# ANALISIS PERFORMA MEMORI SERVER MENGGUNAKAN IDS SURICATA

Skripsi

Untuk memenuhi persyaratan mencapai derajat Sarjana S-1

Program Studi Teknik Informatika


Disusun oleh :

Yazid Ubaidilah

NIM. 10651015



PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA

YOGYAKARTA


2014

-

Universitas Islam Negeri Sunan Kalijaga FM-UINSK-BM-05-07/R0

**PENGESAHAN SKRIPSI/TUGAS AKHIR**

Nomor : UIN.02/D.ST/PP.01.1/1825/2014

Skripsi/Tugas Akhir dengan judul : Analisis Performa Memori Server Menggunakan IDS Suricata

Yang dipersiapkan dan disusun oleh :
Nama : Yazid Ubaidilah
NIM : 10651015
Telah dimunaqasyahkan pada : Kamis, 19 Juni 2014
Nilai Munaqasyah : A / B
Dan dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga

**TIM MUNAQASYAH :**

Ketua Sidang

Bambang Sugiantoro, M.T
NIP. 19751024 200912 1 002

Penguji I

Sumarsono, M.Kom
NIP.19710209 200501 1 003

Penguji II

Agus Mulyanto, M.Kom
NIP. 19710823 199903 1 003

Yogyakarta, 24 Juni 2014
UIN Sunan Kalijaga
Fakultas Sains dan Teknologi
Dekan

Prof. Dr. H. Akh. Minhaji, M.A, Ph.D
NIP. 19580919 198603 1 002

Universitas Islam Negeri Sunan Kalijaga FM-UINSK-BM-05-03/R0

**SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR**

Hal : Permohonan
Lamp : -

Kepada
Yth. Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
di Yogyakarta

*Assalamu'alaikum wr. wb.*

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka kami selaku pembimbing berpendapat bahwa skripsi Saudara:

Nama : Yazid Ubaidilah
NIM : 10651015
Judul Skripsi : Analisis Performa Memori Server Menggunakan IDS Suricata

sudah dapat diajukan kembali kepada Program Studi Tekni Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Teknik Informatika

Dengan ini kami mengharap agar skripsi/tugas akhir Saudara tersebut di atas dapat segera dimunaqosyahkan. Atas perhatiannya kami ucapkan terima kasih.

*Wassalamu'alaikum wr. wb.*

Yogyakarta, 16 Juni 2014
Pembimbing

Bambang Sugiantoro, M.T.
NIP: 19751024 200912 1 002

iv

## PERNYATAAN KEASLIAN SKRIPSI

Yang bertanda tangan di bawah ini :

| | |
|---|---|
| Nama | : Yazid Ubaidilah |
| Nim | : 10651015 |
| Program Studi | : Teknik Informatika |
| Fakultas | : Sains dan Teknologi |

Menyatakan bahwa skripsi dengan judul **Analisis Performa Memori Server Menggunakan IDS Suricata** tidak terdapat pada karya yang pernah diajukan untuk memperoleh gelar sarjana di suatu Perguruan Tinggi, dan sepengetahuan saya tidak terdapat karya atau pendapat yang pernah ditulis oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 16 Juni 2014

Yang Menyatakan,


Yazid Ubaidilah
NIM : 10651015

# KATA PENGANTAR

Alhamdulillah wa syukurillah penulis panjatkan sebesar-besarnya tiada henti ke pangkuan Alloh SWT yang telah melimpahkan rahmat, nikmat dan karunia-Nya kepada kita, sehingga penulis dapat menyelesaikan skripsi yang berjudul "*Analisis Performa Memori Server Menggunakan IDS Suricata"* dengan baik. Tak lupa shalawat dan salam senantiasa tercurah kepada junjungan agung Rasulullah SAW yang telah menunjukkan jalan terbaik kepada kita.

Skripsi ini disusun untuk memenuhi sebagian persyaratan mendapatkan gelar kesarjanaan pada Program Studi Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta.

Dalam kesempatan ini, penulis ingin mengucapkan banyak terima kasih kepada :

1. Bapak Prof. Dr. H. Musa Asy'arie, M.A., selaku Rektor UIN Sunan Kalijaga Yogyakarta.
2. Bapak Prof. Drs. H. Akh. Minhaji, M.A., Ph.D., selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
3. Bapak Agus Mulyanto, M.Kom., selaku Ketua Program Studi Teknik Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sekaligus sebagai Penguji II skripsi saya.
4. Bapak Bambang Sugiantoro, M.T., pembimbing dalam menyelesaikan skripsi ini.
5. Bapak Sumarsono, S.T., M.Kom., selaku Penguji I skripsi saya.
6. Bapak Mustakim, MT., selaku Dosen Pembimbing Akademik Kelas K mandiri Teknik Informatika.
7. Bapak dan Ibu Dosen Teknik Informatika UIN Sunan Kalijaga Yogyakarta yang telah banyak berbagi ilmu dan pengalamannya kepada penulis.

8. Ayah dan ibu tercinta yang senantiasa mensupport penulis dengan semua kasih dan sayangnya.
9. Teman-teman Teknik Informatika yang telah memberikan semangat dan pengalamannya kepada penulis.
10. Teman-teman KKN Cokrodirjan.
11. Dan semua pihak yang tidak penulis sebutkan satu per satu.

Penulis menyadari bahwa dalam penyelesaian skripsi ini masih jauh dari kata sempurna, Oleh karena itu kritik dan saran yang bersifat membangun sangat penulis harapkan dari semua pihak demi kesempurnaan di masa mendatang. Semoga skripsi ini bermanfaat bagi pembaca dan penulis khususnya.

Yogyakarta, 29 Mei 2014

Penyusun,

Yazid Ubaidilah

NIM. 10651015

# MOTTO

**“**Talk is cheap, show me the code.(Linus Torvald)**”**

**“**The quieter you become, the more you are able to hear. (Kali Linux)**”**

## PERSEMBAHAN

Alhamdulillah segala doa dan syukur tiada henti terucap ke hadirat Allah SWT, Tuhan seru sekalian alam. Shalawat dan salam teriring kepada junjungan Nabi Agung Muhammad SAW beserta keluarganya semoga senantiasa kita menjadi ummat yang bertaqwa. Saya persembahkan kepada orang-orang yang telah membantu saya dalam menyelesaikan skripsi ini baik berupa dukungan moral dan spiritual.

- ✓ Bapak (Alm.) dan ibu terkasih yang selalu memberikan ananda yang terbaik dalam menjalani setiap liku kehidupan.
- ✓ Paman Mudzakir sekeluarga yang telah banyak membantu mengurusi kuliahku.
- ✓ Mbak Yati sekeluarga yang telah memberi support sampai aku bisa kuliah.
- ✓ Mbak Iqoh sekeluarga yang sudah bersedia memberi aku dukungan tempat tinggal.
- ✓ Eti, ini adalah penyemangat buatmu untuk bisa lebih baik dariku.
- ✓ Najwaku, Deta Oktavia yang senantiasa mengajariku kesetiaan dan kesabaran di saat suka dan duka.
- ✓ Mbah Dede, tempat menumpang paling enak mencari ide.
- ✓ Pagar Nusa, terima kasih sudah menempa pribadiku menjadi lebih baik lagi.
- ✓ Indonesian Backtrack Team, forum komunikasi pentester yang menjadi tempat belajar penulis menyelesaikan skripsi ini.
- ✓ Ubuntu Indonesia Forum, tempat bertapa penulis menyelesaikan skripsi ini.
- ✓ Nadzif terima kasih sudah mengizinkan penulis riset dengan mikrotiknya.

- ✓ Opang, terima kasih sudah membantuku otak-atik mikrotik.
- ✓ Penghuni kost Sugeng (Aris, Achyar, Arya, Piteng, Fajar, Fanni) terima kasih sudah bikin kos layaknya rumah sendiri.

# *ANALISIS PERFORMA MEMORI SERVER MENGGUNAKAN IDS SURICATA*

**Yazid Ubaidilah**

**NIM. 10651015**

## INTISARI

Komputer server adalah tempat di mana sebuah komputer bertindak dalam melayani permintaan atau pengaksesan data maupun proses kerja dari komputer lain melalui jaringan komunikasi.

Salah satu kejahatan komputer server ialah DOS attack yaitu menjadikan sebuah server melayani banyak klien dalam satu waktu. Hal ini yang menyebabkan penggunaan bandwidth dan memori komputer cepat terkuras habis. Sehingga ketika ada klien atau user lain yang berusaha mengakses server tersebut tidak bisa menerima layanan server dikarenakan server telah down.

Setelah diadakan analisis lebih mendalam dilihat dari rangkaian analisis deskriptif pre-test dan post-test dapat disimpulkan bahwa Suricata IDS mampu mengurangi serangan DOS attack dilihat dari log http, fast dan stat yang membuktikan adanya penyusup dengan serangan IP Flooding. Dari log http header, informasi yang berhasil dikumpulkan adalah penyusup menggunakan aplikasi Siege melakukan penyerangan dari IP address 192.168.1.2 miliknya ke IP address server yaitu 192.168.1.1. Sedangkan dari log fast menampilkan data serangan berupa TCP invalid checksum dari IP address penyerang. Untuk log stat berisi keadaan trafik jaringan yang sedang berlangsung dengan lengkap.

Kata Kunci: *Suricata, IDS, penggunaan memori, DOS Attack*

***SERVER MEMORY PERFORMANCE ANALYZE USING IDS SURICATA***

**YAZID UBAIDILAH**

**NIM. 10651015**

## ABSTRACT

We often see the utilization of computers in almost every aspect of life. But over time the security aspects in the exchange of information and data to be ignored even become mandatory aspect to make the exchange of information and data to be safe from people who are not interested. To answer these challenges, Suricata comes as one solution to reducing crime in computer security that make it as an alarm when the computer server where data and information are under attack.

One of the computer crime is a DOS attack is to make a server serving multiple clients at one time. It's led to the use of bandwidth and computer memory quickly drained away. So when there is a client or other user who attempts to access the server can't receive the service of the server because the server was down.

Having conducted more in-depth a series of views of the descriptive analysis of pre-test and post-test can be concluded that the Suricata IDS is able to reduce DOS attack based on http.log, fast.log and stat.log that proved attacker with IP Flooding attack. From http header log the information that could be concluded is attacker uses Siege application doing attacking from IP address 192.168.1.2 to IP address server exactly 192.168.1.1. Fast log shows attacking data absolutely TCPv4 invalid checksum. Stat log can shows the network traffic completely.

Keywords: Suricata, IDS, memory usage, DOS Attack

# DAFTAR ISI

**HALAMAN JUDUL ..... i**

**HALAMAN PENGESAHAN ..... ii**

**HALAMAN PERSETUJUAN ..... iii**

**HALAMAN PERNYATAAN ..... iv**

**KATA PENGANTAR ..... v**

**MOTTO ..... vii**

**PERSEMBAHAN ..... viii**

**INTISARI ..... x**

**ABSTRACT ..... xi**

**DAFTAR ISI ..... xii**

**DAFTAR TABEL ..... xv**

**DAFTAR GAMBAR ..... xvi**

**DAFTAR LAMPIRAN ..... xvii**

**BAB I PENDAHULUAN ..... 1**

1.1. Latar Belakang ..... 1
1.2. Rumusan Masalah ..... 2
1.3. Batasan Masalah ..... 3
1.4. Tujuan Penelitian ..... 3
1.5. Manfaat Penelitian ..... 3
1.6. Keaslian Penelitian ..... 4

**BAB II TINJAUAN PUSTAKA DAN LANDASAN TEORI ..... 5**

2.1. Tinjauan pustaka ..... 5

2.2. Landasan Teori ..... 9

2.2.1. Jaringan Komputer ..... 9

2.2.1.1. Client-Server ..... 11

2.2.1.2. Peer to peer ..... 12
2.2.1.3. LAN ..... 13
2.2.1.4. MAN ..... 16
2.2.1.5. WAN ..... 16
2.2.1.6. Bentuk ancaman pada jaringan komputer ..... 17
2.2.2. Keamanan Jaringan ..... 18
2.2.3. IDS ..... 25
2.2.4. Suricata IDS ..... 36
2.2.5. Linux ..... 37
2.2.6. Web Server ..... 37
2.2.7. Siege ..... 38
2.2.8. PSPP ..... 39
2.2.9. Penelitian Kuantitatif ..... 39
2.2.10. Signifikansi ..... 41
**BAB III METODE PENELITIAN ..... 43**
3.1. Objek Penelitian ..... 43
3.2. Pengumpulan Data ..... 43
3.3. Teknik Pengolahan Data ..... 44
3.3.1. Analisis Deskriptif ..... 44
3.3.1.1. Analisis Inferensi ..... 44
3.4. Pembuatan Sistem ..... 47
3.5. Metode Penelitian ..... 48
3.6. Alur Penelitian ..... 50
**BAB IV HASIL DAN PEMBAHASAN ..... 56**
4.1. Hasil dan Pembahasan ..... 56

4.1.1. Analisis Deskriptif Data Pre-test dan Post-test ..... 60
4.1.1.1. Uji Normalitas Data Pre-test ..... 61
4.1.1.2. Uji Normalitas Data Post-test..... 63
4.1.1.3. Uji Homogenitas ..... 64
4.1.1.4. Uji Korelasi ..... 66
4.1.1.5. Uji T Dua Sampel Dependen ..... 68
4.1.1.6. Data Pengujian Suricata ..... 71
**BAB V PENUTUP..... 72**
5.1. Kesimpulan ..... 72
5.2. Saran..... 72
**DAFTAR PUSTAKA..... 73**
**LAMPIRAN..... 75**

# DAFTAR TABEL

Tabel 2.1 Penelitian Terdahulu .....7

Tabel 3.1 Spesifikasi Infrastruktur Sistem .....48

Tabel 3.2 Desain Penelitian.....49

Tabel 4.1 Perbandingan hasil pengujian pre-test .....58

Tabel 4.2 Perbandingan hasil pengujian post-test.....60

Tabel 4.3 Perbandingan memori .....61

Tabel 4.4 Data Penggunaan Memori Pre-test dan Post-test.....68

Tabel 4.5 Data keseluruhan penggunaan memori .....70

Tabel 4.6 Deteksi serangan..... 71

# DAFTAR GAMBAR

Gambar 2.1 Client-Server .....12

Gambar 2.2 Model Peer to Peer .....13

Gambar 2.3 Wifi dan Ethernet .....15

Gambar 2.4 MAN.....16

Gambar 2.5 WAN .....17

Gambar 2.6 Komponen IDS.....27

Gambar 3.1 Alur Penelitian.....51

Gambar 3.2 Arsitektur Penelitian.....52

Gambar 4.1 Uji normalitas pre-test.....62

Gambar 4.2 Uji normalitas post-test .....64

Gambar 4.3 Uji homogenitas pre-test .....65

Gambar 4.4 Uji homogenitas post-test.....66

Gambar 4.5 Uji korelasi Pre-test.....67

Gambar 4.6 Uji Korelasi post-test.....67

Gambar 4.7 Uji T Sampel Dependen data penelitian…… 69

Gambar 4.8 Output http.log ..... 72

Gambar 4.9 Output fast.log ..... 72

# DAFTAR LAMPIRAN

Gambar pengujian pertama pre-test ..... 75
Gambar pengujian kedua pre-test..... 75
Gambar pengujian ketiga pre-test ..... 76
Gambar pengujian ke-empat pre-test ..... 76
Gambar pengujian kelima pre-test ..... 77
Gambar pengujian pertama post-test..... 77
Gambar pengujian kedua post-test..... 78
Gambar pengujian ketiga post-test..... 78
Gambar pengujian ke-empat post-test..... 79
Gambar pengujian kelima post-test..... 79
File konfigurasi suricata.yaml..... 80
File konfigurasi classification.config..... 111
File konfigurasi reference.config ..... 113
File konfigurasi pada Siege..... 114
Hasil data Suricata pada post-test pertama..... 129
Hasil data Suricata pada post-test kedua..... 132
Hasil data Suricata pada post-test ketiga..... 135
Hasil data Suricata pada post-test ke-empat..... 138
Hasil data Suricata pada post-test kelima..... 141
Instalasi Suricata ..... 143
Perintah pengujian pertama pre-test..... 145
Perintah pengujian kedua pre-test ..... 146
Perintah pengujian ketiga pre-test..... 146

Perintah pengujian ke-empat pre-test ..... 147
Perintah pengujian kelima pre-test ..... 147
Perintah pengujian pertama post-test ..... 148
Perintah pengujian kedua post-test ..... 148
Perintah pengujian ketiga post-test ..... 149
Perintah pengujian keempat post-test ..... 149
Perintah pengujian kelima post-test ..... 150

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang Masalah

Penggunaan media telekomunikasi memang memudahkan pekerjaan kita semua. Namun dari segudang kegunaannya tersimpan ancaman, gangguan dan dampak buruk yang terkadang tidak terpikirkan ketika kita menggunakan telekomunikasi. Teknologi telekomunikasi yang bisa diakses kapan saja dan di mana saja membuat pertukaran data dan informasi begitu mudahnya dilakukan oleh siapa saja. Bahkan tidak sedikit orang melakukan pencurian data dan informasi demi keuntungannya sendiri. Untuk mencegah terjadinya pengaksesan oleh orang yang tidak mempunyai wewenang sistem dapat diperkuat dari sisi keamanannya.

Keamanan jaringan tergantung pada kecepatan pengatur jaringan dalam menindaklanjuti sistem saat terjadi gangguan. Untuk memperkuat keamanan jaringan komputer dapat diterapkan sistem pendeteksi serangan dalam jaringan tersebut.

Server sebagai sarana vital untuk menyimpan database, aplikasi dan layanan penting sangat diperlukan sisi keamanannya. Baik dari segi infrastruktur sendiri maupun dari aplikasi pendukungnya. Diharapkan server terhindar dari hal-

hal yang mengganggu kinerjanya sehingga pelayanan terhadap klien berfungsi secara maksimal.

IDS (Intrusion Detection System) membantu administrator jaringan dalam memantau keadaan sistem dengan mendeteksi dan menganalisa lalu lintas paket-paket data yang terjadi pada jaringan. Suricata adalah salah satu IDS engine open source yang dirilis oleh OISF (Open Information System Foundation) organisasi non-profit yang didanai oleh pemerintahan Amerika Serikat.

Dalam pengamatan penulis IDS snort paling banyak digunakan karena snort merupakan *standard de facto* IDS di dunia. Akan tetapi kemunculan Suricata sebagai IDS belum banyak dilakukan riset dalam dunia pendidikan yaitu lingkup skripsi. Oleh karena itu, penulis mencoba melakukan riset kecil tentang Suricata IDS. Di mana penulis menitikberatkan pada penelitian bagaimana performa server sebelum dan sesudah adanya Suricata. Dalam hal ini pemakaian memori komputer yang menjadi tolok ukur.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, penulis mengambil rumusan masalah bagaimana kinerja server sebelum dan sesudah adanya Suricata IDS dalam menangani IP Flooding/DOS attack yang menguras memori komputer.

## 1.3 Batasan Masalah

Rumusan masalah pada penelitian ini akan dibatasi oleh beberapa hal sebagai berikut :

1. Pengetesan Suricata IDS hanya test performance engine.
2. Pola serangan yang diberikan IP flooding.
3. Suricata IDS diinstall di OS Ubuntu 12.04 LTS.
4. Analisis bersifat analisis kuantitatif.
5. Log output dari Suricata berupa http, fast dan stat.

## 1.4 Tujuan Penelitian

Adapun tujuan dari penelitian Suricata IDS ini ialah menguji kinerja Suricata IDS sebagai engine IDS yang baru dalam mengatasi serangan DOS attack.

## 1.5 Manfaat Penelitian

Dari hasil penelitian ini diharapkan :

1. Dapat mengetahui kehandalan Suricata IDS sebagai pertahanan sistem.
2. Dapat memahami bahaya yang mengancam sistem dengan memantau aktivitas jaringan.

## 1.6 Keaslian Penelitian

Penelitian tentang IDS banyak menggunakan Snort sebagai engine akan tetapi Suricata IDS baru dilakukan penelitian tentang *test performance* itupun baru segelintir penelitian. Sedangkan Analisis Suricata IDS dengan pola serangan

tertentu di UIN Sunan Kalijaga Yogyakarta sepengetahuan penulis belum pernah dilakukan.

