

LAPORAN TUGAS BESAR PERETASAN ETIS

CYBER GUARDIAN AI: SISTEM DETEKSI DAN RESPON ANCAMAN

BERBASIS KECERDASAN BUATAN



Dosen Pengampu:

Wilfridus Handaya. ST., M. Cs.

Kelompok 4:

Valencia Ussi	(230712435)
Bernardus Anggi	(230712442)
Jessica M. Pratami	(230712444)
Feleciatus Ian	(230712446)
Steven J. Nainggolan	(230712558)

FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ATMA JAYA YOGYAKARTA
TAHUN 2025/2026

DAFTAR ISI

DAFTAR ISI	ii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Tujuan Penelitian	1
1.3 Rumusan Masalah	2
1.4 Batasan Penelitian	2
BAB II METODOLOGI.....	3
2.1 Implementasi Suricata	3
2.2 Pembangunan <i>Guardian AI Dashboard</i>	3
2.3 Integrasi Sistem <i>Guardian AI</i>	3
2.4 Topologi Sistem.....	3
2.5 Alur Program.....	5
BAB III IMPLEMENTASI, HASIL, DAN ANALISIS	8
3.1. Aktivasi <i>Guardian AI</i>	8
3.2. Menjalankan Suricata.....	10
3.3. Log Traffic pada Suricata.....	11
3.4. Uji Serangan pada <i>Guardian Ai</i>	11
3.5. Simulasi Serangan	12

BAB I

PENDAHULUAN

1.1 Latar Belakang

Keamanan jaringan merupakan aspek yang sangat krusial dalam sistem komputer modern. Lingkungan digital yang selalu terhubung dengan internet membuat berbagai bentuk ancaman seperti *scanning*, *brute force*, *malware traffic*, dan eksplorasi kerentanan dapat muncul kapan saja. Serangan-serangan ini tidak hanya dapat mengganggu stabilitas jaringan, tetapi juga dapat menyebabkan kebocoran data maupun hilangnya integritas sistem.

Untuk mengidentifikasi aktivitas mencurigakan tersebut, diperlukan sebuah *Intrusion Detection System* (IDS) yang mampu melakukan pemantauan secara real time. Suricata merupakan salah satu IDS *open-source* yang banyak digunakan karena memiliki kemampuan untuk menganalisis trafik jaringan dan menghasilkan log rinci melalui file eve.json. Namun, proses membaca log ini secara manual tidaklah mudah karena formatnya yang kompleks dan jumlah event yang sangat besar.

Dari permasalahan tersebut, dikembangkanlah Guardian AI Dashboard, sebuah aplikasi yang berjalan di Ubuntu Desktop dan dibuat untuk mempermudah proses analisis log Suricata. Aplikasi ini mampu menampilkan live alert, menampilkan jumlah event, mengelompokkan rule alert, dan menambahkan hasil deteksi anomali berbasis AI dalam bentuk dashboard interaktif. Dengan adanya dashboard ini, proses pemantauan jaringan menjadi lebih mudah, lebih cepat, dan lebih jelas terutama dalam konteks pembelajaran maupun presentasi.

1.2 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. Membangun lingkungan simulasi keamanan jaringan menggunakan Suricata sebagai IDS dan Guardian AI Dashboard sebagai platform visualisasi.
2. Mengamati bagaimana aktivitas mencurigakan dicatat dalam log Suricata dan ditampilkan secara real time melalui dashboard.
3. Melatih kemampuan dalam merancang topologi jaringan untuk skenario uji keamanan dan peretasan etis.

4. Mengevaluasi efektivitas Guardian AI dalam mendeteksi pola anomali dan serangan berbasis rule maupun AI.
5. Menghubungkan hasil analisis log Suricata dengan deteksi visual pada dashboard untuk mendapatkan gambaran lengkap dari serangan.

1.3 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah:

1. Bagaimana membangun lingkungan uji yang aman untuk mensimulasikan serangan jaringan berbasis Suricata dan Guardian AI.
2. Bagaimana proses integrasi antara Suricata, log eve.json, dan Guardian AI agar alert dapat muncul secara otomatis.
3. Bagaimana dashboard dapat membantu proses analisis *traffic* secara *real time* pada saat serangan berlangsung.
4. Sejauh mana Guardian AI mampu memberikan gambaran menyeluruh tentang pola serangan dan aktivitas mencurigakan.

