download PF_RING

```
# cd /usr/src
# svn --force export https://svn.ntop.org/svn/ntop/trunk/PF_RING/ PF_RING_CURRENT_SVN
# mkdir /usr/src/pf_ring-4
# cp -Rf /usr/src/PF_RING_CURRENT_SVN/kernel/* /usr/src/pf_ring-4/
```

2.3 Configure PF_RING Driver

```
# cd /usr/src/pf_ring-4/

# nano dkms.conf
```

```
PACKAGE_NAME="pf_ring"
PACKAGE_VERSION="4"
BUILT_MODULE_NAME[0]="pf_ring"
DEST_MODULE_LOCATION[0]="/kernel/net/pf_ring/"
AUTOINSTALL="yes"
```

```
Then press CRTL+X, Y, enter

# dkms add -m pf_ring -v 4
# dkms build -m pf_ring -v 4
# dkms install -m pf_ring -v 4
```

At the end the install command you should get DKMS: INSTALL Completed

NOTE To remove the pf_ring driver, type the following command:

```
#sudo dkms remove –m pf_ring –v 4  --all
```

2.4 Configure PF_RING Aware Driver

```
# mkdir /usr/src/e1000e-pf_ring-1.3.6
# cp -Rf /usr/src/PF_RING_CURRENT_SVN/drivers/PF_RING_aware/intel/e1000e/e1000e-
1.3.6/src/* /usr/src/e1000e-pf_ring-1.3.6/

# cp -f /usr/src/PF_RING_CURRENT_SVN/kernel/linux/pf_ring.h /usr/src/e1000epf_ring-1.3.6/

# cd /usr/src/e1000e-pf_ring-1.3.6/

# sed -i –e
's/\.\.\/\.\.\/\.\.\/\.\.\/\.\.\/\.\.\/kernel\/linux\/pf\_ring\.h/pf\_ring\.h/' netdev.c

#nano dkms.conf
```

```
PACKAGE_NAME="e1000e-pf_ring"
PACKAGE_VERSION="1.3.6"
BUILT_MODULE_NAME[0]="e1000e"
DEST_MODULE_LOCATION[0]="/kernel/drivers/net/e1000e/"
AUTOINSTALL="yes"
```

```
#dkms add -m e1000e-pf_ring -v 1.3.6
```

2.5 Install PF_RING

```
#sudo cp -f /usr/src/PF_RING_CURRENT_SVN/kernel/linux/pf_ring.h
/opt/PF_RING/include/linux/

#cd /usr/src/PF_RING_CURRENT_SVN/userland/lib

#sudo ./configure  --prefix=/opt/PF_RING/

#sudo cp -f pfring_e1000e_dna.h /opt/PF_RING/include

#sudo make

#sudo make install
```

2.6 Install libpcap with PF_RING

```
# cd /usr/src/PF_RING_CURRENT_SVN/userland/libpcap-1.1.1-ring

# sed -i –e 's/\.\.\/lib\/libpfring\.a/\/opt\/PF_RING\/lib\/libpfring\.a/' Makefile.in

#./configure --prefix=/opt/PF_RING

#sudo make

#sudo make install
```

2.7 Install tcpdump with PF_RING

```
.# cd /usr/src/PF_RING_CURRENT_SVN/userland/tcpdump-4.1.1

# sed -i -e 's/\.\.\/lib\/libpfring\.a/\/opt\/PF_RING\/lib\/libpfring\.a/' Makefile.in

# sed -i -e 's/-I \.\.\/libpcap-1\.0\.0-ring/-I \/opt\/PF_RING\/include/' Makefile.in
# sed -i -e 's/-L \.\.\/libpcap-1\.0\.0-ring\/-L /\/opt\/PF_RING\/lib\//' Makefile.in

# ./configure LD_RUN_PATH="/opt/PF_RING/lib:/usr/lib:/usr/local/lib"
--prefix=/opt/PF_RING/ --enable-ipv6

#sudo make

#sudo make install
```

3. Suricata Configuration

3.1 A The Stable version:

```
# cd /opt
# wget http://www.openinfosecfoundation.org/download/suricata-1.0.5.tar.gz
# tar xvfz suricata-1.0.5.tar.gz
# cd suricata-1.0.5
Sudo ./configure --enable-pfring --with-libpfring-libraries=/opt/PF_RING/lib
--with-libpfring-includes=/opt/PF_RING/include --with-
libpcaplibraries=/opt/PF_RING/lib --with-libpcap-includes=/opt/PF_RING/include
LD_RUN_PATH="/opt/PF_RING/lib:/usr/lib:/usr/local/lib"                 --
prefix=/opt/PF_RING/ --enable-nfqueue --with-libcap_ng-libraries=/usr/lib -–with-
libhtp-libraries
# sudo  make
# sudo make install
```