# BAB IV

# HASIL DAN PEMBAHASAN

Bab ini akan membahas mengenai analisis data dari hasil pengolahan data penelitian yang telah dilakukan. Hasil analisis data yang diperoleh merupakan gambaran dari hasil seluruh kegiatan penelitian. Data yang ada merupakan data kualitatif hasil tes baik pre-test (sebelum Suricata) maupun post-test (sesudah Suricata). Pengolahan data menggunakan software PSPP.

## 4.1 Hasil dan Pembahasan

Hasil dari penelitian ini dibagi menjadi dua hasil yaitu pre-test dan post-test. Hasil pre-test adalah dimana komputer diserang sebelum menggunakan Suricata IDS. Sedangkan hasil post-test adalah komputer diserang setelah diinstall Suricata IDS. Setiap kali dilakukan penyerangan, diukur penggunaan memori komputer. Pengukuran tersebut menggunakan aplikasi top bawaan sistem operasi Linux.

Sebelum dilakukan pengujian, komputer server menunjukkan penggunaan memori standar yaitu 585.396 KB dengan task 172 buah. Percobaan pertama dalam waktu satu menit menghasilkan output top sebagai berikut dapat kita lihat di lampiran bahwa memori komputer yang digunakan ialah 770.708 KB. Dengan kata lain pengujian pertama memakai

40,01 % dari total RAM. Adapun task yang sedang berjalan berjumlah 176 buah.

Selanjutnya setelah diberi waktu jeda selama satu menit, dilanjutkan dengan pengujian kedua. Pengujian kedua rentang waktu yang diberikan selama dua menit. Hasil dari pengujian kedua dapat kita lihat memori yang dibutuhkan ialah 768.208 KB atau 39,8 % dari RAM dengan total task yang sedang berjalan berjumlah 176.

Pengujian ketiga dilakukan setelah jeda waktu satu menit dari pengujian sebelumnya. Lama waktu untuk melakukan pengujian ini selama tiga menit. Memori yang digunakan dalam pengujian ketiga adalah 43,7 % dari RAM komputer yaitu sebesar 841.900 KB dengan menjalankan task sebanyak 178 buah.

Selanjutnya dilakukan pengujian ke-empat dengan lama waktu pengujian empat menit setelah sebelumnya dijeda satu menit dari pengujian ketiga. Setelah diuji selama empat menit ternyata komputer yang dijadikan IDS membutuhkan memori 44,1 % atau sekitar 849.700 KB dengan 178 task yang ada.

Pengujian terakhir yaitu pengujian selama lima menit dengan jeda waktu satu menit dari pengujian sebelumnya. Dengan lama waktu pengujian lima menit ternyata komputer IDS menjalankan task 178 dengan konsumsi memori 44,2 % yaitu membutuhkan memori 851.848 KB.

Untuk lebih jelasnya dapat kita lihat tabel perbandingan komsumsi memori sebelum diinstall Suricata di bawah ini :

Tabel 4.1 Perbandingan hasil pengujian pre-test

| Menit | Memori | Prosentase | Task |
|---|---|---|---|
| 1 | 770708 | 40,0 | 176 |
| 2 | 768208 | 39,9 | 176 |
| 3 | 841900 | 43,7 | 178 |
| 4 | 849700 | 44,1 | 178 |
| 5 | 851848 | 44,2 | 178 |

Sebelum pengujian ini dilakukan pantauan memori. Memori yang digunakan setelah adanya Suricata IDS namun belum diadakan pengujian menunjukkan angka 1.171.784 KB dengan total task 181 buah. Pengujian tahap kedua yaitu post-test adalah pengujian setelah komputer server diinstall suricata IDS. Sebelum diuji, jangan lupa untuk mengaktifkan Suricata IDS dengan perintah *sudo /usr/bin/suricata -c /etc/suricata/suricata.yaml -i eth1*. Jadi dalam hal ini Suricata aktif dan secara otomatis penggunaan memori komputer juga bertambah banyak daripada sebelum diinstall.

Pengujian post-test pertama diambil setelah pengujian tahap pertama selesai dilakukan. Akan tetapi komputer server di *reboot* terlebih dahulu agar seperti kondisi baru dinyalakan.

Pengujian kali ini dimulai dari penggunaan komsumsi memori lebih besar dari tahap pertama yaitu 1.182.480 KB yaitu 61,4 % dari total RAM yang ada, dengan penggunaan task sebesar 181 buah.

Seperti tahap pertama, jeda yang diberikan untuk pengujian selanjutnya juga selama satu menit, dan lama penyerangan dua menit. Dapat kita lihat bahwa dengan menjalankan task yang sama pengujian sebelumnya yaitu 181 buah, komputer IDS mengalami peningkatan penggunaan memori menjadi 61,9 %. Dengan kata lain komputer membutuhkan memori komputer sebesar 1.193.232 KB.

Pengujian ketiga dengan tenggang waktu selama satu menit juga sama seperti yang lain membuat penyerangan DOS ke komputer server selama tiga menit. Hasil dari pengujian ketiga dapat dilihat bahwa task 181 sama seperti pengujian kedua membutuhkan memori 1.197.644 KB atau lebih ringkasnya 62,2 % dari total memori komputer.

Pengujian ke-empat dapat dilihat hasilnya setelah dijeda waktu satu menit dari pengujian sebelumnya diberi waktu empat menit untuk penyerangan. Pengujian ke-empat menunjukkan konsumsi memori yang lebih banyak dari pengujian sebelumnya yaitu pengujian ketiga dengan menjalankan task 184 buah dibutuhkan memori 62,4 % atau 1.202.396 KB.

Pengujian post-test terakhir yaitu pengujian kelima dengan waktu jeda satu menit dari pengujian ke-empat dan pengujian selama lima menit

dapat melaksanakan task sebanyak 184 buah dengan total memori yang digunakan berkisar 62,7 % atau 1.207.644 KB.

Agar lebih mudah membedakan hasil pengujian tahap kedua (post-test) berikut rangkuman pengujiannya :

Tabel 4.2 Tabel hasil pengujian post-test

| Menit | Mem Used | Percentage | Task |
|---|---|---|---|
| 1 | 1182480 | 61,4 | 181 |
| 2 | 1193232 | 61,9 | 181 |
| 3 | 1197644 | 62,2 | 181 |
| 4 | 1202396 | 62,4 | 184 |
| 5 | 1207644 | 62,7 | 184 |

Secara keseluruhan hasil pengujian Suricata menggunakan serangan IP Flooding yang difokuskan terhadap penggunaan memori komputer dapat dilihat di tabel 4.3.

### 4.1.1 Analisis Deskriptif Data Pre-test dan Post-test

Analisis deskriptif bertujuan untuk memperoleh gambaran umum tentang data hasil pengujian serangan pre-test dan post-test komputer server yang didapatkan hasilnya berupa rata-rata dan standar deviasi.

Selanjutnya untuk mengetahui perbedaan kemampuan komputer server dalam menghadapi treatment berupa serangan DOS dari attacker dengan pembeda sebelum dan sesudah diinstall Suricata IDS dapat dilihat dari hasil analisis gain. Berdasarkan pengolahan data memori komputer server, gambaran umum tersebut tersaji pada tabel di bawah ini. Nilai besarnya memori pada pre-test minimum 746.780 sedangkan nilai maksimumnya adalah 760.140 dengan nilai rata-rata penggunaan memori 750.546,40 dengan standar deviasi 5.591,79. Untuk post-test memori minimum yang digunakan ialah 1.052.124 sampai nilai maksimum ialah 1.061.488 dengan standar deviasi lebih kecil daripada pre-test yaitu hanya mencapai 4.622,6 dengan nilai rata-rata pemakaian memori mencapai 1.056.064 .

Tabel 4.3 Perbandingan memori

| Pre-test | | | | Post-test | | | |
|---|---|---|---|---|---|---|---|
| Min | Max | Mean | SD | Min | Max | Mean | SD |
| 768208 | 851848 | 816472,80 | 43086,84 | 1182480 | 1207644 | 1196679,20 | 9583,06 |

#### 4.1.1.1 Uji normalitas Data Pre-test

Langkah awal dalam menguji hasil percobaan serangan adalah dengan menguji normalitas data pre-test server, apakah data yang dihasilkan itu berdistribusi normal atau tidak. Untuk menguji normalitas data pre-test, dapat digunakan uji statistik *Kolmogorov-Smirnov*. Dengan rumus hipotesis pengujian normalitas data sebagai berikut :

$H_0$ : nilai pre-test berasal dari data yang berdistribusi normal.

$H_1$: nilai pre-test berasal dari data yang berdistribusi tidak normal.

Dengan taraf signifikansi 0,05. Sedangkan kriteria pengujiannya sebagai berikut :

1. Jika nilai signifikansi lebih dari atau sama dengan 0,05 maka $H_0$ diterima.
2. Jika nilai signifikansi kurang dari 0,05 maka $H_0$ ditolak.

Adapun hasil dari uji normalitas pre-test dengan metode *Kolmogorov-Smirnov* dengan tool PSPP dapat dilihat di bawah ini.

One-Sample Kolmogorov-Smirnov Test

| | | Memori | Task | Menit |
|---|---|---|---|---|
| N | | 5 | 5 | 5 |
| Normal Parameters | Mean | 816472.80 | 177.20 | 3.00 |
| | Std. Deviation | 43086.84 | 1.10 | 1.58 |
| Most Extreme Differences | Absolute | .32 | .37 | .14 |
| | Positive | .26 | .26 | .14 |
| | Negative | -.32 | -.37 | -.14 |
| Kolmogorov-Smirnov Z | | .72 | .82 | .31 |
| Asymp. Sig. (2-tailed) | | .68 | .51 | 1.00 |

Gambar 4.1 Uji normalitas pre-test

Dari gambar tersebut tingkat signifikansi data pre-test mulai dari variabel menit, memori dan task masing-masing bernilai 1,00; 0,68; dan 0,51 yang

berarti melebihi nilai signifikansi sebesar 0,05. Hal ini membuktikan bahwa distribusi data pre-test termasuk distribusi data yang normal.

#### 4.1.1.2 Uji Normalitas Data Post-test

Setelah data pre-test kita uji normalitasnya, selanjutnya diuji pula normalitas data post-test yang telah ada. Data post-test juga diuji dengan uji statistik *Kolmogorov-Smirnov* dengan ketentuan hipotesis pengujian normalitas data sebagai berikut :

$H_0$ : nilai post-test berasal dari data yang berdistribusi normal.

$H_1$: nilai post-test berasal dari data yang berdistribusi tidak normal.

Dengan taraf signifikansi 0,05. Kriteria pengujian data post-test ditetapkan sebagai berikut :

1. Jika nilai signifikansi lebih dari atau sama dengan 0,05 maka $H_0$ diterima.
2. Jika nilai signifikansi kurang dari 0,05 maka $H_0$ ditolak.

Berikut hasil uji normalitas data post-test yang telah dilakukan :

One-Sample Kolmogorov-Smirnov Test

| | | Memori | Task | Menit |
|---|---|---|---|---|
| N | | 5 | 5 | 5 |
| Normal Parameters | Mean | 1196679.20 | 182.20 | 3.00 |
| | Std. Deviation | 9583.06 | 1.64 | 1.58 |
| Most Extreme Differences | Absolute | .16 | .37 | .14 |
| | Positive | .13 | .37 | .14 |
| | Negative | -.16 | -.26 | -.14 |
| Kolmogorov-Smirnov Z | | .36 | .82 | .31 |
| Asymp. Sig. (2-tailed) | | 1.00 | .51 | 1.00 |

Gambar 4.2 Uji normalitas Post-test

Dapat dilihat dari gambar di atas taraf signifikansi dari variabel menit senilai 1,00, variabel memori senilai 1,00 dan variabel task senilai 0,51 yang berarti telah melebihi taraf signifikansi yang telah ditetapkan yaitu sebesar 0,05. Hal ini membuktikan bahwa $H_0$ diterima dan data post-test memang termasuk distribusi data yang normal pula.

### 4.1.1.3. Uji Homogenitas

Setelah diketahui bahwa data dari pre-test maupun post-test adalah data yang berdistribusi normal, langkah berikutnya ialah uji homogenitas yang dimaksudkan untuk memperlihatkan bahwa dua atau lebih kelompok data sampel berasal dari populasi yang memiliki variansi yang sama.

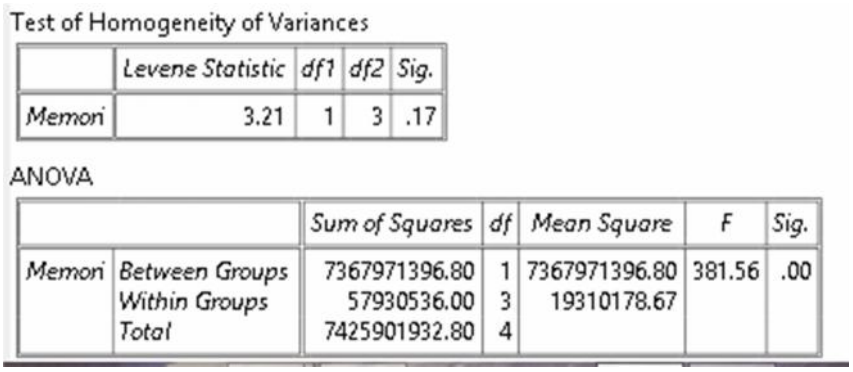
Uji homogenitas menggunakan *Lavene Test dengan tingkat signifikansi 5 %*. Ketentuan dalam uji homogenitas :

$H_0$ : varians pada tiap kelompok sama (homogen)

$H_1$ : varians pada tiap kelompok berbeda (heterogen)

ketentuan kriteria uji homogenitas ialah jika signifikansi yang diperoleh melebihi α, maka variansi setiap sampel homogen. Namun apabila signifikansi yang diperoleh < α, maka variansi setiap sampel bersifat heterogen.

Berikut hasil uji homogenitas data pre-test yaitu data hasil pengujian sebelum adanya instalasi Suricata IDS :



Test of Homogeneity of Variances

| | Levene Statistic | df1 | df2 | Sig. |
|---|---|---|---|---|
| Memori | 3.21 | 1 | 3 | .17 |

ANOVA

| | | Sum of Squares | df | Mean Square | F | Sig. |
|---|---|---|---|---|---|---|
| Memori | Between Groups | 7367971396.80 | 1 | 7367971396.80 | 381.56 | .00 |
| | Within Groups | 57930536.00 | 3 | 19310178.67 | | |
| | Total | 7425901932.80 | 4 | | | |

Gambar 4.3 Uji homogenitas pre-test

Terlihat dari hasil pengolahan data melalui software aplikasi PSPP menunjukkan bahwa data tersebut berasal dari variansi yang homogen. Hasil uji homogenitas mencapai nilai signifikansi 0,17 melebihi 0,05 yang berarti $H_0$ diterima.

Sedangkan hasil uji homogenitas data setelah diinstall Suricata dapat dilihat di gambar 4.4 di bawah ini :

Test of Homogeneity of Variances

| | Levene Statistic | df1 | df2 | Sig. |
|---|---|---|---|---|
| Memori | 1.60 | 1 | 3 | .30 |

ANOVA

| | | Sum of Squares | df | Mean Square | F | Sig. |
|---|---|---|---|---|---|---|
| Memori | Between Groups | 231896482.13 | 1 | 231896482.13 | 5.14 | .11 |
| | Within Groups | 135443466.67 | 3 | 45147822.22 | | |
| | Total | 367339948.80 | 4 | | | |

Gambar 4.4 Uji homogenitas post-test

Uji homogenitas pada data hasil pengujian setelah diinstall Suricata memperlihatkan angka nilai signifikansi sebesar 0,30 yang berarti bahwa data-data tersebut merupakan data yang mempunyai variansi yang sama.

#### 4.1.1.4 Uji Korelasi

Pengujian ada tidaknya korelasi antar variabel dapat digunakan dengan metode *bivariate correlation* atau sering disebut juga Product Moment Pearson. Dalam melakukan uji korelasi perlu memperhatikan *Test of Significant* yaitu meliputi Two-Tailed (uji dua sisi) digunakan dalam kondisi belum diketahui bentuk hubungan antar variabel dan One-Tailed (satu sisi) digunakan untuk menguji test of significant daari dua variabel akan tetapi telah diketahui adanya arah kecenderungan hubungan negative atau positif di antara dua variabel yang berhubungan.

Pengujian korelasi pertama dilakukan dengan data pre-test menghasilkan output sebagai berikut :

Correlations

| | | Memori | Task |
|---|---|---|---|
| Memori | Pearson Correlation | 1.00 | 1.00 |
| | Sig. (2-tailed) | | .00 |
| | N | 5 | 5 |
| Task | Pearson Correlation | 1.00 | 1.00 |
| | Sig. (2-tailed) | .00 | |
| | N | 5 | 5 |

Gambar 4.5 Uji korelasi Pre-test

Dari hasil di atas dapat diketahui bahwa besarnya penggunaan memori oleh server berdasarkan korelasi dengan banyaknya task yang berjalan dengan tingkat signifikansi sebesar 0,00. Hal ini berarti penggunaan memori tidak ada hubungannya dengan banyaknya task yang berjalan.

Untuk pengujian kedua yaitu uji korelasi post-test dapat dilihat di gambar 4.6 :

Correlations

| | | Memori | Task |
|---|---|---|---|
| Memori | Pearson Correlation | 1.00 | .79 |
| | Sig. (2-tailed) | | .11 |
| | N | 5 | 5 |
| Task | Pearson Correlation | .79 | 1.00 |
| | Sig. (2-tailed) | .11 | |
| | N | 5 | 5 |

Gambar 4.6 Uji korelasi post-test

Dapat kita lihat dari hasil uji korelasi antara task dan memori setelah diinstall Suricata berbeda dengan uji korelasi sebelumnya. Dengan tingkat korelasi sebesar 0,11 berarti penggunaan memori memiliki hubungan yang erat dengan banyaknya task.

#### 4.1.1.5 Uji T Dua Sampel Dependen

Langkah analisis selanjutnya yaitu menganalisis keefektifan penelitian ini. Apakah dengan menginstall Suricata IDS dapat mengurangi serangan DOS atau malah sebaliknya dapat membuat komputer server lebih cepat kehabisan memori. Sampel dependen atau sampel berpasangan biasanya diambil dari satu kelompok sampel yang diberikan dua perlakuan yang berbeda. Penelitian ini juga termasuk sampel dependen dikarenakan ada perlakuan berbeda yaitu penyerangan kepada komputer server sebelum Suricata IDS diinstall dan penyerangan sesudah IDS diinstall.

Untuk lebih jelasnya kita bandingkan data penggunaan konsumsi memori komputer server. Berikut data penggunaan konsumsi memori komputer server sebelum dan sesudah diinstall Suricata IDS :

Tabel 4.4 Data Penggunaan Memori Pre-test dan Post-test

| Pengujian | Memori | |
|---|---|---|
| | Sebelum | Sesudah |
| 1 | 770708 | 1182480 |
| 2 | 768208 | 1193232 |
| 3 | 841900 | 1197644 |
| 4 | 849700 | 1202396 |
| 5 | 851848 | 1207644 |

Dengan menggunakan tingkat kepercayaan 95 %. Apakah instalasi Suricata tersebut efektif untuk menghalau serangan DOS atau tidak ?

Dari permasalahan di atas dapat disusun hipotesa sebagai berikut :

- H0 : d = 0 (penggunaan memori sebelum dan sesudah diinstall Suricata sama).
- H0 : d ≠ 0 (penggunaan memori sebelum dan sesudah diinstall Suricata berbeda).

Kriteria pengujian data pre-test dan post-test ditetapkan sebagai berikut :

1. Jika nilai signifikansi lebih dari atau sama dengan 0,05 maka $H_0$ diterima.
2. Jika nilai signifikansi kurang dari 0,05 maka $H_0$ ditolak.

Paired Sample Statistics

| | | Mean | N | Std. Deviation | S.E. Mean |
|---|---|---|---|---|---|
| Pair 1 | Sebelum | 816472.80 | 5 | 43086.84 | 19269.02 |
| | Sesudah | 1196679.20 | 5 | 9583.06 | 4285.67 |

Paired Samples Correlations

| | | N | Correlation | Sig. |
|---|---|---|---|---|
| Pair 1 | Sebelum & Sesudah | 5 | .86 | .06 |

Paired Samples Test

| | Paired Differences | | | | | t | df | Sig. (2-tailed) |
|---|---|---|---|---|---|---|---|---|
| | | | | 95% Confidence Interval of the Difference | | | | |
| | Mean | Std. Deviation | Std. Error Mean | Lower | Upper | | | |
| Pair 1 Sebelum - Sesudah | -380206.40 | 35199.80 | 15741.83 | -423912.72 | -336500.08 | -24.15 | 4 | .00 |

Gambar 4.7 Uji T Sampel Dependen data penelitian.