1.4 Batasan Penelitian

Penelitian ini dibatasi pada:

1. Lingkungan dibangun pada jaringan lokal menggunakan virtualisasi (VirtualBox).
2. Sistem hanya menggunakan Suricata sebagai IDS utama.
3. Jenis serangan yang diuji terbatas pada *port scanning*, *brute force*, dan serangan web sederhana.
4. Guardian AI digunakan sebatas alat monitoring visual tanpa respons otomatis.
5. Analisis difokuskan pada trafik yang terekam melalui log eve.json.

BAB II

METODOLOGI

Tujuan utama dari proyek ini adalah membangun sistem monitoring jaringan yang mengintegrasikan Suricata IDS dengan *dashboard real time* berbasis FastAPI. Metodologi yang digunakan meliputi:

2.1 Implementasi Suricata

Suricata diinstal pada Ubuntu Desktop, kemudian dilakukan konfigurasi untuk mengaktifkan *output log eve.json* secara lengkap. File eve.json menjadi sumber utama seluruh data yang akan dianalisis oleh *dashboard*.

2.2 Pembangunan Guardian AI Dashboard

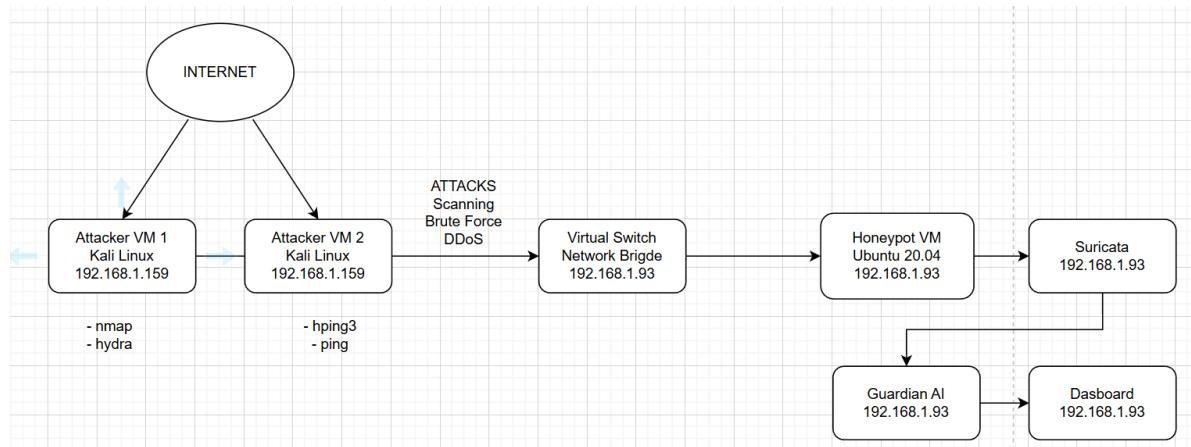
Dashboard dibangun menggunakan FastAPI sebagai backend dan Chart.js sebagai visualisasi frontend. Data dari eve.json diproses menggunakan script Python untuk menghasilkan:

- Live alert (menggunakan Server-Sent Events / SSE),
- Perhitungan jumlah total event,
- Pengelompokan rule alert,
- Deteksi anomali berbasis machine learning.

2.3 Integrasi Sistem Guardian AI

Seluruh komponen kemudian disatukan dalam satu platform yang disebut Cyber Guardian AI, yang mampu bekerja secara real time dan memberikan gambaran menyeluruh kondisi trafik jaringan.

2.4 Topologi Sistem



Komponen Sistem

Attacker VM 1

- OS: Kali Linux
- IP: **192.168.1.159** (*disarankan beda dengan VM2, mis. 192.168.1.158*)
- Tools: **nmap**, **hydra**
- Fungsi: mensimulasikan *port scanning & brute-force* ke target

Attacker VM 2

- OS: Kali Linux
- IP: **192.168.1.159** (*gunakan IP berbeda di implementasi nyata*)
- Tools: **hping3**, **ping**
- Fungsi: mensimulasikan DoS/flood & ICMP ke target

Virtual Switch / Network Bridge

- Peran: jembatan trafik dari attacker ke segmen target
- IP/Manajemen: **192.168.1.93**
- Fitur: *port mirroring/SPAN* agar Suricata & Wireshark menerima salinan paket

Honeypot VM (Target)

- OS: Ubuntu 20.04
- IP: **192.168.1.93**
- Layanan: Cowrie (SSH/Telnet), Dionaea (*opsional*)
- Fungsi: mencatat IP, kredensial, dan perintah penyerang; meneruskan log ke analitik