3.1B Beta Version

```
cd /usr/src/PF_RING_CURRENT_SVN/userland/
sudo git clone git://phalanx.openinfosecfoundation.org/oisf.git oisfnew
cd oisfnew
sudo ./autogen.sh
sudo ./configure --enable-pfring --with-libpfring-libraries=/opt/PF_RING/lib --
with-libpfring-includes=/opt/PF_RING/include --with-libpcap-
libraries=/opt/PF_RING/lib --with-libpcap-includes=/opt/PF_RING/include
LD_RUN_PATH="/opt/PF_RING/lib:/usr/lib:/usr/local/lib" --prefix=/opt/PF_RING/ --
enable-nfqueue --with-libcap_ng-libraries=/usr/lib -–with-libhtp-libraries
# sudo  make
# sudo make install
```

3.2 Configure Suricata

```
#mkdir  /etc/suricata
#mkdir  /var/log/suricata
```

3.2A Stable Version:

```
#sudo cp /opt/suricata/suricata.yaml classification.conf /etc/suricata
```

3.2B Beta Version:

```
#sudo cp /usr/src/PF_RING_CURRENT_SVN/userland/lib/oisfnew suricata.yaml
classification.config /etc/surciata
```

3.3 Adding Rules to Suricata

See Suricata Wiki:
https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Rule_Management_with_Oinkmaster

3.4.Run Suricata:

```
#sudo /opt/PF_RING/bin/suricata  –c  /etc/suricata/suricata.yaml –i eth0
```

DEBIAN: Getting Started with Suricata

INTRO:

This is a guide to install Suricata with PF_RING. This installation was completed using Virtualbox version 4.0.6 and Debian iso 6.0.3-i386.

Sections:

1. Basic Suricata installation and how to enable some features
2. PF_RING (capture accelerator) Suricata installation with features from Section 1
3. Suricata Configuration

Required Packages To Build Suricata:

```
sudo apt-get -y install libpcre3 libpcre3-dbg libpcre3-dev

build-essential autoconf automake libtool libpcap-dev libnet1-dev

libyaml-0-2 libyaml-dev zlib1g zlib1g-dev libcap-ng-dev libcap-ng0
```

Install Suricata as and IDS and IPS system

```
#sudo apt-get -y install libnetfilter-queue-dev libnetfilter-queue1 libnfnetlink-dev
libnfnetlink


# cd /opt

# wget http://www.openinfosecfoundation.org/download/suricata-1.0.5.tar.gz

# tar -xvfz suricata-1.0.5.tar.gz

# cd suricata-1.0.5

#sudo ./configure  --enable-nfqueue

#sudo make

#sudo make install
```

To install enable additional features of Suricata , install the following packages and append the ./configure line as stated:

To enable HTP Library(HTML pre-processor):

```
#apt-get install htp

./configure --with-libhtp-libraries
```

To enable libcap_ng (dropping privileges) :

```
#apt-get install libcap-ng-dev

./configure  --with-libcap_ng-libraries=/usr/lib
```

2. PF_RING installation

This installation will install PF_RING as well as enable the above features.

2.1 Download the required packages:

```
#apt-get install build-essential libpcre3-dev libpcap-dev libnet1-dev libyamldev
libnetfilter-queue-dev zlib1g-dev htp subversion flex bison linux- headers-2.6.32-5-686
dkms libcap-ng-dev
```

Note This installation was done with linux-headers-3.0.0-12 and linux-headers-3.0.0-12-generic. Using linux-headers-2.6.32-5 caused issues while building the e1000e-pf_ring module

2.2 Download PF_RING

```
# cd /usr/src
# svn --force export https://svn.ntop.org/svn/ntop/trunk/PF_RING/ PF_RING_CURRENT_SVN
# mkdir /usr/src/pf_ring-4
# cp -Rf /usr/src/PF_RING_CURRENT_SVN/kernel/* /usr/src/pf_ring-4/
```

2.3 Configure PF_RING Driver

```
# cd /usr/src/pf_ring-4/

# nano dkms.conf
```

```
PACKAGE_NAME="pf_ring"
PACKAGE_VERSION="4"
BUILT_MODULE_NAME[0]="pf_ring"
DEST_MODULE_LOCATION[0]="/kernel/net/pf_ring/"
AUTOINSTALL="yes"
```

```
Then press CRTL+X, Y, enter

# dkms add -m pf_ring -v 4
# dkms build -m pf_ring -v 4
# dkms install -m pf_ring -v 4
```