Dari hasil analisis di atas dapat disimpulkan bahwa rata-rata memori komputer sebelum diinstall Suricata setelah diberi serangan DOS adalah 750.546,40 dengan standar deviasi 5.591,79 sedangkan penggunaan memori

setelah diinstall Suricata ialah sebesar 1.056.064,00 dengan nilai standar deviasi 4622,60.

Pada output kedua di atas angka signifikansi korelasi bernilai 0,06 antara dua variabel yaitu sebelum dan sesudah diinstall Suricata memiliki hubungan yang erat.

Sedangkan output ketiga setelah dibandingkan hasil uji T sampel dependen tampak bahwa nilai T* = -24,15. Dalam pengambilan kesimpulan digunakan nilai signifikansi. Tertera pada gambar di atas bahwa nilai signifikansi 0,00 < 0,005. Hal ini mutlak $H_0$ ditolak, sehingga dapat disimpulkan bahwa instalasi Suricata IDS benar-benar efektif dalam melakukan pertahanan terhadap serangan DOS.

Secara keseluruhan hasil penggunaan memori dengan menggunakan tool pembeda Suricata dan *treatment* berupa serangan IP Flooding dapat dilihat di tabel berikut ini :

Tabel 4.5 Data keseluruhan penggunaan memori

| Pre-test | | Post-test | |
|---|---|---|---|
| Normal | IP Flooding | Normal | IP Flooding |
| 585396 | 770708 | 1171784 | 1182480 |
| | 768208 | | 1193232 |
| | 841900 | | 1197644 |
| | 849700 | | 1202396 |
| | 851848 | | 1207644 |

#### 4.1.1.6. Data Pengujian Suricata

Setelah mengetahui bagaimana hasil akhir dari analisis pre-test dan post-test, ada baiknya kita sajikan hasil pengujian dilihat dari sistem Suricata itu sendiri. Setelah Suricata IDS diaktifkan, secara otomatis Suricata akan mencatat apapun yang berjalan di trafik jaringan baik itu lalu lintas jaringan normal maupun serangan. Di bawah ini adalah hasil dari pengujian post-test :

Tabel 4.6 Deteksi serangan

| pengujian | deteksi |
|---|---|
| 1 | 1419 |
| 2 | 2837 |
| 3 | 6038 |
| 4 | 9724 |
| 5 | 16705 |

Dari hasil pendeteksian di atas terlihat bahwa Suricata mampu mendeteksi adanya penyusup yang melakukan IP Flooding. Dari rangkaian pengujian dan analisis terhadap penggunaan memori baik sebelum dan sesudah diinstall Suricata dapat disimpulkan bahwa meskipun penggunaan memori setelah diinstall Suricata lebih tinggi disbanding memori yang digunakan sebelum adanya instalasi Suricata, akan tetapi Suricata mampu menghasilkan *alert* berupa output http.log, fast.log dan stats.log yang masing-masing dapat menerangkan dari mana penyerang berasal baik

berupa informasi IP address yang digunakan, http header yang dikirim maupun jenis serangan yang digunakan.

Di bawah ini adalah contoh output http.log dan fast.log :

```
06/21/2014-01:47:13.089609 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:34565 -> 192.168.1.1:80
06/21/2014-02:49:04.013171 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33311 -> 192.168.1.1:80
06/21/2014-02:49:04.013812 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33312 -> 192.168.1.1:80
06/21/2014-02:49:04.013887 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33313 -> 192.168.1.1:80
06/21/2014-02:49:04.014651 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33314 -> 192.168.1.1:80
06/21/2014-02:49:04.015444 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33315 -> 192.168.1.1:80
06/21/2014-02:49:04.016427 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33316 -> 192.168.1.1:80
06/21/2014-02:49:04.018106 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33319 -> 192.168.1.1:80
06/21/2014-02:49:04.019702 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33321 -> 192.168.1.1:80
06/21/2014-02:49:04.018738 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33320 -> 192.168.1.1:80
06/21/2014-02:49:04.019765 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33322 -> 192.168.1.1:80
06/21/2014-02:49:04.017191 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33317 -> 192.168.1.1:80
06/21/2014-02:49:04.017250 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33318 -> 192.168.1.1:80
06/21/2014-02:49:04.020710 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33323 -> 192.168.1.1:80
06/21/2014-02:49:05.005490 192.168.1.1 [**] / [**] JoeDog/1.00 [en] (X11; I; Siege 2.70) [**] 192.168.1.2:33324 -> 192.168.1.1:80
```

Gambar 4.8 Output http.log

Dapat dilihat dari gambar di atas bahwa Suricata mampu mengenali IP address yang telah melakukan IP Flooding adalah 192.168.1.1.

```
06/21/2014-01:46:16.900306  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32864
06/21/2014-01:46:16.902080  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32871
06/21/2014-01:46:16.905017  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32875
06/21/2014-01:46:16.905632  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32876
06/21/2014-01:46:16.905247  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32875
06/21/2014-01:46:16.902286  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32871
06/21/2014-01:46:16.902826  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32870
06/21/2014-01:46:16.903146  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32870
06/21/2014-01:46:16.902566  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32868
06/21/2014-01:46:16.902796  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32868
06/21/2014-01:46:16.903372  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32871
06/21/2014-01:46:16.903558  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32871
06/21/2014-01:46:17.894041  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32877
06/21/2014-01:46:17.894286  [**] [1:2200074:1] SURICATA TCPv4 invalid checksum [**] [Classification: (null)] [Priority: 3] (TCP) 192.168.1.1:80 -> 192.168.1.2:32877
```

Gambar 4.9 Output fast log

Serangan yang berhasil dideteksi berupa TCPv4 invalid checksum dengan klasifikasi keamanan tingkat 3 dari IP address 192.168.1.2 yaitu IP Flooding. Deteksi ini berjalan sesuai dengan *rule* yang telah dibuat didalam folder /etc/suricata/rules.

# DAFTAR PUSTAKA


*Keamanan Jaringan.* (2013, April 11). Retrieved Desember 05, 2013, from Wikipedia: http://id.wikipedia.org/wiki/Keamanan_jaringan

Al Fatta, H. (2007). *Analisis dan Perancangan Sistem Informasi untuk Keunggulan Bersaing Perusahaan & Organisasi Modern.* Yogyakarta: CV. Andi Offset.

Albin, E. (2011). *A Comparative Analysis of The Snort and Suricata Intrusion-Detection Systems.* Monterey: Naval Postgraduate School.

Alexander, L. A. (2010, November 19). *Ancaman keamanan data dan jenis gangguan.* Retrieved Desember 5, 2013, from Double klikk: http://dobelklikk.wordpress.com/2010/11/19/ancaman-keamanan-data-dan-jenis-jenis-gangguanancaman/

*Complete list of Suricata Features*. (n.d.). Retrieved March 20, 2014, from Suricata IDS: http://suricata-ids.org/features/all-features/

cyruslab. (2012, 10 18). *Building an IDS : installing snorby, suricata and barnyard2*. Retrieved March 24, 2014, from The Network Journal: http://cyruslab.net/2012/10/18/building-an-ids-part-1-installing-pre-requisites-and-snorby/

Day, D. J., & Burns, B. M. (2011). ICDS 2011. *A Performance Analysis of Snort and Suricata Network Intrusion Detection and Prevention Engines*, 187-192.

firnsy. (n.d.). *Barnyard2*. Retrieved March 24, 2014, from Github.com: https://github.com/firnsy/barnyard2

Fuzi, F. (2011). *An Analysis of Intrusion Detection System.* Melaka: Universiti Teknikal Malaysia Melaka.

Kacha, C., & Shevade, K. A. (2012). IJETAE_1212_44. *Comparison of Different Intrusion Detection and Prevention Systems*, 243-245.

Kusumawati, M. (2010). *Implementasi IDS (Intrusion Detection System) Serta Monitoring Jaringan dengan Interface Web Berbasis Base pada Keamanan Jaringan.* Depok: UI Press.

L. Person, L., & S. Davie, B. (2012). *Computer Networks: A Systems Approach.* Elsevier.

McRee, R. (2010). ISSA Journal. *Suricata: An introduction*, 40-42.

Meghanathan, D. N. (2012). *Intrusion Detection Systems.* Jackson State University.

Messer, W. H. (2011). *Performance Testing Suricata : The Effect of Configuration Variables On Offline Suricata Performance.* Georgia Institute of Technology.

Messer, W. H. (2011). *Performance Testing Suricata: The Effect of Configuration Variables On Offline Suricata Performance.* Georgia Institute of Technology.

Mulyono. (2013). *Perancangan dan Implementasi Sistem Monitoring Jaringan LAN (Local Area Network) dengan Notifikasi SMS.* Yogyakarta: UIN Sunan Kalijaga.

Naimzada, A. K., Stefani, S., & Torriero, A. (2009). *Networks, Topology and Dynamics: Theory and Applications to Economics and Social Systems.* Milano: Springer.

Panwar, S. S., Mao, S., Ryoo, J. d., & Li, Y. (2004). *TCP/IP Essentials : A Lab-Based Approach.* Cambridge University Press.

Putri, L. (2011). *Implementasi Intrusion Detection System (IDS) Menggunakan Snort pada Jaringan Wireless.* Jakarta: UIN Syarif Hidayatullah.

Qudratullah, M. F., & Suphandi, E. D. (n.d.). *Handout Praktikum Metode Statistika.* Yogyakarta.

Rahardjo, B. (2005). *Keamanan Sistem Informasi Berbasis Internet.* Jakarta: PT Insan Infonesia & PT INDOCISC.

Rob. (2013, July 2). *What is Linux*. Retrieved March 20, 2014, from Linux.org: http://www.linux.org/threads/what-is-linux.4076/

Saputra, A. (2005). *Pengembangan perangkat wireless IDS (Intrusion Detection System) berbasis embedded sytem .* Jakarta: UIN Syarif Hidayatulloh.

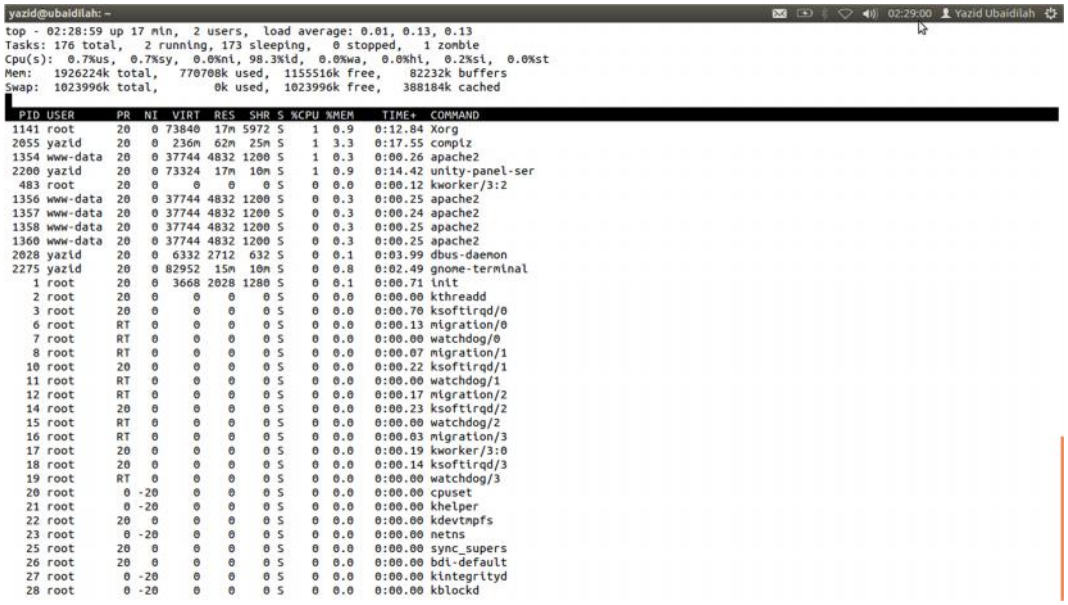
Stammler, J. H. (2011). *Suricata Performance White Paper.*

Syafrizal, M. (2005). *Pengantar Jaringan KOmputer.* Yogyakarta: Andi.

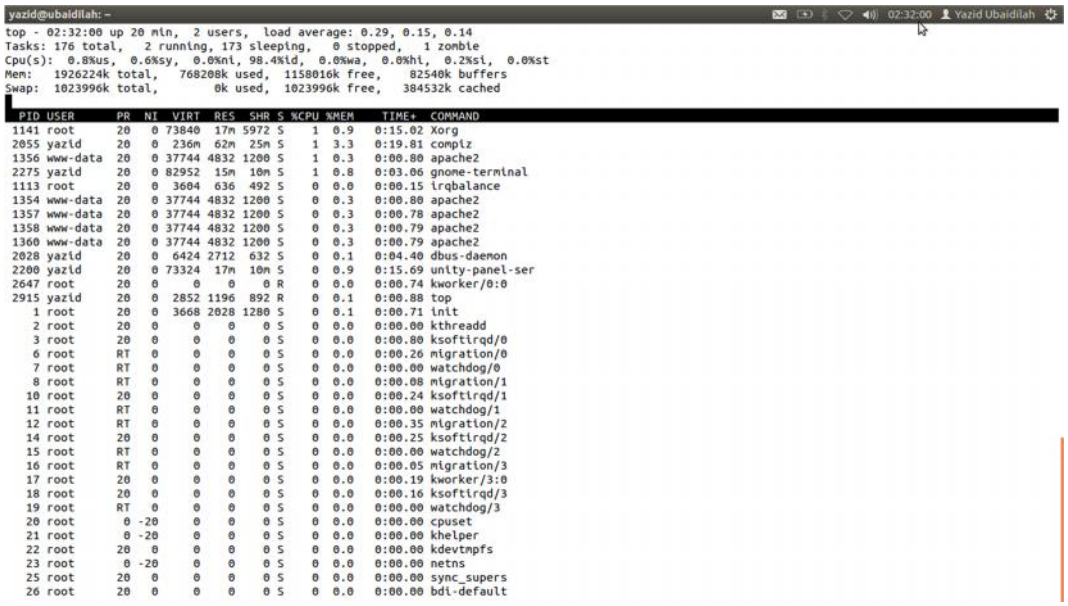
Yuhefizar. (2008). *10 Jam menguasai internet, teknologi dan aplikasinya.* Jakarta: Elex Media Komputindo.

# LAMPIRAN

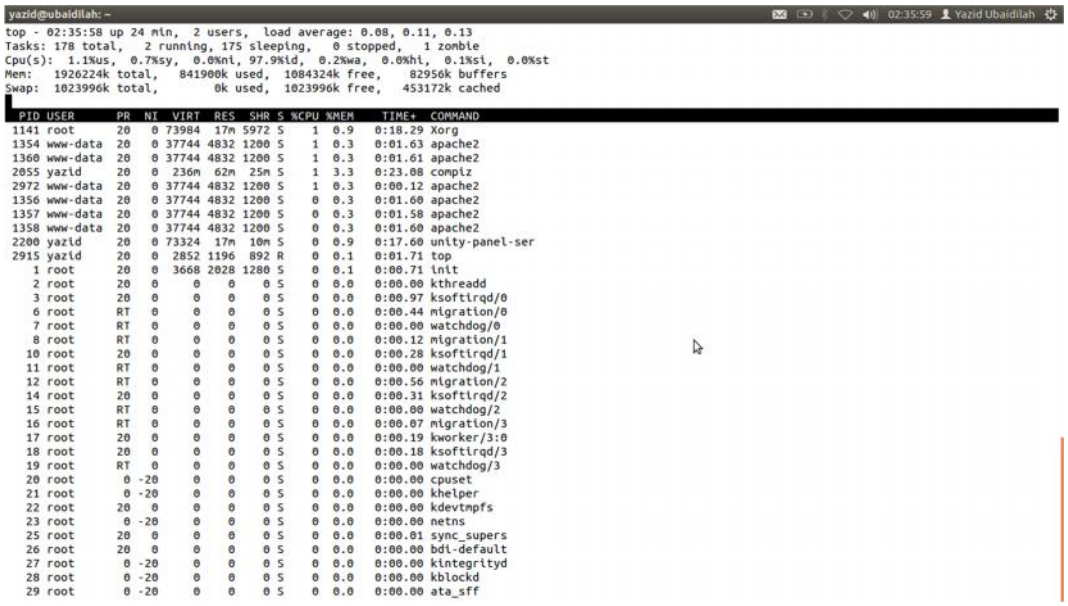
1. **Gambar pengujian pertama pre-test**



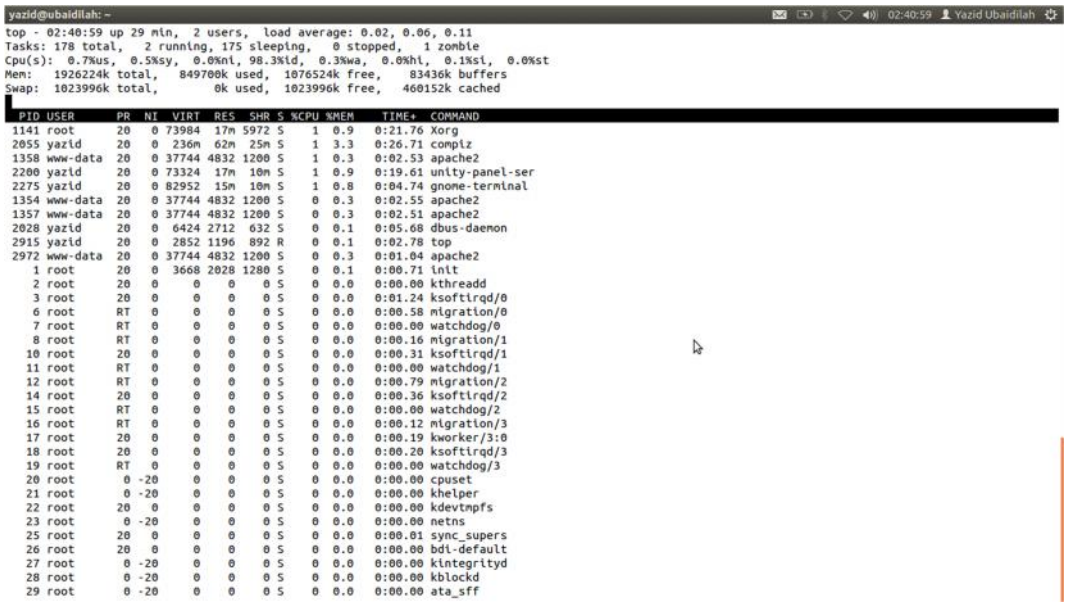

2. **Gambar pengujian kedua pre-test**



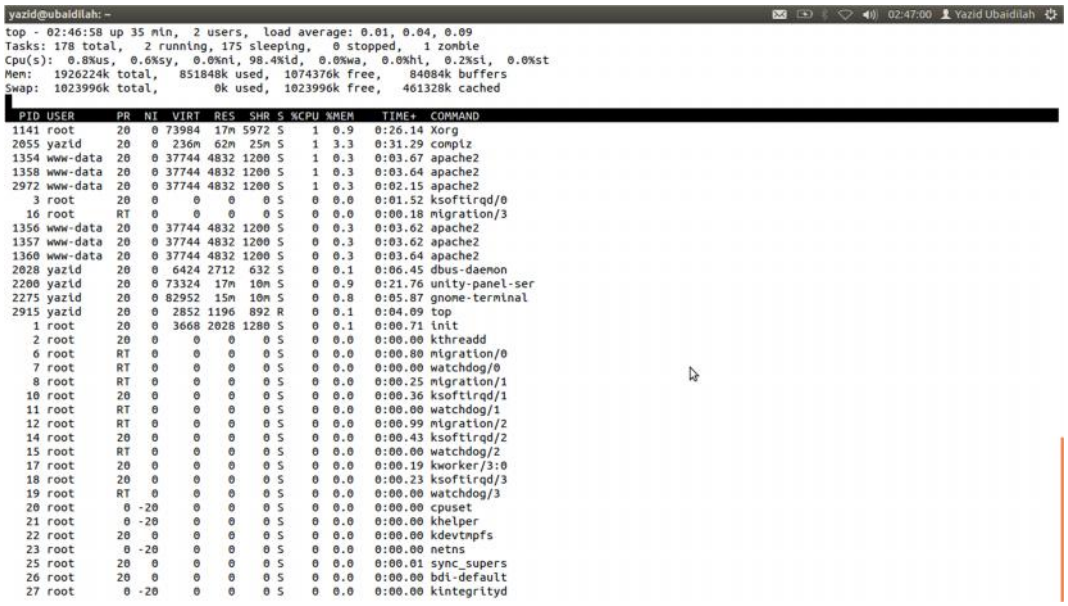
3. **Gambar pengujian ketiga pre-test**



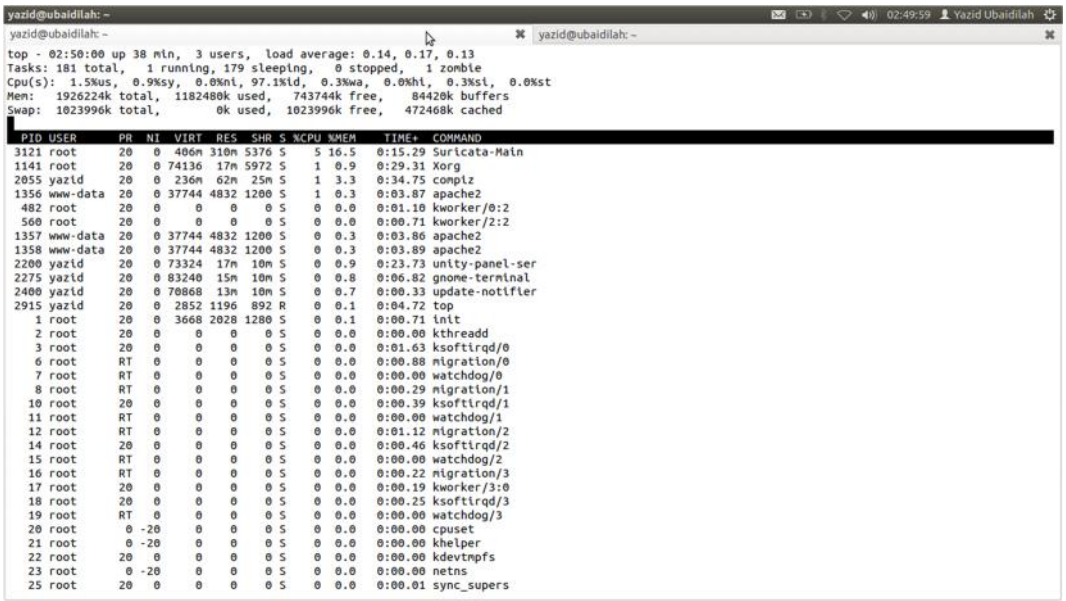

4. **Gambar pengujian ke-empat pre-test**



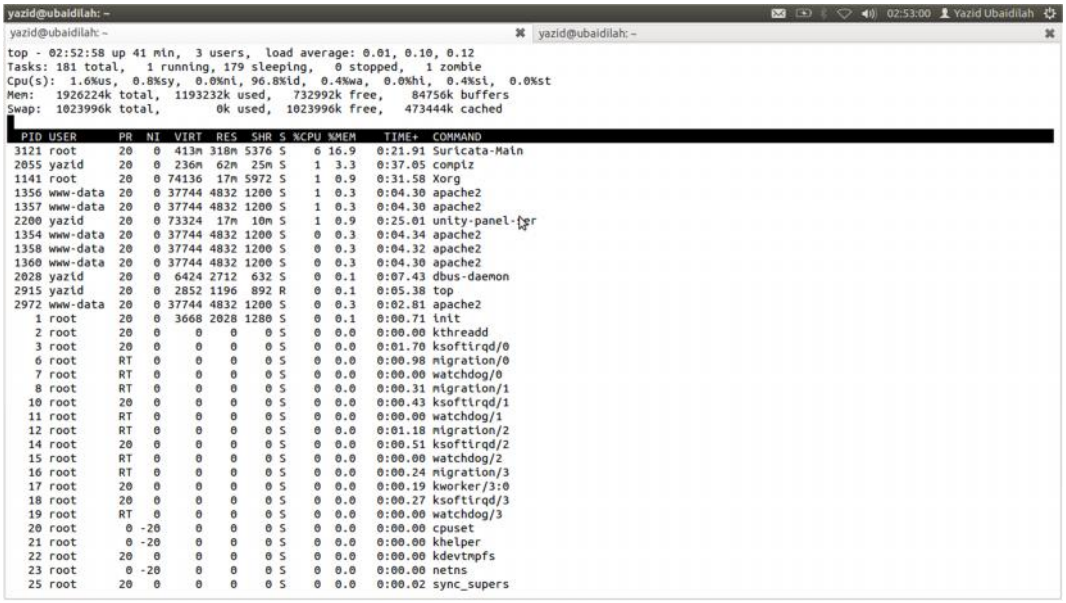
5. **Gambar pengujian kelima pre-test**



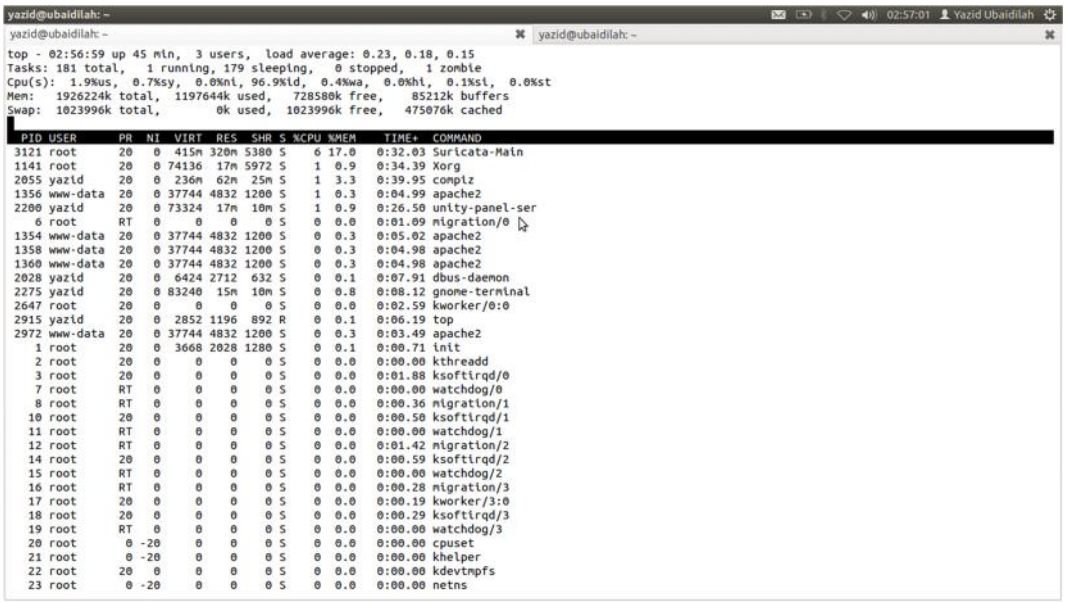

6. **Gambar pengujian pertama post-test**



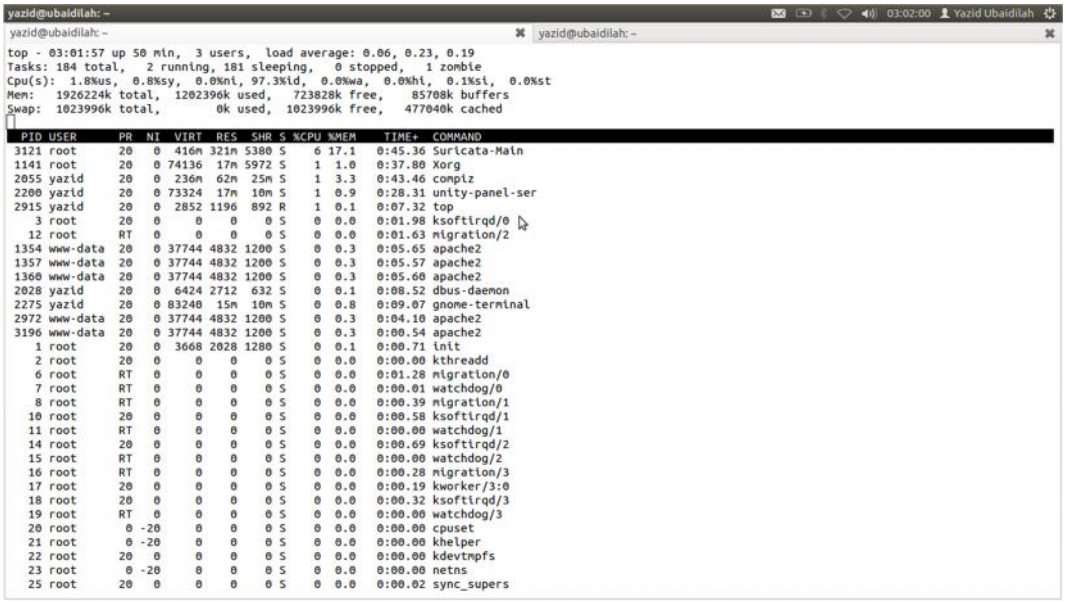
7. **Gambar pengujian kedua post-test**



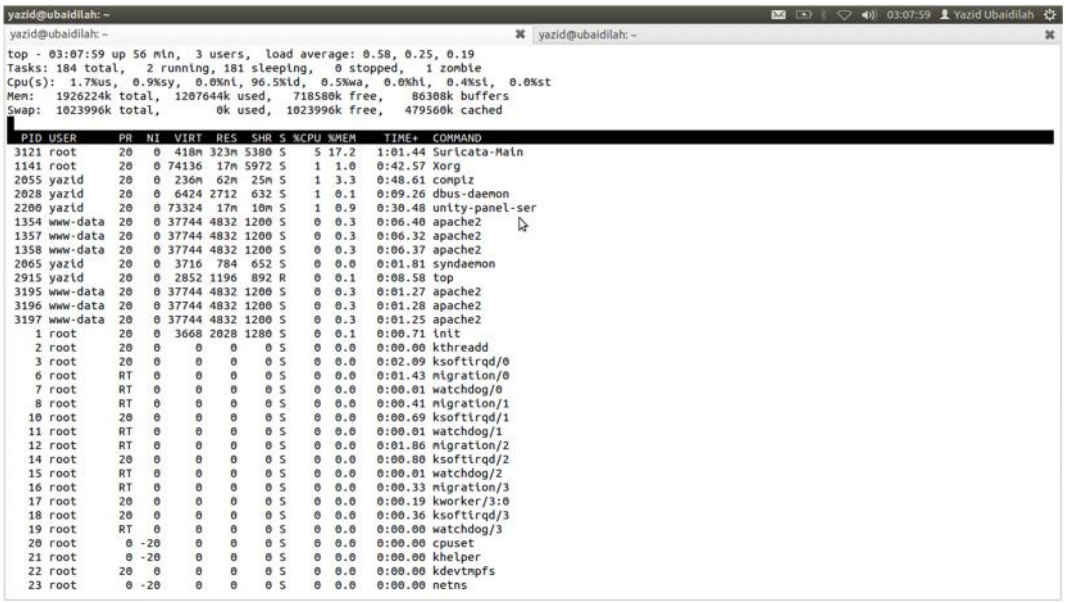

8. **Gambar pengujian ketiga post-test**



9. **Gambar pengujian ke-empat post-test**




10. **Gambar pengujian kelima post-test**



**11. File konfigurasi suricata.yaml**

```
%YAML 1.1
---

# Suricata configuration file. In addition to the comments
describing all
# options in this file, full documentation can be found at:
#
https://redmine.openinfosecfoundation.org/projects/suricata/wi
ki/Suricatayaml


# Number of packets allowed to be processed simultaneously.
Default is a
# conservative 1024. A higher number will make sure CPU's/CPU
cores will be
# more easily kept busy, but may negatively impact caching.
#
# If you are using the CUDA pattern matcher (mpm-algo: ac-
cuda), different rules
# apply. In that case try something like 60000 or more. This
is because the CUDA
# pattern matcher buffers and scans as many packets as
possible in parallel.
#max-pending-packets: 1024

# Runmode the engine should use. Please check --list-runmodes
to get the available
# runmodes for each packet acquisition method. Defaults to
"autofp" (auto flow pinned
# load balancing).
#runmode: autofp

# Specifies the kind of flow load balancer used by the flow
pinned autofp mode.
#
# Supported schedulers are:
#
# round-robin       - Flows assigned to threads in a round
robin fashion.
# active-packets    - Flows assigned to threads that have the
lowest number of
#                     unprocessed packets (default).
# hash              - Flow alloted usihng the address hash.
More of a random
#                     technique. Was the default in Suricata
1.2.1 and older.
#
#autofp-scheduler: active-packets

# If suricata box is a router for the sniffed networks, set it
to 'router'. If
```

```
# it is a pure sniffing setup, set it to 'sniffer-only'.
# If set to auto, the variable is internally switch to
'router' in IPS mode
# and 'sniffer-only' in IDS mode.
# This feature is currently only used by the reject* keywords.
host-mode: auto

# Run suricata as user and group.
#run-as:
#  user: suri
#  group: suri

# Default pid file.
# Will use this file if no --pidfile in command options.
#pid-file: /var/run/suricata.pid

# Daemon working directory
# Suricata will change directory to this one if provided
# Default: "/"
#daemon-directory: "/"

# Preallocated size for packet. Default is 1514 which is the
classical
# size for pcap on ethernet. You should adjust this value to
the highest
# packet size (MTU + hardware header) on your system.
#default-packet-size: 1514

# The default logging directory.  Any log or output file will
be
# placed here if its not specified with a full path name.
This can be
# overridden with the -l command line parameter.
default-log-dir: /var/log/suricata/

# Unix command socket can be used to pass commands to
suricata.
# An external tool can then connect to get information from
suricata
# or trigger some modifications of the engine. Set enabled to
yes
# to activate the feature. You can use the filename variable
to set
# the file name of the socket.
unix-command:
  enabled: no
  #filename: custom.socket

# Configure the type of alert (and other) logging you would
like.
outputs:

  # a line based alerts log similar to Snort's fast.log
  - fast:
```

```
      enabled: yes
      filename: fast.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or
'unix_dgram'

  # Extensible Event Format (nicknamed EVE) event log in JSON
format
  - eve-log:
      enabled: yes
      type: file #file|syslog|unix_dgram|unix_stream
      filename: eve.json
      # the following are valid when type: syslog above
      #identity: "suricata"
      #facility: local5
      #level: Info ## possible levels: Emergency, Alert,
Critical,
                   ## Error, Warning, Notice, Info, Debug
      types:
        - alert
        - http:
            extended: yes     # enable this for extended
logging information
            # custom allows additional http fields to be
included in eve-log
            # the example below adds three additional fields
when uncommented
            #custom: [Accept-Encoding, Accept-Language,
Authorization]
        - dns
        - tls:
            extended: yes     # enable this for extended
logging information
        - files:
            force-magic: no   # force logging magic on all
logged files
            force-md5: no     # force logging of md5 checksums
        #- drop
        - ssh

  # alert output for use with Barnyard2
  - unified2-alert:
      enabled: yes
      filename: unified2.alert

      # File size limit.  Can be specified in kb, mb, gb.
Just a number
      # is parsed as bytes.
      #limit: 32mb

      # Sensor ID field of unified2 alerts.
      #sensor-id: 0
```

```
      # HTTP X-Forwarded-For support by adding the unified2
extra header that
      # will contain the actual client IP address or by
overwriting the source
      # IP address (helpful when inspecting traffic that is
being reversed
      # proxied).
      xff:
        enabled: no
        # Two operation modes are available, "extra-data" and
"overwrite". Note
        # that in the "overwrite" mode, if the reported IP
address in the HTTP
        # X-Forwarded-For header is of a different version of
the packet
        # received, it will fall-back to "extra-data" mode.
        mode: extra-data
        # Header name were the actual IP address will be
reported, if more than
        # one IP address is present, the last IP address will
be the one taken
        # into consideration.
        header: X-Forwarded-For

  # a line based log of HTTP requests (no alerts)
  - http-log:
      enabled: yes
      filename: http.log
      append: yes
      #extended: yes     # enable this for extended logging
information
      #custom: yes       # enabled the custom logging format
(defined by customformat)
      #customformat: "%{%D-%H:%M:%S}t.%z %{X-Forwarded-For}i
%H %m %h %u %s %B %a:%p -> %A:%P"
      #filetype: regular # 'regular', 'unix_stream' or
'unix_dgram'

  # a line based log of TLS handshake parameters (no alerts)
  - tls-log:
      enabled: no  # Log TLS connections.
      filename: tls.log # File to store TLS logs.
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or
'unix_dgram'
      #extended: yes # Log extended information like
fingerprint
      certs-log-dir: certs # directory to store the
certificates files

  # a line based log of DNS requests and/or replies (no
alerts)
  - dns-log:
      enabled: no
```

```
      filename: dns.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or
'unix_dgram'

  # a line based log to used with pcap file study.
  # this module is dedicated to offline pcap parsing (empty
output
  # if used with another kind of input). It can interoperate
with
  # pcap parser like wireshark via the suriwire plugin.
  - pcap-info:
      enabled: no

  # Packet log... log packets in pcap format. 2 modes of
operation: "normal"
  # and "sguil".
  #
  # In normal mode a pcap file "filename" is created in the
default-log-dir,
  # or are as specified by "dir". In Sguil mode "dir"
indicates the base directory.
  # In this base dir the pcaps are created in th directory
structure Sguil expects:
  #
  # $sguil-base-dir/YYYY-MM-DD/$filename.<timestamp>
  #
  # By default all packets are logged except:
  # - TCP streams beyond stream.reassembly.depth
  # - encrypted streams after the key exchange
  #
  - pcap-log:
      enabled:  no
      filename: log.pcap

      # File size limit.  Can be specified in kb, mb, gb.
Just a number
      # is parsed as bytes.
      limit: 1000mb

      # If set to a value will enable ring buffer mode. Will
keep Maximum of "max-files" of size "limit"
      max-files: 2000

      mode: normal # normal or sguil.
      #sguil-base-dir: /nsm_data/
      #ts-format: usec # sec or usec second format (default)
is filename.sec usec is filename.sec.usec
      use-stream-depth: no #If set to "yes" packets seen after
reaching stream inspection depth are ignored. "no" logs all
packets

  # a full alerts log containing much information for
signature writers
```

```
  # or for investigating suspected false positives.
  - alert-debug:
      enabled: no
      filename: alert-debug.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or
'unix_dgram'

  # alert output to prelude (http://www.prelude-
technologies.com/) only
  # available if Suricata has been compiled with --enable-
prelude
  - alert-prelude:
      enabled: no
      profile: suricata
      log-packet-content: no
      log-packet-header: yes

  # Stats.log contains data from various counters of the
suricata engine.
  # The interval field (in seconds) tells after how long
output will be written
  # on the log file.
  - stats:
      enabled: yes
      filename: stats.log
      interval: 8

  # a line based alerts log similar to fast.log into syslog
  - syslog:
      enabled: no
      # reported identity to syslog. If ommited the program
name (usually
      # suricata) will be used.
      #identity: "suricata"
      facility: local5
      #level: Info ## possible levels: Emergency, Alert,
Critical,
                   ## Error, Warning, Notice, Info, Debug

  # a line based information for dropped packets in IPS mode
  - drop:
      enabled: no
      filename: drop.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or
'unix_dgram'

  # output module to store extracted files to disk
  #
  # The files are stored to the log-dir in a format
"file.<id>" where <id> is
  # an incrementing number starting at 1. For each file
"file.<id>" a meta
```

```
  # file "file.<id>.meta" is created.
  #
  # File extraction depends on a lot of things to be fully
done:
  # - stream reassembly depth. For optimal results, set this
to 0 (unlimited)
  # - http request / response body sizes. Again set to 0 for
optimal results.
  # - rules that contain the "filestore" keyword.
  - file-store:
      enabled: no       # set to yes to enable
      log-dir: files    # directory to store the files
      force-magic: no   # force logging magic on all stored
files
      force-md5: no     # force logging of md5 checksums
      #waldo: file.waldo # waldo file to store the file_id
across runs

  # output module to log files tracked in a easily parsable
json format
  - file-log:
      enabled: no
      filename: files-json.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or
'unix_dgram'

      force-magic: no   # force logging magic on all logged
files
      force-md5: no     # force logging of md5 checksums

# Magic file. The extension .mgc is added to the value here.
#magic-file: /usr/share/file/magic
magic-file: /usr/share/file/magic

# When running in NFQ inline mode, it is possible to use a
simulated
# non-terminal NFQUEUE verdict.
# This permit to do send all needed packet to suricata via
this a rule:
#        iptables -I FORWARD -m mark ! --mark $MARK/$MASK -j
NFQUEUE
# And below, you can have your standard filtering ruleset. To
activate
# this mode, you need to set mode to 'repeat'
# If you want packet to be sent to another queue after an
ACCEPT decision
# set mode to 'route' and set next-queue value.
# On linux >= 3.1, you can set batchcount to a value > 1 to
improve performance
# by processing several packets before sending a verdict
(worker runmode only).
# On linux >= 3.6, you can set the fail-open option to yes to
have the kernel
```

```
# accept the packet if suricata is not able to keep pace.
nfq:
#  mode: accept
#  repeat-mark: 1
#  repeat-mask: 1
#  route-queue: 2
#  batchcount: 20
#  fail-open: yes

# af-packet support
# Set threads to > 1 to use PACKET_FANOUT support
af-packet:
  - interface: eth0
    # Number of receive threads (>1 will enable experimental
flow pinned
    # runmode)
    threads: 1
    # Default clusterid.  AF_PACKET will load balance packets
based on flow.
    # All threads/processes that will participate need to have
the same
    # clusterid.
    cluster-id: 99
    # Default AF_PACKET cluster type. AF_PACKET can load
balance per flow or per hash.
    # This is only supported for Linux kernel > 3.1
    # possible value are:
    #  * cluster_round_robin: round robin load balancing
    #  * cluster_flow: all packets of a given flow are send to
the same socket
    #  * cluster_cpu: all packets treated in kernel by a CPU
are send to the same socket
    cluster-type: cluster_flow
    # In some fragmentation case, the hash can not be
computed. If "defrag" is set
    # to yes, the kernel will do the needed defragmentation
before sending the packets.
    defrag: yes
    # To use the ring feature of AF_PACKET, set 'use-mmap' to
yes
    use-mmap: yes
    # Ring size will be computed with respect to
max_pending_packets and number
    # of threads. You can set manually the ring size in number
of packets by setting
    # the following value. If you are using flow cluster-type
and have really network
    # intensive single-flow you could want to set the ring-
size independantly of the number
    # of threads:
    #ring-size: 2048
    # On busy system, this could help to set it to yes to
recover from a packet drop
```

```
    # phase. This will result in some packets (at max a ring
flush) being non treated.
    #use-emergency-flush: yes
    # recv buffer size, increase value could improve
performance
    # buffer-size: 32768
    # Set to yes to disable promiscuous mode
    # disable-promisc: no
    # Choose checksum verification mode for the interface. At
the moment
    # of the capture, some packets may be with an invalid
checksum due to
    # offloading to the network card of the checksum
computation.
    # Possible values are:
    #  - kernel: use indication sent by kernel for each packet
(default)
    #  - yes: checksum validation is forced
    #  - no: checksum validation is disabled
    #  - auto: suricata uses a statistical approach to detect
when
    #  checksum off-loading is used.
    # Warning: 'checksum-validation' must be set to yes to
have any validation
    #checksum-checks: kernel
    # BPF filter to apply to this interface. The pcap filter
syntax apply here.
    #bpf-filter: port 80 or udp
    # You can use the following variables to activate
AF_PACKET tap od IPS mode.
    # If copy-mode is set to ips or tap, the traffic coming to
the current
    # interface will be copied to the copy-iface interface. If
'tap' is set, the
    # copy is complete. If 'ips' is set, the packet matching a
'drop' action
    # will not be copied.
    #copy-mode: ips
    #copy-iface: eth1
  - interface: eth1
    threads: 1
    cluster-id: 98
    cluster-type: cluster_flow
    defrag: yes
    # buffer-size: 32768
    # disable-promisc: no
  # Put default values here
  - interface: default
    #threads: 2
    #use-mmap: yes

legacy:
  uricontent: enabled
```

```
# You can specify a threshold config file by setting
"threshold-file"
# to the path of the threshold config file:
# threshold-file: /etc/suricata/threshold.config

# The detection engine builds internal groups of signatures.
The engine
# allow us to specify the profile to use for them, to manage
memory on an
# efficient way keeping a good performance. For the profile
keyword you
# can use the words "low", "medium", "high" or "custom". If
you use custom
# make sure to define the values at "- custom-values" as your
convenience.
# Usually you would prefer medium/high/low.
#
# "sgh mpm-context", indicates how the staging should allot
mpm contexts for
# the signature groups.  "single" indicates the use of a
single context for
# all the signature group heads.  "full" indicates a mpm-
context for each
# group head.  "auto" lets the engine decide the distribution
of contexts
# based on the information the engine gathers on the patterns
from each
# group head.
#
# The option inspection-recursion-limit is used to limit the
recursive calls
# in the content inspection code.  For certain payload-sig
combinations, we
# might end up taking too much time in the content inspection
code.
# If the argument specified is 0, the engine uses an
internally defined
# default limit.  On not specifying a value, we use no limits
on the recursion.
detect-engine:
  - profile: medium
  - custom-values:
      toclient-src-groups: 2
      toclient-dst-groups: 2
      toclient-sp-groups: 2
      toclient-dp-groups: 3
      toserver-src-groups: 2
      toserver-dst-groups: 4
      toserver-sp-groups: 2
      toserver-dp-groups: 25
  - sgh-mpm-context: auto
  - inspection-recursion-limit: 3000
  # When rule-reload is enabled, sending a USR2 signal to the
Suricata process
```

```
  # will trigger a live rule reload. Experimental feature, use
with care.
  #- rule-reload: true
  # If set to yes, the loading of signatures will be made
after the capture
  # is started. This will limit the downtime in IPS mode.
  #- delayed-detect: yes

# Suricata is multi-threaded. Here the threading can be
influenced.
threading:
  # On some cpu's/architectures it is beneficial to tie
individual threads
  # to specific CPU's/CPU cores. In this case all threads are
tied to CPU0,
  # and each extra CPU/core has one "detect" thread.
  #
  # On Intel Core2 and Nehalem CPU's enabling this will
degrade performance.
  #
  set-cpu-affinity: no
  # Tune cpu affinity of suricata threads. Each family of
threads can be bound
  # on specific CPUs.
  cpu-affinity:
    - management-cpu-set:
        cpu: [ 0 ]  # include only these cpus in affinity
settings
    - receive-cpu-set:
        cpu: [ 0 ]  # include only these cpus in affinity
settings
    - decode-cpu-set:
        cpu: [ 0, 1 ]
        mode: "balanced"
    - stream-cpu-set:
        cpu: [ "0-1" ]
    - detect-cpu-set:
        cpu: [ "all" ]
        mode: "exclusive" # run detect threads in these cpus
        # Use explicitely 3 threads and don't compute number
by using
        # detect-thread-ratio variable:
        # threads: 3
        prio:
          low: [ 0 ]
          medium: [ "1-2" ]
          high: [ 3 ]
          default: "medium"
    - verdict-cpu-set:
        cpu: [ 0 ]
        prio:
          default: "high"
    - reject-cpu-set:
        cpu: [ 0 ]
```

```
        prio:
          default: "low"
    - output-cpu-set:
        cpu: [ "all" ]
        prio:
           default: "medium"
  #
  # By default Suricata creates one "detect" thread per
available CPU/CPU core.
  # This setting allows controlling this behaviour. A ratio
setting of 2 will
  # create 2 detect threads for each CPU/CPU core. So for a
dual core CPU this
  # will result in 4 detect threads. If values below 1 are
used, less threads
  # are created. So on a dual core CPU a setting of 0.5
results in 1 detect
  # thread being created. Regardless of the setting at a
minimum 1 detect
  # thread will always be created.
  #
  detect-thread-ratio: 1.5

# Cuda configuration.
cuda:
  # The "mpm" profile.  On not specifying any of these
parameters, the engine's
  # internal default values are used, which are same as the
ones specified in
  # in the default conf file.
  mpm:
    # The minimum length required to buffer data to the gpu.
    # Anything below this is MPM'ed on the CPU.
    # Can be specified in kb, mb, gb.  Just a number indicates
it's in bytes.
    # A value of 0 indicates there's no limit.
    data-buffer-size-min-limit: 0
    # The maximum length for data that we would buffer to the
gpu.
    # Anything over this is MPM'ed on the CPU.
    # Can be specified in kb, mb, gb.  Just a number indicates
it's in bytes.
    data-buffer-size-max-limit: 1500
    # The ring buffer size used by the CudaBuffer API to
buffer data.
    cudabuffer-buffer-size: 500mb
    # The max chunk size that can be sent to the gpu in a
single go.
    gpu-transfer-size: 50mb
    # The timeout limit for batching of packets in
microseconds.
    batching-timeout: 2000
    # The device to use for the mpm.  Currently we don't
support load balancing
```

```
    # on multiple gpus.  In case you have multiple devices on
your system, you
    # can specify the device to use, using this conf.  By
default we hold 0, to
    # specify the first device cuda sees.  To find out device-
id associated with
    # the card(s) on the system run "suricata --list-cuda-
cards".
    device-id: 0
    # No of Cuda streams used for asynchronous processing. All
values > 0 are valid.
    # For this option you need a device with Compute
Capability > 1.0.
    cuda-streams: 2

# Select the multi pattern algorithm you want to run for
scan/search the
# in the engine. The supported algorithms are b2g, b2gc, b2gm,
b3g, wumanber,
# ac and ac-gfbs.
#
# The mpm you choose also decides the distribution of mpm
contexts for
# signature groups, specified by the conf - "detect-
engine.sgh-mpm-context".
# Selecting "ac" as the mpm would require "detect-engine.sgh-
mpm-context"
# to be set to "single", because of ac's memory requirements,
unless the
# ruleset is small enough to fit in one's memory, in which
case one can
# use "full" with "ac".  Rest of the mpms can be run in "full"
mode.
#
# There is also a CUDA pattern matcher (only available if
Suricata was
# compiled with --enable-cuda: b2g_cuda. Make sure to update
your
# max-pending-packets setting above as well if you use
b2g_cuda.

mpm-algo: ac

# The memory settings for hash size of these algorithms can
vary from lowest
# (2048) - low (4096) - medium (8192) - high (16384) - higher
(32768) - max
# (65536). The bloomfilter sizes of these algorithms can vary
from low (512) -
# medium (1024) - high (2048).
#
# For B2g/B3g algorithms, there is a support for two different
scan/search
```

```
# algorithms. For B2g the scan algorithms are B2gScan &
B2gScanBNDMq, and
# search algorithms are B2gSearch & B2gSearchBNDMq. For B3g
scan algorithms
# are B3gScan & B3gScanBNDMq, and search algorithms are
B3gSearch &
# B3gSearchBNDMq.
#
# For B2g the different scan/search algorithms and, hash and
bloom
# filter size settings. For B3g the different scan/search
algorithms and, hash
# and bloom filter size settings. For wumanber the hash and
bloom filter size
# settings.

pattern-matcher:
  - b2gc:
      search-algo: B2gSearchBNDMq
      hash-size: low
      bf-size: medium
  - b2gm:
      search-algo: B2gSearchBNDMq
      hash-size: low
      bf-size: medium
  - b2g:
      search-algo: B2gSearchBNDMq
      hash-size: low
      bf-size: medium
  - b3g:
      search-algo: B3gSearchBNDMq
      hash-size: low
      bf-size: medium
  - wumanber:
      hash-size: low
      bf-size: medium

# Defrag settings:

defrag:
  memcap: 32mb
  hash-size: 65536
  trackers: 65535 # number of defragmented flows to follow
  max-frags: 65535 # number of fragments to keep (higher than
trackers)
  prealloc: yes
  timeout: 60

# Enable defrag per host settings
#  host-config:
#
#    - dmz:
#        timeout: 30
```

```
#        address: [192.168.1.0/24, 127.0.0.0/8, 1.1.1.0/24,
2.2.2.0/24, "1.1.1.1", "2.2.2.2", "::1"]
#
#    - lan:
#        timeout: 45
#        address:
#          - 192.168.0.0/24
#          - 192.168.10.0/24
#          - 172.16.14.0/24

# Flow settings:
# By default, the reserved memory (memcap) for flows is 32MB.
This is the limit
# for flow allocation inside the engine. You can change this
value to allow
# more memory usage for flows.
# The hash-size determine the size of the hash used to
identify flows inside
# the engine, and by default the value is 65536.
# At the startup, the engine can preallocate a number of
flows, to get a better
# performance. The number of flows preallocated is 10000 by
default.
# emergency-recovery is the percentage of flows that the
engine need to
# prune before unsetting the emergency state. The emergency
state is activated
# when the memcap limit is reached, allowing to create new
flows, but
# prunning them with the emergency timeouts (they are defined
below).
# If the memcap is reached, the engine will try to prune flows
# with the default timeouts. If it doens't find a flow to
prune, it will set
# the emergency bit and it will try again with more agressive
timeouts.
# If that doesn't work, then it will try to kill the last time
seen flows
# not in use.
# The memcap can be specified in kb, mb, gb.  Just a number
indicates it's
# in bytes.

flow:
  memcap: 64mb
  hash-size: 65536
  prealloc: 10000
  emergency-recovery: 30

# This option controls the use of vlan ids in the flow (and
defrag)
# hashing. Normally this should be enabled, but in some
(broken)
```

```
# setups where both sides of a flow are not tagged with the
same vlan
# tag, we can ignore the vlan id's in the flow hashing.
vlan:
  use-for-tracking: true

# Specific timeouts for flows. Here you can specify the
timeouts that the