At the end the install command you should get DKMS: INSTALL Completed

NOTE To remove the pf_ring driver, type the following command:

```
#sudo dkms remove –m pf_ring –v 4  --all
```

2.4 Install PF_RING

```
#sudo cp -f /usr/src/PF_RING_CURRENT_SVN/kernel/linux/pf_ring.h
/opt/PF_RING/include/linux/
```

```
#cd /usr/src/PF_RING_CURRENT_SVN/userland/lib

#sudo ./configure  --prefix=/opt/PF_RING/

#sudo cp -f pfring_e1000e_dna.h /opt/PF_RING/include

#sudo make

#sudo make install
```

3. Suricata Configuration

3.1 A The Stable version:

```
# cd /opt
# wget http://www.openinfosecfoundation.org/download/suricata-1.0.5.tar.gz
# tar xvfz suricata-1.0.5.tar.gz
# cd suricata-1.0.5
Sudo ./configure --enable-pfring --with-libpfring-libraries=/opt/PF_RING/lib
--with-libpfring-includes=/opt/PF_RING/include --with-
libpcaplibraries=/opt/PF_RING/lib --with-libpcap-includes=/opt/PF_RING/include
LD_RUN_PATH="/opt/PF_RING/lib:/usr/lib:/usr/local/lib"                  --
prefix=/opt/PF_RING/ --enable-nfqueue --with-libcap_ng-libraries=/usr/lib -–with-
libhtp-libraries
# sudo  make
# sudo make install
```

3.1B Beta Version

```
cd /usr/src/PF_RING_CURRENT_SVN/userland/
sudo git clone git://phalanx.openinfosecfoundation.org/oisf.git oisfnew
cd oisfnew
sudo ./autogen.sh
sudo ./configure --enable-pfring --with-libpfring-libraries=/opt/PF_RING/lib --
with-libpfring-includes=/opt/PF_RING/include --with-libpcap-
libraries=/opt/PF_RING/lib --with-libpcap-includes=/opt/PF_RING/include
LD_RUN_PATH="/opt/PF_RING/lib:/usr/lib:/usr/local/lib" --prefix=/opt/PF_RING/ --
enable-nfqueue --with-libcap_ng-libraries=/usr/lib -–with-libhtp-libraries
# sudo  make
# sudo make install
```

3.2 Configure Suricata

```
#mkdir  /etc/suricata
#mkdir  /var/log/suricata
```

3.2A Stable Version:

```
#sudo cp /opt/suricata/suricata.yaml classification.conf /etc/suricata
```

3.2B Beta Version:

```
#sudo cp /usr/src/PF_RING_CURRENT_SVN/userland/lib/oisfnew suricata.yaml
classification.config /etc/surciata
```

3.3 Adding Rules to Suricata

See Suricata Wiki:
https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Rule_Management_with_Oinkmaster
3.4.Run Suricata:

```
#sudo /opt/PF_RING/bin/suricata  –c  /etc/suricata/suricata.yaml –i eth0
```

CENTOS 6 Getting Started with Suricata

INTRO:

This is a guide to install Suricata with PF_RING. This installation was completed using Virtualbox version 4.0.6 and CentOS 6.0-i386.

Sections:

1. Basic Suricata installation and how to enable some features
2. PF_RING (capture accelerator) Suricata installation with features from Section 1
3. Suricata Configuration

Required Packages To Build Suricata:

```
sudo yum -y install libpcap libpcap-devel libnet libnet-devel pcre

pcre-devel gcc gcc-c++ automake autoconf libtool make libyaml

libyaml-devel zlib zlib-devel
```

Install Suricata as and IDS and IPS system

Xi386

```
sudo rpm -Uvh
http://rules.emergingthreatspro.com/projects/emergingrepo/i386/libnetfilter_queue-0.0.15-
1.i386.rpm \

http://rules.emergingthreatspro.com/projects/emergingrepo/i386/libnetfilter_queue-devel-
0.0.15-1.i386.rpm \

http://rules.emergingthreatspro.com/projects/emergingrepo/i386/libnfnetlink-0.0.30-
1.i386.rpm \

http://rules.emergingthreatspro.com/projects/emergingrepo/i386/libnfnetlink-devel-0.0.30-
1.i386.rpm
```