# active flows will wait to transit from the current state to
another, on each
# protocol. The value of "new" determine the seconds to wait
after a hanshake or
# stream startup before the engine free the data of that flow
it doesn't
# change the state to established (usually if we don't receive
more packets
# of that flow). The value of "established" is the amount of
# seconds that the engine will wait to free the flow if it
spend that amount
# without receiving new packets or closing the connection.
"closed" is the
# amount of time to wait after a flow is closed (usually
zero).
#
# There's an emergency mode that will become active under
attack circumstances,
# making the engine to check flow status faster. This
configuration variables
# use the prefix "emergency-" and work similar as the normal
ones.
# Some timeouts doesn't apply to all the protocols, like
"closed", for udp and
# icmp.

flow-timeouts:

  default:
    new: 30
    established: 300
    closed: 0
    emergency-new: 10
    emergency-established: 100
    emergency-closed: 0
  tcp:
    new: 60
    established: 3600
    closed: 120
    emergency-new: 10
    emergency-established: 300
    emergency-closed: 20
  udp:
    new: 30
    established: 300
    emergency-new: 10
```

```
    emergency-established: 100
  icmp:
    new: 30
    established: 300
    emergency-new: 10
    emergency-established: 100

# Stream engine settings. Here the TCP stream tracking and
reassembly
# engine is configured.
#
# stream:
#   memcap: 32mb               # Can be specified in kb, mb,
gb.  Just a
#                              # number indicates it's in
bytes.
#   checksum-validation: yes   # To validate the checksum of
received
#                              # packet. If csum validation
is specified as
#                              # "yes", then packet with
invalid csum will not
#                              # be processed by the engine
stream/app layer.
#                              # Warning: locally generated
trafic can be
#                              # generated without checksum
due to hardware offload
#                              # of checksum. You can control
the handling of checksum
#                              # on a per-interface basis via
the 'checksum-checks'
#                              # option
#   prealloc-sessions: 2k      # 2k sessions prealloc'd per
stream thread
#   midstream: false           # don't allow midstream
session pickups
#   async-oneside: false       # don't enable async stream
handling
#   inline: no                 # stream inline mode
#   max-synack-queued: 5       # Max different SYN/ACKs to
queue
#
#   reassembly:
#     memcap: 64mb             # Can be specified in kb, mb,
gb.  Just a number
#                              # indicates it's in bytes.
#     depth: 1mb               # Can be specified in kb, mb,
gb.  Just a number
#                              # indicates it's in bytes.
#     toserver-chunk-size: 2560 # inspect raw stream in chunks
of at least
#                              # this size.  Can be specified
in kb, mb,
```

```
#                                 # gb.  Just a number indicates
it's in bytes.
#                                 # The max acceptable size is
4024 bytes.
#     toclient-chunk-size: 2560 # inspect raw stream in chunks
of at least
#                                 # this size.  Can be specified
in kb, mb,
#                                 # gb.  Just a number indicates
it's in bytes.
#                                 # The max acceptable size is
4024 bytes.
#     randomize-chunk-size: yes # Take a random value for
chunk size around the specified value.
#                                 # This lower the risk of some
evasion technics but could lead
#                                 # detection change between
runs. It is set to 'yes' by default.
#     randomize-chunk-range: 10 # If randomize-chunk-size is
active, the value of chunk-size is
#                                 # a random value between (1 -
randomize-chunk-range/100)*randomize-chunk-size
#                                 # and (1 + randomize-chunk-
range/100)*randomize-chunk-size. Default value
#                                 # of randomize-chunk-range is
10.
#
#     raw: yes                  # 'Raw' reassembly enabled or
disabled.
#                                 # raw is for content
inspection by detection
#                                 # engine.
#
#     chunk-prealloc: 250       # Number of preallocated
stream chunks. These
#                                 # are used during stream
inspection (raw).
#     segments:                 # Settings for reassembly
segment pool.
#       - size: 4               # Size of the (data)segment
for a pool
#         prealloc: 256         # Number of segments to
prealloc and keep
#                                 # in the pool.
#
stream:
  memcap: 32mb
  checksum-validation: yes      # reject wrong csums
  inline: auto                  # auto will use inline mode in
IPS mode, yes or no set it statically
  reassembly:
    memcap: 128mb
    depth: 1mb                  # reassemble 1mb into a stream
    toserver-chunk-size: 2560
```

```
    toclient-chunk-size: 2560
    randomize-chunk-size: yes
    #randomize-chunk-range: 10
    #raw: yes
    #chunk-prealloc: 250
    #segments:
    #  - size: 4
    #    prealloc: 256
    #  - size: 16
    #    prealloc: 512
    #  - size: 112
    #    prealloc: 512
    #  - size: 248
    #    prealloc: 512
    #  - size: 512
    #    prealloc: 512
    #  - size: 768
    #    prealloc: 1024
    #  - size: 1448
    #    prealloc: 1024
    #  - size: 65535
    #    prealloc: 128

# Host table:
#
# Host table is used by tagging and per host thresholding
subsystems.
#
host:
  hash-size: 4096
  prealloc: 1000
  memcap: 16777216

# Logging configuration.  This is not about logging IDS
alerts, but
# IDS output about what its doing, errors, etc.
logging:

  # The default log level, can be overridden in an output
section.
  # Note that debug level logging will only be emitted if
Suricata was
  # compiled with the --enable-debug configure option.
  #
  # This value is overriden by the SC_LOG_LEVEL env var.
  default-log-level: notice

  # The default output format.  Optional parameter, should
default to
  # something reasonable if not provided.  Can be overriden in
an
  # output section.  You can leave this out to get the
default.
  #
```

```
  # This value is overriden by the SC_LOG_FORMAT env var.
  #default-log-format: "[%i] %t - (%f:%l) <%d> (%n) -- "

  # A regex to filter output.  Can be overridden in an output
section.
  # Defaults to empty (no filter).
  #
  # This value is overriden by the SC_LOG_OP_FILTER env var.
  default-output-filter:

  # Define your logging outputs.  If none are defined, or they
are all
  # disabled you will get the default - console output.
  outputs:
  - console:
      enabled: yes
  - file:
      enabled: no
      filename: /var/log/suricata.log
  - syslog:
      enabled: no
      facility: local5
      format: "[%i] <%d> -- "

# Tilera mpipe configuration. for use on Tilera TILE-Gx.
mpipe:

  # Load balancing modes: "static", "dynamic", "sticky", or
"round-robin".
  load-balance: dynamic

  # Number of Packets in each ingress packet queue. Must be
128, 512, 2028 or 65536
  iqueue-packets: 2048

  # List of interfaces we will listen on.
  inputs:
  - interface: xgbe2
  - interface: xgbe3
  - interface: xgbe4


  # Relative weight of memory for packets of each mPipe buffer
size.
  stack:
    size128: 0
    size256: 9
    size512: 0
    size1024: 0
    size1664: 7
    size4096: 0
    size10386: 0
    size16384: 0
```

```
# PF_RING configuration. for use with native PF_RING support
# for more info see http://www.ntop.org/PF_RING.html
pfring:
  - interface: eth0
    # Number of receive threads (>1 will enable experimental
flow pinned
    # runmode)
    threads: 1

    # Default clusterid.  PF_RING will load balance packets
based on flow.
    # All threads/processes that will participate need to have
the same
    # clusterid.
    cluster-id: 99

    # Default PF_RING cluster type. PF_RING can load balance
per flow or per hash.
    # This is only supported in versions of PF_RING > 4.1.1.
    cluster-type: cluster_flow
    # bpf filter for this interface
    #bpf-filter: tcp
    # Choose checksum verification mode for the interface. At
the moment
    # of the capture, some packets may be with an invalid
checksum due to
    # offloading to the network card of the checksum
computation.
    # Possible values are:
    #  - rxonly: only compute checksum for packets received by
network card.
    #  - yes: checksum validation is forced
    #  - no: checksum validation is disabled
    #  - auto: suricata uses a statistical approach to detect
when
    #  checksum off-loading is used. (default)
    # Warning: 'checksum-validation' must be set to yes to
have any validation
    #checksum-checks: auto
  # Second interface
  #- interface: eth1
  #  threads: 3
  #  cluster-id: 93
  #  cluster-type: cluster_flow
  # Put default values here
  - interface: default
    #threads: 2

pcap:
  - interface: eth0
    # On Linux, pcap will try to use mmaped capture and will
use buffer-size
    # as total of memory used by the ring. So set this to
something bigger
```

```
    # than 1% of your bandwidth.
    #buffer-size: 16777216
    #bpf-filter: "tcp and port 25"
    # Choose checksum verification mode for the interface. At
the moment
    # of the capture, some packets may be with an invalid
checksum due to
    # offloading to the network card of the checksum
computation.
    # Possible values are:
    #  - yes: checksum validation is forced
    #  - no: checksum validation is disabled
    #  - auto: suricata uses a statistical approach to detect
when
    #  checksum off-loading is used. (default)
    # Warning: 'checksum-validation' must be set to yes to
have any validation
    #checksum-checks: auto
    # With some accelerator cards using a modified libpcap
(like myricom), you
    # may want to have the same number of capture threads as
the number of capture
    # rings. In this case, set up the threads variable to N to
start N threads
    # listening on the same interface.
    #threads: 16
    # set to no to disable promiscuous mode:
    #promisc: no
    # set snaplen, if not set it defaults to MTU if MTU can be
known
    # via ioctl call and to full capture if not.
    #snaplen: 1518
  # Put default values here
  - interface: default
    #checksum-checks: auto

pcap-file:
  # Possible values are:
  #  - yes: checksum validation is forced
  #  - no: checksum validation is disabled
  #  - auto: suricata uses a statistical approach to detect
when
  #  checksum off-loading is used. (default)
  # Warning: 'checksum-validation' must be set to yes to have
checksum tested
  checksum-checks: auto

# For FreeBSD ipfw(8) divert(4) support.
# Please make sure you have ipfw_load="YES" and
ipdivert_load="YES"
# in /etc/loader.conf or kldload'ing the appropriate kernel
modules.
# Additionally, you need to have an ipfw rule for the engine
to see
```

```
# the packets from ipfw.  For Example:
#
#   ipfw add 100 divert 8000 ip from any to any
#
# The 8000 above should be the same number you passed on the
command
# line, i.e. -d 8000
#
ipfw:

  # Reinject packets at the specified ipfw rule number.  This
config
  # option is the ipfw rule number AT WHICH rule processing
continues
  # in the ipfw processing system after the engine has
finished
  # inspecting the packet for acceptance.  If no rule number
is specified,
  # accepted packets are reinjected at the divert rule which
they entered
  # and IPFW rule processing continues.  No check is done to
verify
  # this will rule makes sense so care must be taken to avoid
loops in ipfw.
  #
  ## The following example tells the engine to reinject
packets
  # back into the ipfw firewall AT rule number 5500:
  #
  # ipfw-reinjection-rule-number: 5500

# Set the default rule path here to search for the files.
# if not set, it will look at the current working dir
default-rule-path: /etc/suricata/rules
rule-files:
 - botcc.rules
 - ciarmy.rules
 - compromised.rules
 - drop.rules
 - dshield.rules
 - emerging-activex.rules
 - emerging-attack_response.rules
 - emerging-chat.rules
 - emerging-current_events.rules
 - emerging-dns.rules
 - emerging-dos.rules
 - emerging-exploit.rules
 - emerging-ftp.rules
 - emerging-games.rules
 - emerging-icmp_info.rules
# - emerging-icmp.rules
 - emerging-imap.rules
 - emerging-inappropriate.rules
 - emerging-malware.rules
```

```
 - emerging-misc.rules
 - emerging-mobile_malware.rules
 - emerging-netbios.rules
 - emerging-p2p.rules
 - emerging-policy.rules
 - emerging-pop3.rules
 - emerging-rpc.rules
 - emerging-scada.rules
 - emerging-scan.rules
 - emerging-shellcode.rules
 - emerging-smtp.rules
 - emerging-snmp.rules
 - emerging-sql.rules
 - emerging-telnet.rules
 - emerging-tftp.rules
 - emerging-trojan.rules
 - emerging-user_agents.rules
 - emerging-voip.rules
 - emerging-web_client.rules
 - emerging-web_server.rules
 - emerging-web_specific_apps.rules
 - emerging-worm.rules
 - tor.rules
 - decoder-events.rules # available in suricata sources under
rules dir
 - stream-events.rules  # available in suricata sources under
rules dir
 - http-events.rules    # available in suricata sources under
rules dir
 - smtp-events.rules    # available in suricata sources under
rules dir
 - dns-events.rules     # available in suricata sources under
rules dir
 - tls-events.rules     # available in suricata sources under
rules dir

classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config

# Holds variables that would be used by the engine.
vars:

  # Holds the address group vars that would be passed in a
Signature.
  # These would be retrieved during the Signature address
parsing stage.
  address-groups:

    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"

    EXTERNAL_NET: "!$HOME_NET"

    HTTP_SERVERS: "$HOME_NET"
```

```
    SMTP_SERVERS: "$HOME_NET"

    SQL_SERVERS: "$HOME_NET"

    DNS_SERVERS: "$HOME_NET"

    TELNET_SERVERS: "$HOME_NET"

    AIM_SERVERS: "$EXTERNAL_NET"

    DNP3_SERVER: "$HOME_NET"

    DNP3_CLIENT: "$HOME_NET"

    MODBUS_CLIENT: "$HOME_NET"

    MODBUS_SERVER: "$HOME_NET"

    ENIP_CLIENT: "$HOME_NET"

    ENIP_SERVER: "$HOME_NET"

  # Holds the port group vars that would be passed in a
Signature.
  # These would be retrieved during the Signature port parsing
stage.
  port-groups:

    HTTP_PORTS: "80"

    SHELLCODE_PORTS: "!80"

    ORACLE_PORTS: 1521

    SSH_PORTS: 22

    DNP3_PORTS: 20000

# Set the order of alerts bassed on actions
# The default order is pass, drop, reject, alert
action-order:
  - pass
  - drop
  - reject
  - alert

# IP Reputation
#reputation-categories-file:
/etc/suricata/iprep/categories.txt
#default-reputation-path: /etc/suricata/iprep
#reputation-files:
# - reputation.list

# Host specific policies for defragmentation and TCP stream
```

```
# reassembly.  The host OS lookup is done using a radix tree,
just
# like a routing table so the most specific entry matches.
host-os-policy:
  # Make the default policy windows.
  windows: [0.0.0.0/0]
  bsd: []
  bsd-right: []
  old-linux: []
  linux: [10.0.0.0/8, 192.168.1.100,
"8762:2352:6241:7245:E000:0000:0000:0000"]
  old-solaris: []
  solaris: ["::1"]
  hpux10: []
  hpux11: []
  irix: []
  macos: []
  vista: []
  windows2k3: []


# Limit for the maximum number of asn1 frames to decode
(default 256)
asn1-max-frames: 256

# When run with the option --engine-analysis, the engine will
read each of
# the parameters below, and print reports for each of the
enabled sections
# and exit.  The reports are printed to a file in the default
log dir
# given by the parameter "default-log-dir", with engine
reporting
# subsection below printing reports in its own report file.
engine-analysis:
  # enables printing reports for fast-pattern for every rule.
  rules-fast-pattern: yes
  # enables printing reports for each rule
  rules: yes

#recursion and match limits for PCRE where supported
pcre:
  match-limit: 3500
  match-limit-recursion: 1500

# Holds details on the app-layer. The protocols section
details each protocol.
# Under each protocol, the default value for detection-enabled
and "
# parsed-enabled is yes, unless specified otherwise.
# Each protocol covers enabling/disabling parsers for all
ipprotos
# the app-layer protocol runs on.  For example "dcerpc" refers
to the tcp
```

```
# version of the protocol as well as the udp version of the
protocol.
# The option "enabled" takes 3 values - "yes", "no",
"detection-only".
# "yes" enables both detection and the parser, "no" disables
both, and
# "detection-only" enables detection only(parser disabled).
app-layer:
  protocols:
    tls:
      enabled: yes
      detection-ports:
        dp: 443

      #no-reassemble: yes
    dcerpc:
      enabled: yes
    ftp:
      enabled: yes
    ssh:
      enabled: yes
    smtp:
      enabled: yes
    imap:
      enabled: detection-only
    msn:
      enabled: detection-only
    smb:
      enabled: yes
      detection-ports:
        dp: 139
    # smb2 detection is disabled internally inside the engine.
    #smb2:
    #  enabled: yes
    dns:
      # memcaps. Globally and per flow/state.
      #global-memcap: 16mb
      #state-memcap: 512kb

      # How many unreplied DNS requests are considered a
flood.
      # If the limit is reached, app-layer-event:dns.flooded;
will match.
      #request-flood: 500

      tcp:
        enabled: yes
        detection-ports:
          dp: 53
      udp:
        enabled: yes
        detection-ports:
          dp: 53
    http:
```

```
      enabled: yes
      # memcap: 64mb


##########################################################################
#############
      # Configure libhtp.
      #
      #
      # default-config:           Used when no server-config
matches
      #   personality:            List of personalities used
by default
      #   request-body-limit:     Limit reassembly of request
body for inspection
      #                           by http_client_body & pcre
/P option.
      #   response-body-limit:    Limit reassembly of response
body for inspection
      #                           by file_data,
http_server_body & pcre /Q option.
      #   double-decode-path:     Double decode path section
of the URI
      #   double-decode-query:    Double decode query section
of the URI
      #
      # server-config:            List of server
configurations to use if address matches
      #   address:                List of ip addresses or
networks for this block
      #   personalitiy:           List of personalities used
by this block
      #   request-body-limit:     Limit reassembly of request
body for inspection
      #                           by http_client_body & pcre
/P option.
      #   response-body-limit:    Limit reassembly of response
body for inspection
      #                           by file_data,
http_server_body & pcre /Q option.
      #   double-decode-path:     Double decode path section
of the URI
      #   double-decode-query:    Double decode query section
of the URI
      #
      #   uri-include-all:        Include all parts of the
URI. By default the
      #                           'scheme', username/password,
hostname and port
      #                           are excluded. Setting this
option to true adds
      #                           all of them to the
normalized uri as inspected
```

```
      #                             by http_uri, urilen, pcre
with /U and the other
      #                             keywords that inspect the
normalized uri.
      #                             Note that this does not
affect http_raw_uri.
      #                             Also, note that including
all was the default in
      #                             1.4 and 2.0beta1.
      #
      #   meta-field-limit:         Hard size limit for request
and response size
      #                             limits. Applies to request
line and headers,
      #                             response line and headers.
Does not apply to
      #                             request or response bodies.
Default is 18k.
      #                             If this limit is reached an
event is raised.
      #
      # Currently Available Personalities:
      #   Minimal
      #   Generic
      #   IDS (default)
      #   IIS_4_0
      #   IIS_5_0
      #   IIS_5_1
      #   IIS_6_0
      #   IIS_7_0
      #   IIS_7_5
      #   Apache_2

##########################################################################
#############
      libhtp:

         default-config:
           personality: IDS

           # Can be specified in kb, mb, gb.  Just a number
indicates
           # it's in bytes.
           request-body-limit: 3072
           response-body-limit: 3072

           # inspection limits
           request-body-minimal-inspect-size: 32kb
           request-body-inspect-window: 4kb
           response-body-minimal-inspect-size: 32kb
           response-body-inspect-window: 4kb
           # Take a random value for inspection sizes around
the specified value.
```

```
            # This lower the risk of some evasion technics but
could lead
            # detection change between runs. It is set to 'yes'
by default.
            #randomize-inspection-sizes: yes
            # If randomize-inspection-sizes is active, the
value of various
            # inspection size will be choosen in the [1 -
range%, 1 + range%]
            # range
            # Default value of randomize-inspection-range is
10.
            #randomize-inspection-range: 10

            # decoding
            double-decode-path: no
            double-decode-query: no

          server-config:

            #- apache:
            #    address: [192.168.1.0/24, 127.0.0.0/8, "::1"]
            #    personality: Apache_2
            #    # Can be specified in kb, mb, gb.  Just a
number indicates
            #    # it's in bytes.
            #    request-body-limit: 4096
            #    response-body-limit: 4096
            #    double-decode-path: no
            #    double-decode-query: no

            #- iis7:
            #    address:
            #      - 192.168.0.0/24
            #      - 192.168.10.0/24
            #    personality: IIS_7_0
            #    # Can be specified in kb, mb, gb.  Just a
number indicates
            #    # it's in bytes.
            #    request-body-limit: 4096
            #    response-body-limit: 4096
            #    double-decode-path: no
            #    double-decode-query: no

# Profiling settings. Only effective if Suricata has been
built with the
# the --enable-profiling configure flag.
#
profiling:
  # Run profiling for every xth packet. The default is 1,
which means we
  # profile every packet. If set to 1000, one packet is
profiled for every
  # 1000 received.
```

```
  #sample-rate: 1000

  # rule profiling
  rules:

    # Profiling can be disabled here, but it will still have a
    # performance impact if compiled in.
    enabled: yes
    filename: rule_perf.log
    append: yes

    # Sort options: ticks, avgticks, checks, matches, maxticks
    sort: avgticks

    # Limit the number of items printed at exit.
    limit: 100

  # per keyword profiling
  keywords:
    enabled: yes
    filename: keyword_perf.log
    append: yes

  # packet profiling
  packets:

    # Profiling can be disabled here, but it will still have a
    # performance impact if compiled in.
    enabled: yes
    filename: packet_stats.log
    append: yes

    # per packet csv output
    csv:

      # Output can be disabled here, but it will still have a
      # performance impact if compiled in.
      enabled: no
      filename: packet_stats.csv

  # profiling of locking. Only available when Suricata was
built with
  # --enable-profiling-locks.
  locks:
    enabled: no
    filename: lock_stats.log
    append: yes

# Suricata core dump configuration. Limits the size of the
core dump file to
# approximately max-dump. The actual core dump size will be a
multiple of the
# page size. Core dumps that would be larger than max-dump are
truncated. On
```

```
# Linux, the actual core dump size may be a few pages larger
than max-dump.
# Setting max-dump to 0 disables core dumping.
# Setting max-dump to 'unlimited' will give the full core dump
file.
# On 32-bit Linux, a max-dump value >= ULONG_MAX may cause the
core dump size
# to be 'unlimited'.

coredump:
  max-dump: unlimited

napatech:
    # The Host Buffer Allowance for all streams
    # (-1 = OFF, 1 - 100 = percentage of the host buffer that
can be held back)
    hba: -1

    # use_all_streams set to "yes" will query the Napatech
service for all configured
    # streams and listen on all of them. When set to "no" the
streams config array
    # will be used.
    use-all-streams: yes

    # The streams to listen on
    streams: [1, 2, 3]

# Includes.  Files included here will be handled as if they
were
# inlined in this configuration file.
#include: include1.yaml
#include: include2.yaml
```

**12. File konfigurasi classification.config**

```
# $Id$
# classification.config taken from Snort 2.8.5.3. Snort is
governed by the GPLv2
#
# The following includes information for prioritizing rules
#
# Each classification includes a shortname, a description, and
a default
# priority for that classification.
#
# This allows alerts to be classified and prioritized.  You
can specify
# what priority each classification has.  Any rule can
override the default
# priority for that rule.
#
```

```
# Here are a few example rules:
#
#   alert TCP any any -> any 80 (msg: "EXPLOIT ntpdx
overflow";
#    dsize: > 128; classtype:attempted-admin; priority:10;
#
#   alert TCP any any -> any 25 (msg:"SMTP expn root";
flags:A+; \
#          content:"expn root"; nocase; classtype:attempted-
recon;)
#
# The first rule will set its type to "attempted-admin" and
override
# the default priority for that type to 10.
#
# The second rule set its type to "attempted-recon" and set
its
# priority to the default for that type.
#

#
# config classification:shortname,short description,priority
#

config classification: not-suspicious,Not Suspicious Traffic,3
config classification: unknown,Unknown Traffic,3
config classification: bad-unknown,Potentially Bad Traffic, 2
config classification: attempted-recon,Attempted Information
Leak,2
config classification: successful-recon-limited,Information
Leak,2
config classification: successful-recon-largescale,Large Scale
Information Leak,2
config classification: attempted-dos,Attempted Denial of
Service,2
config classification: successful-dos,Denial of Service,2
config classification: attempted-user,Attempted User Privilege
Gain,1
config classification: unsuccessful-user,Unsuccessful User
Privilege Gain,1
config classification: successful-user,Successful User
Privilege Gain,1
config classification: attempted-admin,Attempted Administrator
Privilege Gain,1
config classification: successful-admin,Successful
Administrator Privilege Gain,1


# NEW CLASSIFICATIONS
config classification: rpc-portmap-decode,Decode of an RPC
Query,2
config classification: shellcode-detect,Executable code was
detected,1
```

```
config classification: string-detect,A suspicious string was
detected,3
config classification: suspicious-filename-detect,A suspicious
filename was detected,2
config classification: suspicious-login,An attempted login
using a suspicious username was detected,2
config classification: system-call-detect,A system call was
detected,2
config classification: tcp-connection,A TCP connection was
detected,4
config classification: trojan-activity,A Network Trojan was
detected, 1
config classification: unusual-client-port-connection,A client
was using an unusual port,2
config classification: network-scan,Detection of a Network
Scan,3
config classification: denial-of-service,Detection of a Denial
of Service Attack,2
config classification: non-standard-protocol,Detection of a
non-standard protocol or event,2
config classification: protocol-command-decode,Generic
Protocol Command Decode,3
config classification: web-application-activity,access to a
potentially vulnerable web application,2
config classification: web-application-attack,Web Application
Attack,1
config classification: misc-activity,Misc activity,3
config classification: misc-attack,Misc Attack,2
config classification: icmp-event,Generic ICMP event,3
config classification: kickass-porn,SCORE! Get the lotion!,1
config classification: policy-violation,Potential Corporate
Privacy Violation,1
config classification: default-login-attempt,Attempt to login
by a default username and password,2
```

**13. File konfigurasi reference.config**

```
# config reference: system URL

config reference: bugtraq   http://www.securityfocus.com/bid/
config reference: bid      http://www.securityfocus.com/bid/
config reference: cve       http://cve.mitre.org/cgi-
bin/cvename.cgi?name=
config reference: secunia   http://www.secunia.com/advisories/

#whitehats is unfortunately gone
config reference: arachNIDS http://www.whitehats.com/info/IDS

config reference: McAfee    http://vil.nai.com/vil/content/v_
config reference: nessus
http://cgi.nessus.org/plugins/dump.php3?id=
config reference: url       http://
config reference: et        http://doc.emergingthreats.net/
```

```
config reference: etpro     http://doc.emergingthreatpro.com/
config reference: telus     http://

config reference: md5
http://www.threatexpert.com/report.aspx?md5=
```