X86_64:

```
sudo rpm -Uvh
http://rules.emergingthreatspro.com/projects/emergingrepo/x86_64/libnetfilter_queue-
0.0.15-1.x86_64.rpm \

http://rules.emergingthreatspro.com/projects/emergingrepo/x86_64/libnetfilter_queue-
devel-0.0.15-1.x86_64.rpm \

http://rules.emergingthreatspro.com/projects/emergingrepo/x86_64/libnfnetlink-0.0.30-
1.x86_64.rpm \

http://rules.emergingthreatspro.com/projects/emergingrepo/x86_64/libnfnetlink-devel-
0.0.30-1.x86_64.rpm

# cd /opt

# wget http://www.openinfosecfoundation.org/download/suricata-1.0.5.tar.gz
# tar -xvfz suricata-1.0.5.tar.gz

# cd suricata-1.0.5

#sudo ./configure  --enable-nfqueue

#sudo make

#sudo make install
```

To install enable additional features of Suricata , install the following packages and append the ./configure line as stated:

To Enable HTP Library(HTML pre-processor) feature:

```
#wget http://www.openinfosecfoundation.org/download/libhtp-0.2.3.tar.gz

#tar -xzvf libhtp-0.2.3.tar.gz

#cd libhtp-0.2.3

#./configure

#make

#make install
```

To enable libcap_ng (dropping privileges) feature:

```
#wget http://people.redhat.com/sgrubb/libcap-ng/libcap-ng-0.6.4.tar.gz
#tar -xzvf libcap-ng-0.6.4.tar.gz
#cd libcap-ng-0.6.4
#./configure
#make
#sudo make install
```

2.PF_RING installation

This installation will install PF_RING as well as enable the above features.

2.1 Download the required packages:

```
#apt-get install build-essential libpcre3-dev libpcap-dev libnet1-dev libyamldev
libnetfilter-queue-dev zlib1g-dev htp subversion flex bison kernel-devel dkms nano
```

Note Please you will need to download the correct kernel headers for your specific system

2.2 Download PF_RING

```
# cd /usr/src

# svn --force export https://svn.ntop.org/svn/ntop/trunk/PF_RING/ PF_RING_CURRENT_SVN

# mkdir /usr/src/pf_ring-4

# cp -Rf /usr/src/PF_RING_CURRENT_SVN/kernel/* /usr/src/pf_ring-4/
```

2.3 Configure PF_RING Driver

```
# cd /usr/src/pf_ring-4/
```

```
# nano dkms.conf
```

PACKAGE_NAME="pf_ring"
PACKAGE_VERSION="4"
BUILT_MODULE_NAME[0]="pf_ring"
DEST_MODULE_LOCATION[0]="/kernel/net/pf_ring/"
AUTOINSTALL="yes"

Then press CRTL+X, Y, Enter

```
# dkms add -m pf_ring -v 4

# dkms build -m pf_ring -v 4 –kernelsourcedir /usr/src/kernels/(Insert kernel header)

# dkms install -m pf_ring -v 4
```

At the end the install command you should get DKMS: INSTALL Completed

NOTE To remove the pf_ring driver type the following command:

```
#sudo dkms remove –m pf_ring –v 4  --all
```

2.4 Install PF_RING

```
#sudo cp -f /usr/src/PF_RING_CURRENT_SVN/kernel/linux/pf_ring.h
/opt/PF_RING/include/linux/

#cd /usr/src/PF_RING_CURRENT_SVN/userland/lib

#sudo ./configure  --prefix=/opt/PF_RING/

#sudo cp -f pfring_e1000e_dna.h /opt/PF_RING/include

#sudo make

#sudo make install
```

3. Download and configure Suricata with pf_ring

3.1 A Stable version:

```
# cd /opt

# wget http://www.openinfosecfoundation.org/download/suricata-1.0.5.tar.gz

# tar xvfz suricata-1.0.5.tar.gz

# cd suricata-1.0.5

#Sudo ./configure --enable-pfring --with-libpfring-libraries=/opt/PF_RING/lib
```

```
--with-libpfring-includes=/opt/PF_RING/include                          --with-
libpcaplibraries=/opt/PF_RING/lib                                      --with-libpcap-
includes=/opt/PF_RING/include
LD_RUN_PATH="/opt/PF_RING/lib:/usr/lib:/usr/local/lib"                 --
prefix=/opt/PF_RING/

# sudo  make

# sudo make install
```

3.2 Configure Suricata

```
#mkdir  /etc/suricata

#mkdir  /var/log/suricata
```

3.2A Stable Version:

```
#sudo cp /opt/suricata/suricata.yaml classification.conf /etc/suricata
```

3.3 Adding Rules to Suricata

See Suricata Wiki:https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Rule_Management_with_Oinkmaster

3.4.Run Suricata:

```
#sudo /opt/PF_RING/bin/suricata  –c  /etc/suricata/suricata.yaml –i eth0
```

FreeBSD 8 Getting Started with Suricata

INTRO:

This is a guide to install Suricata. This installation was completed using Virtualbox version 4.0.6 and Freebsd8.