**14. File konfigurasi pada Siege**

# Updated by Siege 2.70, May-17-2011

# Copyright 2000-2007 by Jeffrey Fulmer, et al.

#

# Siege configuration file -- edit as necessary

# For more information about configuring and running

# this program, visit: http://www.joedog.org/

#

# Variable declarations. You can set variables here

# for use in the directives below. Example:

# PROXY = proxy.joedog.org

# Reference variables inside ${} or $(), example:

# proxy-host = ${PROXY}

# You can also reference ENVIRONMENT variables without

# actually declaring them, example:

# logfile = $(HOME)/var/siege.log

#

# Signify verbose mode, true turns on verbose output

# ex: verbose = true|false

#

```
verbose = true

#
# CSV Verbose format: with this option, you can choose
# to format verbose output in traditional siege format
# or comma separated format. The latter will allow you
# to redirect output to a file for import into a spread
# sheet, i.e., siege > file.csv
# ex: csv = true|false (default false)
#
# csv = true

#
# Full URL verbose format: By default siege displays
# the URL path and not the full URL. With this option,
# you # can instruct siege to show the complete URL.
# ex: fullurl = true|false (default false)
#
# fullurl = true

#
# Display id: in verbose mode, display the siege user
# id associated with the HTTP transaction information
# ex: display-id = true|false
#
```

```
# display-id =

#
# Show logfile location.  By default, siege displays the
# logfile location at the end of every run when logging
# You can turn this message off with this directive.
# ex: show-logfile = false
#
show-logfile = true

#
# Default logging status, true turns logging on.
# ex: logging = true|false
#
logging = true

#
# Logfile, the default siege logfile is $PREFIX/var/siege.log
# This directive allows you to choose an alternative log file.
# Environment variables may be used as shown in the examples:
# ex: logfile = /home/jeff/var/log/siege.log
#     logfile = ${HOME}/var/log/siege.log
#     logfile = ${LOGFILE}
#
# logfile =
```

```
#
# HTTP protocol.  Options HTTP/1.1 and HTTP/1.0.
# Some webservers have broken implementation of the
# 1.1 protocol which skews throughput evaluations.
# If you notice some siege clients hanging for
# extended periods of time, change this to HTTP/1.0
# ex: protocol = HTTP/1.1
#     protocol = HTTP/1.0
#
protocol = HTTP/1.1


#
# Chunked encoding is required by HTTP/1.1 protocol
# but siege allows you to turn it off as desired.
#
# ex: chunked = true
#
chunked = true


#
# Cache revalidation.
# Siege supports cache revalidation for both ETag and
# Last-modified headers. If a copy is still fresh, the
# server responds with 304.
```

```
# HTTP/1.1 200   0.00 secs:    2326 bytes ==> /apache_pb.gif
# HTTP/1.1 304   0.00 secs:       0 bytes ==> /apache_pb.gif
# HTTP/1.1 304   0.00 secs:       0 bytes ==> /apache_pb.gif
#
# ex: cache = true
#
cache = false


#
# Connection directive. Options "close" and "keep-alive"
# Starting with release 2.57b3, siege implements persistent
# connections in accordance to RFC 2068 using both chunked
# encoding and content-length directives to determine the
# page size. To run siege with persistent connections set
# the connection directive to keep-alive. (Default close)
# CAUTION: use the keep-alive directive with care.
# DOUBLE CAUTION: this directive does not work well on HPUX
# TRIPLE CAUTION: don't use keep-alives until further notice
# ex: connection = close
#     connection = keep-alive
#
connection = close


#
# Default number of simulated  concurrent users
```

```
# ex: concurrent = 25
#
concurrent = 15


#
# Default duration of the siege.  The right hand argument has
# a modifier which specifies the time units, H=hours, M=minutes,
# and S=seconds. If a modifier is not specified, then minutes
# are assumed.
# ex: time = 50M
#
# time =


#
# Repetitions.  The length of siege may be specified in client
# reps rather then a time duration.  Instead of specifying a time
# span, you can tell each siege instance to hit the server X number
# of times.  So if you chose 'reps = 20' and you've selected 10
# concurrent users, then siege will hit the server 200 times.
# ex: reps = 20
#
# reps =


#
# Default URLs file, set at configuration time, the default
```

```
# file is PREFIX/etc/urls.txt.  So if you configured siege
# with --prefix=/usr/local then the urls.txt file is installed
# int /usr/local/etc/urls.txt.  Use the "file = " directive to
# configure an alternative URLs file. You may use environment
# variables as shown in the examples below:
# ex: file = /export/home/jdfulmer/MYURLS.txt
#     file = $HOME/etc/urls.txt
#     file = $URLSFILE
#
# file =

#
# Default URL, this is a single URL that you want to test. This
# is usually set at the command line with the -u option.  When
# used, this option overrides the urls.txt (-f FILE/--file=FILE)
# option. You will HAVE to comment this out for in order to use
# the urls.txt file option.
# ex: url = https://shemp.whoohoo.com/docs/index.jsp
#
# url =

#
# Default delay value, see the siege(1) man page.
# This value is used for load testing, it is not used
# for benchmarking.
```

```
# ex: delay = 3
#
delay = 1

#
# Connection timeout value. Set the value in seconds for
# socket connection timeouts. The default value is 30 seconds.
# ex: timeout = 30
#
# timeout =

#
# Session expiration: This directive allows you to delete all
# cookies after you pass through the URLs. This means siege will
# grab a new session with each run through its URLs. The default
# value is false.
# ex: expire-session = true
#
# expire-session =

#
# Failures: This is the number of total connection failures allowed
# before siege aborts. Connection failures (timeouts, socket failures,
# etc.) are combined with 400 and 500 level errors in the final stats,
# but those errors do not count against the abort total.  If you set
```

```
# this total to 10, then siege will abort after ten socket timeouts,
# but it will NOT abort after ten 404s. This is designed to prevent
# a run-away mess on an unattended siege. The default value is 1024
# ex: failures = 50
#
# failures =

#
# Internet simulation. If true, siege clients will hit
# the URLs in the urls.txt file randomly, thereby simulating
# internet usage.  If false, siege will run through the
# urls.txt file in order from first to last and back again.
# ex: internet = true
#
internet = false

#
# Default benchmarking value, If true, there is NO delay
# between server requests, siege runs as fast as the web
# server and the network will let it.  Set this to false
# for load testing.
# ex: benchmark = true
#
benchmark = false
```

```
#
# Set the siege User-Agent to identify yourself at the
# host, the default is: JoeDog/1.00 [en] (X11; I; Siege #.##)
# But that wreaks of corporate techno speak.  Feel free
# to make it more interesting :-)  Since Limey is recovering
# from minor surgery as I write this, I'll dedicate the
# example to him...
# ex: user-agent = Limey The Bulldog
#
# user-agent =

#
# Accept-encoding. This option allows you to specify
# acceptable encodings returned by the server. Use this
# directive to turn on compression. By default we accept
# gzip compression.
#
# ex: accept-encoding = *
#     accept-encoding = gzip
#     accept-encoding = compress;q=0.5;gzip;q=1
accept-encoding = gzip

#
# TURN OFF THAT ANNOYING SPINNER!
# Siege spawns a thread and runs a spinner to entertain you
```

```
# as it collects and computes its stats. If you don't like
# this feature, you may turn it off here.
# ex: spinner = false
#
spinner = true

#
# WWW-Authenticate login. When siege hits a webpage
# that requires basic authentication, it will search its
# logins for authentication which matches the specific realm
# requested by the server. If it finds a match, it will send
# that login information. If it fails to match the realm, it
# will send the default login information. (Default is "all").
# You may configure siege with several logins as long as no
# two realms match. The format for logins is:
# username:password[:realm] where "realm" is optional.
# If you do not supply a realm, then it will default to "all"
# ex: login = jdfulmer:topsecret:Admin
#     login = jeff:supersecret
#
# login =

#
# WWW-Authenticate username and password.  When siege
# hits a webpage that requires authentication, it will
```

```
# send this user name and password to the server.  Note
# this is NOT form based authentication.  You will have
# to construct URLs for that.
# ex: username = jdfulmer
#     password = whoohoo
#
# username =
# password =


#
# ssl-cert
# This optional feature allows you to specify a path to a client
# certificate. It is not neccessary to specify a certificate in
# order to use https. If you don't know why you would want one,
# then you probably don't need this feature.  Use openssl to
# generate a certificate and key with the following command:
#   $ openssl req -nodes -new -days 365 -newkey rsa:1024 \
#                 -keyout key.pem -out cert.pem
# Specify a path to cert.pem as follows:
# ex: ssl-cert = /home/jeff/.certs/cert.pem
#
# ssl-cert =


#
# ssl-key
```

```
# Use this option to specify the key you generated with the command
# above. ex: ssl-key = /home/jeff/.certs/key.pem
# You may actually skip this option and combine both your cert and
# your key in a single file:
#   $ cat key.pem > client.pem
#   $ cat cert.pem >> client.pem
# Now set the path for ssl-cert:
# ex: ssl-cert = /home/jeff/.certs/client.pem
# (in this scenario, you comment out ssl-key)
#
# ssl-key =


#
# ssl-timeout
# This option sets a connection timeout for the ssl library
# ex: ssl-timeout = 30
#
# ssl-timeout =


#
# ssl-ciphers
# You can use this feature to select a specific ssl cipher
# for HTTPs. To view the ones available with your library run
# the following command: openssl ciphers
# ex: ssl-ciphers = EXP-RC4-MD5
```

```
#
# ssl-ciphers =

#
# Login URL. This is the first URL to be hit by every siege
# client. This feature was designed to allow you to login to
# a server and establish a session. It will only be hit once
# so if you need to hit this URL more then once, make sure it
# also appears in your urls.txt file.
#
# ex: login-url = http://eos.haha.com/login.jsp POST name=jeff&pass=foo
#
# Siege versions after 2.69 support multi logins; you can configure
# them with multiple login-url directives. Place each one on a separate
# line. Siege loops through each login then starts again at the beginning
# after it uses the last one. If you have more users than login-urls, then
# siege starts reassigning ones that have already been used.
#
# ex: login-url = http://www.haha.com/login.php?name=homer&pass=whoohoo
#     login-url = http://www.haha.com/login.php?name=marge&pass=ohhomie
#     login-url = http://www.haha.com/login.php?name=bart&pass=eatMyShorts
#
# login-url =

#
```

```
# Proxy protocol. This option allows you to select a proxy
# server stress testing. The proxy will request the URL(s)
# specified by -u"my.url.org" OR from the urls.txt file.
#
# ex: proxy-host = proxy.whoohoo.org
#     proxy-port = 8080
#
# proxy-host =
# proxy-port =


#
# Proxy-Authenticate. When scout hits a proxy server which
# requires username and password authentication, it will this
# username and password to the server. The format is username,
# password and optional realm each separated by a colon. You
# may enter more than one proxy-login as long as each one has
# a different realm. If you do not enter a realm, then scout
# will send that login information to all proxy challenges. If
# you have more than one proxy-login, then scout will attempt
# to match the login to the realm.
# ex: proxy-login: jeff:secret:corporate
#     proxy-login: jeff:whoohoo
#
# proxy-login =
```

```
#
# Redirection support.  This option allows to to control
# whether a Location: hint will be followed.  Most users
# will want to follow redirection information, but sometimes
# it's desired to just get the Location information.
#
# ex: follow-location = false
#
# follow-location =

# Zero-length data.  siege can be configured to disregard
# results in which zero bytes are read after the headers.
# Alternatively, such results can be counted in the final
# tally of outcomes.
#
# ex: zero-data-ok = false
#
# zero-data-ok =

#
# end of siegerc
```

**15. Hasil data Suricata pada post-test pertama**

```
---------------------------------------------------------------
Date: 6/21/2014 -- 02:50:01 (uptime: 0d, 00h 02m 06s)
```

```
-------------------------------------------------------------------
Counter                  | TM Name                 | Value
-------------------------------------------------------------------
capture.kernel_packets    | RxPcapeth01             | 18167
capture.kernel_drops      | RxPcapeth01             | 0
capture.kernel_ifdrops    | RxPcapeth01             | 0
dns.memuse                | RxPcapeth01             | 0
dns.memcap_state          | RxPcapeth01             | 0
dns.memcap_global         | RxPcapeth01             | 0
decoder.pkts              | RxPcapeth01             | 18167
decoder.bytes             | RxPcapeth01             | 2297223
decoder.invalid           | RxPcapeth01             | 0
decoder.ipv4              | RxPcapeth01             | 18165
decoder.ipv6              | RxPcapeth01             | 0
decoder.ethernet          | RxPcapeth01             | 18167
decoder.raw               | RxPcapeth01             | 0
decoder.sll               | RxPcapeth01             | 0
decoder.tcp               | RxPcapeth01             | 18165
decoder.udp               | RxPcapeth01             | 0
decoder.sctp              | RxPcapeth01             | 0
decoder.icmpv4            | RxPcapeth01             | 0
decoder.icmpv6            | RxPcapeth01             | 0
decoder.ppp               | RxPcapeth01             | 0
decoder.pppoe             | RxPcapeth01             | 0
decoder.gre               | RxPcapeth01             | 0
```

decoder.vlan | RxPcapeth01 | 0

decoder.vlan_qinq | RxPcapeth01 | 0

decoder.teredo | RxPcapeth01 | 0

decoder.ipv4_in_ipv6 | RxPcapeth01 | 0

decoder.ipv6_in_ipv6 | RxPcapeth01 | 0

decoder.avg_pkt_size | RxPcapeth01 | 126

decoder.max_pkt_size | RxPcapeth01 | 513

defrag.ipv4.fragments | RxPcapeth01 | 0

defrag.ipv4.reassembled | RxPcapeth01 | 0

defrag.ipv4.timeouts | RxPcapeth01 | 0

defrag.ipv6.fragments | RxPcapeth01 | 0

defrag.ipv6.reassembled | RxPcapeth01 | 0

defrag.ipv6.timeouts | RxPcapeth01 | 0

defrag.max_frag_hits | RxPcapeth01 | 0

tcp.sessions | Detect | 1812

tcp.ssn_memcap_drop | Detect | 0

tcp.pseudo | Detect | 0

tcp.invalid_checksum | Detect | 1419

tcp.no_flow | Detect | 0

tcp.reused_ssn | Detect | 0

tcp.memuse | Detect | 1379000

tcp.syn | Detect | 1812

tcp.synack | Detect | 1711

tcp.rst | Detect | 0

dns.memuse | Detect | 0

dns.memcap_state | Detect | 0

dns.memcap_global | Detect | 0

tcp.segment_memcap_drop | Detect | 0

tcp.stream_depth_reached | Detect | 0

tcp.reassembly_memuse | Detect | 73457184

tcp.reassembly_gap | Detect | 0

http.memuse | Detect | 7001846

http.memcap | Detect | 0

detect.alert | Detect | 1419

flow_mgr.closed_pruned | FlowManagerThread | 0

flow_mgr.new_pruned | FlowManagerThread | 0

flow_mgr.est_pruned | FlowManagerThread | 0

flow.memuse | FlowManagerThread | 6462208

flow.spare | FlowManagerThread | 10000

flow.emerg_mode_entered | FlowManagerThread | 0

flow.emerg_mode_over | FlowManagerThread | 0

---

**16. Hasil data Suricata pada post-test kedua**

---

Date: 6/21/2014 -- 02:53:03 (uptime: 0d, 00h 05m 08s)

---

Counter | TM Name | Value

---

capture.kernel_packets | RxPcapeth01 | 53727

capture.kernel_drops | RxPcapeth01 | 0

| | | |
|---|---|---|
| capture.kernel_ifdrops | RxPcapeth01 | 0 |
| dns.memuse | RxPcapeth01 | 0 |
| dns.memcap_state | RxPcapeth01 | 0 |
| dns.memcap_global | RxPcapeth01 | 0 |
| decoder.pkts | RxPcapeth01 | 54077 |
| decoder.bytes | RxPcapeth01 | 6838422 |
| decoder.invalid | RxPcapeth01 | 0 |
| decoder.ipv4 | RxPcapeth01 | 54075 |
| decoder.ipv6 | RxPcapeth01 | 0 |
| decoder.ethernet | RxPcapeth01 | 54077 |
| decoder.raw | RxPcapeth01 | 0 |
| decoder.sll | RxPcapeth01 | 0 |
| decoder.tcp | RxPcapeth01 | 54075 |
| decoder.udp | RxPcapeth01 | 0 |
| decoder.sctp | RxPcapeth01 | 0 |
| decoder.icmpv4 | RxPcapeth01 | 0 |
| decoder.icmpv6 | RxPcapeth01 | 0 |
| decoder.ppp | RxPcapeth01 | 0 |
| decoder.pppoe | RxPcapeth01 | 0 |
| decoder.gre | RxPcapeth01 | 0 |
| decoder.vlan | RxPcapeth01 | 0 |
| decoder.vlan_qinq | RxPcapeth01 | 0 |
| decoder.teredo | RxPcapeth01 | 0 |
| decoder.ipv4_in_ipv6 | RxPcapeth01 | 0 |
| decoder.ipv6_in_ipv6 | RxPcapeth01 | 0 |

decoder.avg_pkt_size | RxPcapeth01 | 126

decoder.max_pkt_size | RxPcapeth01 | 513

defrag.ipv4.fragments | RxPcapeth01 | 0

defrag.ipv4.reassembled | RxPcapeth01 | 0

defrag.ipv4.timeouts | RxPcapeth01 | 0

defrag.ipv6.fragments | RxPcapeth01 | 0

defrag.ipv6.reassembled | RxPcapeth01 | 0

defrag.ipv6.timeouts | RxPcapeth01 | 0

defrag.max_frag_hits | RxPcapeth01 | 0

tcp.sessions | Detect | 5395

tcp.ssn_memcap_drop | Detect | 0

tcp.pseudo | Detect | 0

tcp.invalid_checksum | Detect | 393

tcp.no_flow | Detect | 0

tcp.reused_ssn | Detect | 0

tcp.memuse | Detect | 2955400

tcp.syn | Detect | 5395

tcp.synack | Detect | 5294

tcp.rst | Detect | 0

dns.memuse | Detect | 0

dns.memcap_state | Detect | 0

dns.memcap_global | Detect | 0

tcp.segment_memcap_drop | Detect | 0

tcp.stream_depth_reached | Detect | 0

tcp.reassembly_memuse | Detect | 73457184

tcp.reassembly_gap | Detect | 0

http.memuse | Detect | 15901772

http.memcap | Detect | 0

detect.alert | Detect | 2857

flow_mgr.closed_pruned | FlowManagerThread | 1751

flow_mgr.new_pruned | FlowManagerThread | 101

flow_mgr.est_pruned | FlowManagerThread | 0

flow.memuse | FlowManagerThread | 6797520

flow.spare | FlowManagerThread | 10009

flow.emerg_mode_entered | FlowManagerThread | 0

flow.emerg_mode_over | FlowManagerThread | 0

---

**17. Hasil data Suricata pada post-test ketiga**

---

---

Date: 6/21/2014 -- 02:57:01 (uptime: 0d, 00h 09m 06s)

---

Counter | TM Name | Value

---

capture.kernel_packets | RxPcapeth01 | 108144

capture.kernel_drops | RxPcapeth01 | 0

capture.kernel_ifdrops | RxPcapeth01 | 0

dns.memuse | RxPcapeth01 | 0

dns.memcap_state | RxPcapeth01 | 0

dns.memcap_global | RxPcapeth01 | 0

| | | |
|---|---|---|
| decoder.pkts | RxPcapeth01 | 108144 |
| decoder.bytes | RxPcapeth01 | 13677585 |
| decoder.invalid | RxPcapeth01 | 0 |
| decoder.ipv4 | RxPcapeth01 | 108142 |
| decoder.ipv6 | RxPcapeth01 | 0 |
| decoder.ethernet | RxPcapeth01 | 108144 |
| decoder.raw | RxPcapeth01 | 0 |
| decoder.sll | RxPcapeth01 | 0 |
| decoder.tcp | RxPcapeth01 | 108142 |
| decoder.udp | RxPcapeth01 | 0 |
| decoder.sctp | RxPcapeth01 | 0 |
| decoder.icmpv4 | RxPcapeth01 | 0 |
| decoder.icmpv6 | RxPcapeth01 | 0 |
| decoder.ppp | RxPcapeth01 | 0 |
| decoder.pppoe | RxPcapeth01 | 0 |
| decoder.gre | RxPcapeth01 | 0 |
| decoder.vlan | RxPcapeth01 | 0 |
| decoder.vlan_qinq | RxPcapeth01 | 0 |
| decoder.teredo | RxPcapeth01 | 0 |
| decoder.ipv4_in_ipv6 | RxPcapeth01 | 0 |
| decoder.ipv6_in_ipv6 | RxPcapeth01 | 0 |
| decoder.avg_pkt_size | RxPcapeth01 | 126 |
| decoder.max_pkt_size | RxPcapeth01 | 513 |
| defrag.ipv4.fragments | RxPcapeth01 | 0 |
| defrag.ipv4.reassembled | RxPcapeth01 | 0 |

defrag.ipv4.timeouts | RxPcapeth01 | 0

defrag.ipv6.fragments | RxPcapeth01 | 0

defrag.ipv6.reassembled | RxPcapeth01 | 0

defrag.ipv6.timeouts | RxPcapeth01 | 0

defrag.max_frag_hits | RxPcapeth01 | 0

tcp.sessions | Detect | 10792

tcp.ssn_memcap_drop | Detect | 0

tcp.pseudo | Detect | 0

tcp.invalid_checksum | Detect | 393

tcp.no_flow | Detect | 0

tcp.reused_ssn | Detect | 0

tcp.memuse | Detect | 3047240

tcp.syn | Detect | 10792

tcp.synack | Detect | 10691

tcp.rst | Detect | 0

dns.memuse | Detect | 0

dns.memcap_state | Detect | 0

dns.memcap_global | Detect | 0

tcp.segment_memcap_drop | Detect | 0

tcp.stream_depth_reached | Detect | 0

tcp.reassembly_memuse | Detect | 73457184

tcp.reassembly_gap | Detect | 0

http.memuse | Detect | 16585349

http.memcap | Detect | 0

detect.alert | Detect | 6038

flow_mgr.closed_pruned | FlowManagerThread | 7091

flow_mgr.new_pruned | FlowManagerThread | 101

flow_mgr.est_pruned | FlowManagerThread | 0

flow.memuse | FlowManagerThread | 6821760

flow.spare | FlowManagerThread | 10033

flow.emerg_mode_entered | FlowManagerThread | 0

flow.emerg_mode_over | FlowManagerThread | 0

---

**18. Hasil data Suricata pada post-test ke-empat**

---\

Date: 6/21/2014 -- 03:02:02 (uptime: 0d, 00h 14m 07s)

---

Counter | TM Name | Value

---

capture.kernel_packets | RxPcapeth01 | 180833

capture.kernel_drops | RxPcapeth01 | 0

capture.kernel_ifdrops | RxPcapeth01 | 0

dns.memuse | RxPcapeth01 | 0

dns.memcap_state | RxPcapeth01 | 0

dns.memcap_global | RxPcapeth01 | 0

decoder.pkts | RxPcapeth01 | 181131

decoder.bytes | RxPcapeth01 | 22909884

decoder.invalid | RxPcapeth01 | 0

decoder.ipv4 | RxPcapeth01 | 181129

decoder.ipv6 | RxPcapeth01 | 0

```
decoder.ethernet          | RxPcapeth01             | 181131
decoder.raw               | RxPcapeth01             | 0
decoder.sll               | RxPcapeth01             | 0
decoder.tcp               | RxPcapeth01             | 181129
decoder.udp               | RxPcapeth01             | 0
decoder.sctp              | RxPcapeth01             | 0
decoder.icmpv4            | RxPcapeth01             | 0
decoder.icmpv6            | RxPcapeth01             | 0
decoder.ppp               | RxPcapeth01             | 0
decoder.pppoe             | RxPcapeth01             | 0
decoder.gre               | RxPcapeth01             | 0
decoder.vlan              | RxPcapeth01             | 0
decoder.vlan_qinq         | RxPcapeth01             | 0
decoder.teredo            | RxPcapeth01             | 0
decoder.ipv4_in_ipv6      | RxPcapeth01             | 0
decoder.ipv6_in_ipv6      | RxPcapeth01             | 0
decoder.avg_pkt_size      | RxPcapeth01             | 126
decoder.max_pkt_size      | RxPcapeth01             | 513
defrag.ipv4.fragments     | RxPcapeth01             | 0
defrag.ipv4.reassembled   | RxPcapeth01             | 0
defrag.ipv4.timeouts      | RxPcapeth01             | 0
defrag.ipv6.fragments     | RxPcapeth01             | 0
defrag.ipv6.reassembled   | RxPcapeth01             | 0
defrag.ipv6.timeouts      | RxPcapeth01             | 0
defrag.max_frag_hits      | RxPcapeth01             | 0
```

tcp.sessions | Detect | 18078

tcp.ssn_memcap_drop | Detect | 0

tcp.pseudo | Detect | 0

tcp.invalid_checksum | Detect | 393

tcp.no_flow | Detect | 0

tcp.reused_ssn | Detect | 0

tcp.memuse | Detect | 3033240

tcp.syn | Detect | 18078

tcp.synack | Detect | 17977

tcp.rst | Detect | 0

dns.memuse | Detect | 0

dns.memcap_state | Detect | 0

dns.memcap_global | Detect | 0

tcp.segment_memcap_drop | Detect | 0

tcp.stream_depth_reached | Detect | 0

tcp.reassembly_memuse | Detect | 73457184

tcp.reassembly_gap | Detect | 0

http.memuse | Detect | 16518188

http.memcap | Detect | 0

detect.alert | Detect | 9724

flow_mgr.closed_pruned | FlowManagerThread | 14426

flow_mgr.new_pruned | FlowManagerThread | 101

flow_mgr.est_pruned | FlowManagerThread | 0

flow.memuse | FlowManagerThread | 6823156

flow.spare | FlowManagerThread | 10038

flow.emerg_mode_entered | FlowManagerThread | 0

flow.emerg_mode_over | FlowManagerThread | 0

---

**19. Hasil data Suricata pada Pengujian post-test kelima**

---

---

Date: 6/21/2014 -- 03:07:59 (uptime: 0d, 00h 20m 04s)

---

Counter | TM Name | Value

---

capture.kernel_packets | RxPcapeth01 | 269891

capture.kernel_drops | RxPcapeth01 | 0

capture.kernel_ifdrops | RxPcapeth01 | 0

dns.memuse | RxPcapeth01 | 0

dns.memcap_state | RxPcapeth01 | 0

dns.memcap_global | RxPcapeth01 | 0

decoder.pkts | RxPcapeth01 | 270130

decoder.bytes | RxPcapeth01 | 34166310

decoder.invalid | RxPcapeth01 | 0

decoder.ipv4 | RxPcapeth01 | 270128

decoder.ipv6 | RxPcapeth01 | 0

decoder.ethernet | RxPcapeth01 | 270130

decoder.raw | RxPcapeth01 | 0

decoder.sll | RxPcapeth01 | 0

decoder.tcp | RxPcapeth01 | 270128

decoder.udp | RxPcapeth01 | 0

decoder.sctp | RxPcapeth01 | 0

decoder.icmpv4 | RxPcapeth01 | 0

decoder.icmpv6 | RxPcapeth01 | 0

decoder.ppp | RxPcapeth01 | 0

decoder.pppoe | RxPcapeth01 | 0

decoder.gre | RxPcapeth01 | 0

decoder.vlan | RxPcapeth01 | 0

decoder.vlan_qinq | RxPcapeth01 | 0

decoder.teredo | RxPcapeth01 | 0

decoder.ipv4_in_ipv6 | RxPcapeth01 | 0

decoder.ipv6_in_ipv6 | RxPcapeth01 | 0

decoder.avg_pkt_size | RxPcapeth01 | 126

decoder.max_pkt_size | RxPcapeth01 | 513

defrag.ipv4.fragments | RxPcapeth01 | 0

defrag.ipv4.reassembled | RxPcapeth01 | 0

defrag.ipv4.timeouts | RxPcapeth01 | 0

defrag.ipv6.fragments | RxPcapeth01 | 0

defrag.ipv6.reassembled | RxPcapeth01 | 0

defrag.ipv6.timeouts | RxPcapeth01 | 0

defrag.max_frag_hits | RxPcapeth01 | 0

tcp.sessions | Detect | 26960

tcp.ssn_memcap_drop | Detect | 0

tcp.pseudo | Detect | 0

tcp.invalid_checksum | Detect | 393

tcp.no_flow | Detect | 0

tcp.reused_ssn | Detect | 0

tcp.memuse | Detect | 3043460

tcp.syn | Detect | 26960

tcp.synack | Detect | 26859

tcp.rst | Detect | 0

dns.memuse | Detect | 0

dns.memcap_state | Detect | 0

dns.memcap_global | Detect | 0

tcp.segment_memcap_drop | Detect | 0

tcp.stream_depth_reached | Detect | 0

tcp.reassembly_memuse | Detect | 73457184

tcp.reassembly_gap | Detect | 0

http.memuse | Detect | 16638887

http.memcap | Detect | 0

detect.alert | Detect | 16705

flow_mgr.closed_pruned | FlowManagerThread | 23281

flow_mgr.new_pruned | FlowManagerThread | 101

flow_mgr.est_pruned | FlowManagerThread | 0

flow.memuse | FlowManagerThread | 6837176

flow.spare | FlowManagerThread | 10032

flow.emerg_mode_entered | FlowManagerThread | 0

flow.emerg_mode_over | FlowManagerThread | 0

**20. Instalasi Suricata**

Adapun instalasi Suricata yang telah dilakukan adalah sebagai berikut :

- Install environtment

  Untuk dapat menjalankan mesin Suricata, terlebih dahulu dipersiapkan lingkungan kerja bagi Suricata.

  $ **sudo apt-get install apache2 php5 mysql-server php5-mysql**

- Install pre-requisite

  # **apt-get install libpcre3 libpcre3-dbg libpcre3-dev **
  **build-essential autoconf automake libtool libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev zlib1g zlib1g-dev pkg-config**

  jangan lupa untuk menginstall interpreter Python yaitu dengan cara

  **# apt-get install python**

- Download dan Install Suricata

  **$ wget http://www.openinfosecfoundation.org/download/suricata-1.0.2.tar.gz**
  **$ tar xzvf suricata-1.0.2.tar.gz**
  **$ cd suricata-1.0.2/**
  **$ ./configure**
  **$ sudo mkdir /var/log/suricata/**
  **$ make**
  **$ sudo make install**

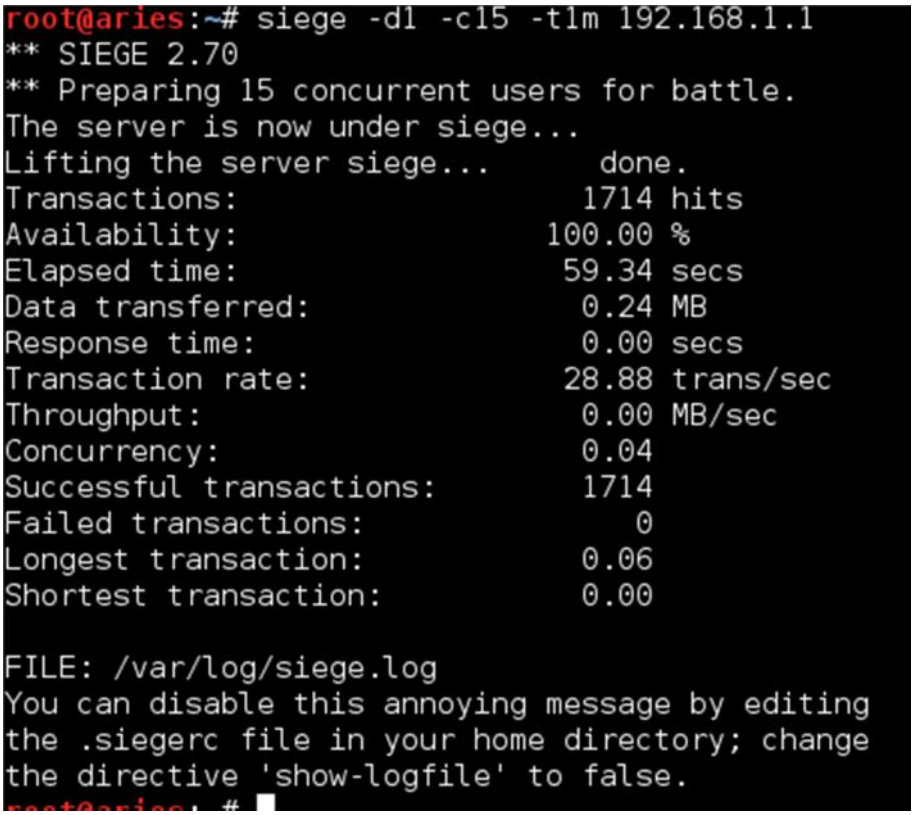
  Untuk dapat beroperasi, Suricata membutuhkan file konfigurasi. Oleh karena itu salin file classification.config, reference.config serta suricata.yaml ke folder /etc/suricata.

- Setting rules
  Download rules IDS dari Emerging Threat lalu ekstrak file ke folder /etc/suricata.

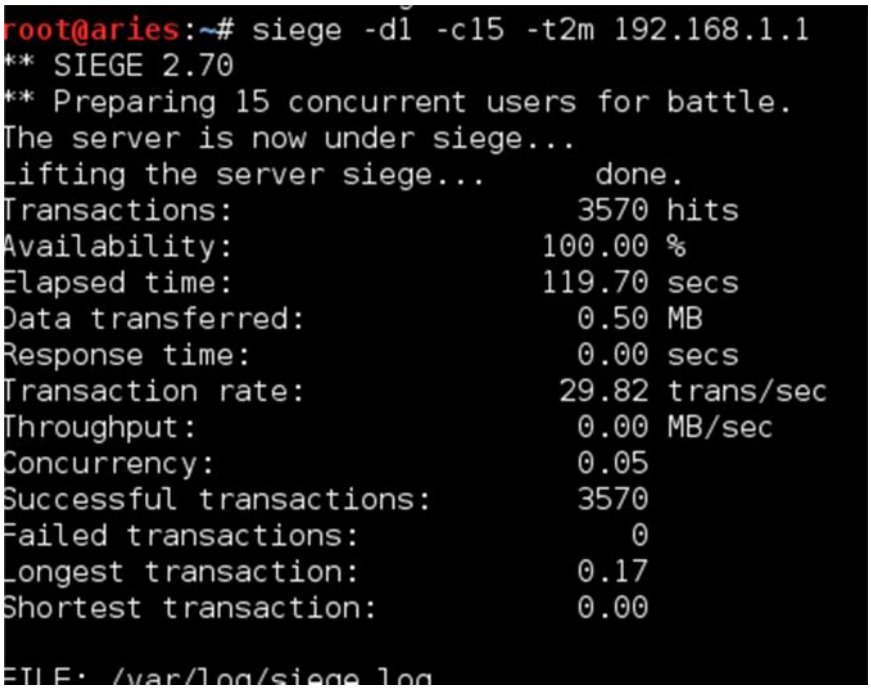
Untuk menjalankan Suricata IDS dapat dilakukan dari terminal Linux dengan memasukkan perintah sebagai berikut :

#**/usr/bin/suricata -c /etc/suricata/suricata.yaml -i eth1**

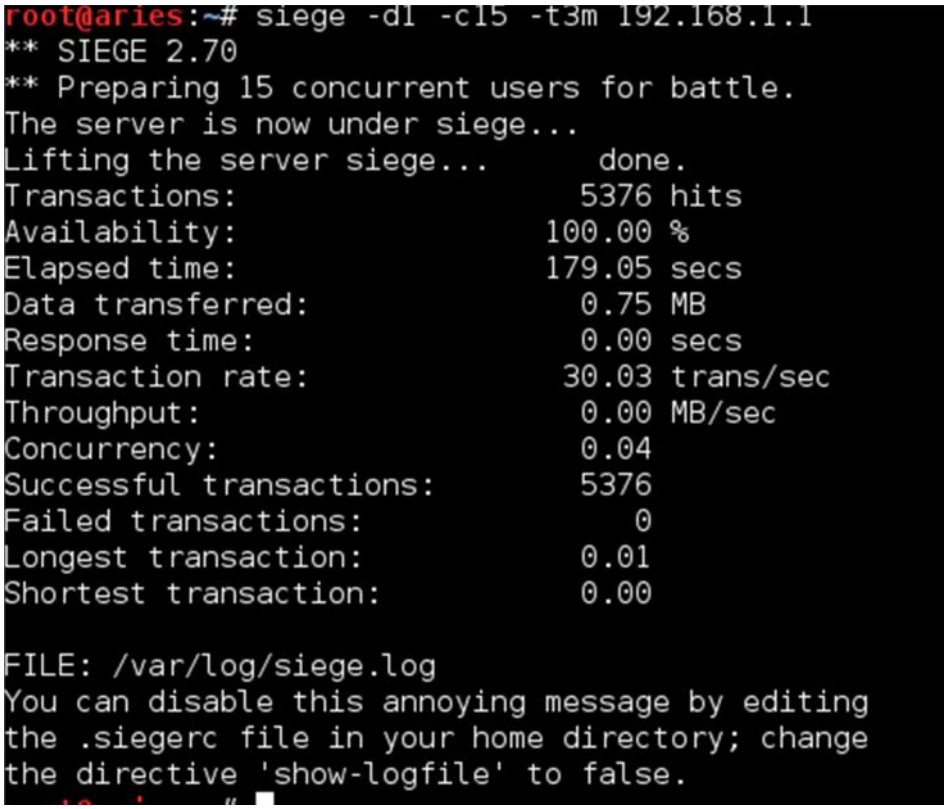
**21. Perintah pengujian pertama pre-test**



**22. Perintah pengujian kedua pre-test**




**23. Perintah pengujian ketiga pre-test**



**24. Perintah pengujian keempat pre-test**

```
root@aries:~# siege -d1 -c15 -t4m 192.168.1.1
** SIEGE 2.70
** Preparing 15 concurrent users for battle.
The server is now under siege...
Lifting the server siege...      done.
Transactions:                   7207 hits
Availability:                 100.00 %
Elapsed time:                 239.24 secs
Data transferred:               1.00 MB
Response time:                  0.00 secs
Transaction rate:              30.12 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                    0.04
Successful transactions:        7207
Failed transactions:               0
Longest transaction:            0.01
Shortest transaction:           0.00

FILE: /var/log/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
root@aries:~#
```

**25. Perintah pengujian kelima pre-test**

```
root@aries:~# siege -d1 -c15 -t5m 192.168.1.1
** SIEGE 2.70
** Preparing 15 concurrent users for battle.
The server is now under siege...
Lifting the server siege...      done.
Transactions:                   8851 hits
Availability:                 100.00 %
Elapsed time:                 299.88 secs
Data transferred:               1.23 MB
Response time:                  0.00 secs
Transaction rate:              29.52 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                    0.04
Successful transactions:        8851
Failed transactions:               0
Longest transaction:            0.98
Shortest transaction:           0.00

FILE: /var/log/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
root@aries:~#
```

**26. Perintah pengujian pertama post-test**

```
root@aries:~# siege -d1 -c15 -t1m 192.168.1.1
** SIEGE 2.70
** Preparing 15 concurrent users for battle.
The server is now under siege...
Lifting the server siege...      done.
Transactions:                   1843 hits
Availability:                 100.00 %
Elapsed time:                  59.26 secs
Data transferred:               0.26 MB
Response time:                  0.00 secs
Transaction rate:              31.10 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                    0.04
Successful transactions:        1843
Failed transactions:               0
Longest transaction:            0.01
Shortest transaction:           0.00

FILE: /var/log/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
root@aries:~#
```

**27. Perintah pengujian kedua post-test**

```
root@aries:~# siege -d1 -c15 -t2m 192.168.1.1
** SIEGE 2.70
** Preparing 15 concurrent users for battle.
The server is now under siege...
Lifting the server siege...      done.
Transactions:                   3552 hits
Availability:                 100.00 %
Elapsed time:                 119.24 secs
Data transferred:               0.49 MB
Response time:                  0.00 secs
Transaction rate:              29.79 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                    0.04
Successful transactions:        3552
Failed transactions:               0
Longest transaction:            0.01
Shortest transaction:           0.00

FILE: /var/log/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
root@aries:~#
```

**28. Perintah pengujian ketiga post-test**

```
root@aries:~# siege -d1 -c15 -t3m 192.168.1.1
** SIEGE 2.70
** Preparing 15 concurrent users for battle.
The server is now under siege...
Lifting the server siege...      done.
Transactions:                   5442 hits
Availability:                 100.00 %
Elapsed time:                 179.77 secs
Data transferred:               0.76 MB
Response time:                  0.00 secs
Transaction rate:              30.27 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                    0.05
Successful transactions:        5442
Failed transactions:               0
Longest transaction:            0.21
Shortest transaction:           0.00

FILE: /var/log/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
root@aries:~#
```

**29. Perintah pengujian keempat post-test**

```
root@aries:~# siege -d1 -c15 -t4m 192.168.1.1
** SIEGE 2.70
** Preparing 15 concurrent users for battle.
The server is now under siege...
Lifting the server siege...      done.
Transactions:                   7241 hits
Availability:                 100.00 %
Elapsed time:                 239.21 secs
Data transferred:               1.01 MB
Response time:                  0.00 secs
Transaction rate:              30.27 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                    0.04
Successful transactions:        7241
Failed transactions:               0
Longest transaction:            0.96
Shortest transaction:           0.00

FILE: /var/log/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
root@aries:~#
```

**30. Perintah pengujian kelima post-test**