Sections:

1. Basic Suricata installation and how to enable some features
2. PF_RING (capture accelerator) Suricata installation with features from Section 1
3. Suricata Configuration

Required Packages To Build Suricata:

```
sudo yum -y install libpcap libpcap-devel libnet libnet-devel pcre

pcre-devel gcc gcc-c++ automake autoconf libtool make libyaml

libyaml-devel zlib zlib-devel

usr/ports/something/subversion
```

Install Suricata as and IDS and IPS system

Xi386

```
sudo rpm -Uvh
http://rules.emergingthreatspro.com/projects/emergingrepo/i386/libnetfilter_queue
-0.0.15-1.i386.rpm \

http://rules.emergingthreatspro.com/projects/emergingrepo/i386/libnetfilter_queue
-devel-0.0.15-1.i386.rpm \

http://rules.emergingthreatspro.com/projects/emergingrepo/i386/libnfnetlink-
0.0.30-1.i386.rpm \

http://rules.emergingthreatspro.com/projects/emergingrepo/i386/libnfnetlink-
devel-0.0.30-1.i386.rpm
```

X86_64

```
sudo rpm -Uvh
http://rules.emergingthreatspro.com/projects/emergingrepo/x86_64/libnetfilter_que
ue-0.0.15-1.x86_64.rpm \

http://rules.emergingthreatspro.com/projects/emergingrepo/x86_64/libnetfilter_que
ue-devel-0.0.15-1.x86_64.rpm \

http://rules.emergingthreatspro.com/projects/emergingrepo/x86_64/libnfnetlink-
0.0.30-1.x86_64.rpm \
http://rules.emergingthreatspro.com/projects/emergingrepo/x86_64/libnfnetlink-
devel-0.0.30-1.x86_64.rpm

# cd /opt

# wget http://www.openinfosecfoundation.org/download/suricata-1.0.5.tar.gz

# tar -xvfz suricata-1.0.5.tar.gz

# cd suricata-1.0.5

#sudo ./configure  --enable-nfqueue

#sudo make

#sudo make install
```

To install enable additional features of Suricata , install the following packages and append the ./configure line as stated:

To Enable HTP Library(HTML pre-processor) feature:

```
wget http://www.openinfosecfoundation.org/download/libhtp-0.2.3.tar.gz

tar -xzvf libhtp-0.2.3.tar.gz

cd libhtp-0.2.3

./configure

make

make install
```

To enable libcap_ng (dropping privileges) feature:

```
wget http

wget http://people.redhat.com/sgrubb/libcap-ng/libcap-ng-0.6.4.tar.gz

tar -xzvf libcap-ng-0.6.4.tar.gz

cd libcap-ng-0.6.4

./configure

make

sudo make install
```

3. Download and configure Suricata with pf_ring

3. A The Stable version:

```
# cd /opt
# wget http://www.openinfosecfoundation.org/download/suricata-1.0.5.tar.gz
# tar xvfz suricata-1.0.5.tar.gz
# cd suricata-1.0.5
Sudo ./configure --enable-pfring --with-libpfring-libraries=/opt/PF_RING/lib
--with-libpfring-includes=/opt/PF_RING/include --with-libpcaplibraries=/
opt/PF_RING/lib --with-libpcap-includes=/opt/PF_RING/include
LD_RUN_PATH="/opt/PF_RING/lib:/usr/lib:/usr/local/lib" --prefix=/opt/PF_RING/
# sudo  make
# sudo make install
```

3.2 Configure Suricata

```
#mkdir  /etc/suricata
#mkdir /etc/suricata/rules
#mkdir  /var/log/suricata
#cp /opt/suricata/suricata yaml classification.conf /etc/suricata
```

3.3 Adding Rules to Suricata
See Suricata Wiki:

https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Rule_Management_with_Oinkmaster

3.4.Run Suricata:

```
#sudo /opt/PF_RING/bin/suricata  –c  /etc/suricata/suricata.yaml –i em0
```