Suricata (IDS)

- Lokasi: VM/host terpisah atau co-located
- Input: salinan trafik dari port mirror / AF-PACKET
- Output: EVE JSON (*alerts, flows, http, dns, tls*)
- Fungsi: deteksi *signature/anomali* untuk serangan jaringan

Guardian AI (Korelasi & Analitik)

- Input: log Cowrie/Dionaea + event Suricata
- Fungsi: korelasi, skoring risiko, ringkasan insiden
- Output: alert terstruktur ke Dashboard

Dashboard (Monitoring)

- Opsi: Kibana / Grafana (sesuaikan)
- IP: 192.168.1.93
- Fungsi: visualisasi alert & statistik (top attacker IP, top username, jenis serangan, timeline)

2.5 Alur Program

1. Trafik Jaringan Masuk ke Sistem

Semua aktivitas jaringan (*flow, request, response, port scan, upload/download*, dll.) melewati interface yang dipantau oleh Suricata. Suricata bekerja sebagai sensor utama yang membaca packet secara realtime.

Output utama: paket → event Suricata

2. Suricata Mencatat *Event* ke dalam eve.json

Suricata mengubah setiap paket menjadi event JSON, lalu menyimpannya ke:

/var/log/suricata/eve.json

Tipe *event* yang dicatat:

- alert → rule Suricata terpicu
- flow → aliran trafik (bytes toserver/toclient, dest_port, proto)
dns, http, tls, ssh (jika diaktifkan)

Output: file eve.json berisi *event-event* jaringan.

3. Aplikasi Guardian AI Menjalankan Tailer

Dashboard menjalankan background thread: tail_file() yang berfungsi seperti tail -f, yaitu: Membaca eve.json dari akhir file, Mengambil event baru secara terus-menerus Mengelompokkan *event* menjadi *event alert*, *event flow*, atau *event* lain yang diabaikan
Output: event diteruskan ke modul analisis.

4. Pemrosesan Event Rule Alert

Setiap kali Suricata memunculkan alert, Dashboard menambahkan alert ke statistik, melakukan klasifikasi serangan berdasarkan signature, menampilkan alert ke dashboard melalui SSE

Jenis serangan yang dikenali:

- Port Scan
- Brute Force SSH
- DoS / DDoS
- SQL Injection
- DNS Tunneling
- dll.

Output: Rule Alerts tampil di tabel “Live Alerts”.

5. Pemrosesan Event Flow untuk Modul AI

Event flow digunakan oleh modul AI untuk membentuk vektor fitur:[bytes_toserver, bytes_toclient, dest_port, proto]. Menyimpan flow ke buffer, Jika buffer flow mencapai batas awal (misal 20 atau 50 *flow*), AI melakukan training awal untuk memahami pola trafik normal.

Output: Model AI terbentuk (*baseline normal traffic*).

6. Deteksi Anomaly Menggunakan AI

Setelah model AI terlatih:

Setiap *flow* baru dihitung skor *anomaly*-nya menggunakan *Isolation Forest*. Jika skor lebih rendah dari *threshold* → dianggap anomali. Jika *anomaly* valid → disimpan sebagai AI Alert

AI Summary ikut diperbarui berdasarkan:

- jumlah *anomaly*
- pola *source-destination*
- port yang diserang
- pola upload/download besar

AI dapat mendekripsi pola seperti:

- Trafik upload besar (exfiltration)
- Trafik download besar
- Flood ringan yang tidak memicu rule
- Pola port tertentu yang tidak muncul pada trafik normal

Output: AI Alerts tampil di dashboard.

7. Event dikirim ke Frontend melalui SSE

Dashboard menggunakan Server-Sent Events pada endpoint /stream. SSE mengirim data secara realtime:

- stats (*events, rules, anomaly*)
- *rule alerts*
- *AI alerts*
- *summary* terbaru
- *Frontend* langsung memperbarui tampilan tanpa *reload*.

8. Frontend Menampilkan Data ke Dashboard

Frontend akan:

- Meng-update kartu statistik (*Total Events, Rule Alerts, AI Anomalies*)
- Menampilkan alert baru di tabel *Live Alerts*
- Meng-update grafik cumulative *events*
- Meng-update AI *Summary* setiap kali stats berubah

Output: User melihat semua aktivitas jaringan secara realtime.

9. Fitur Tambahan

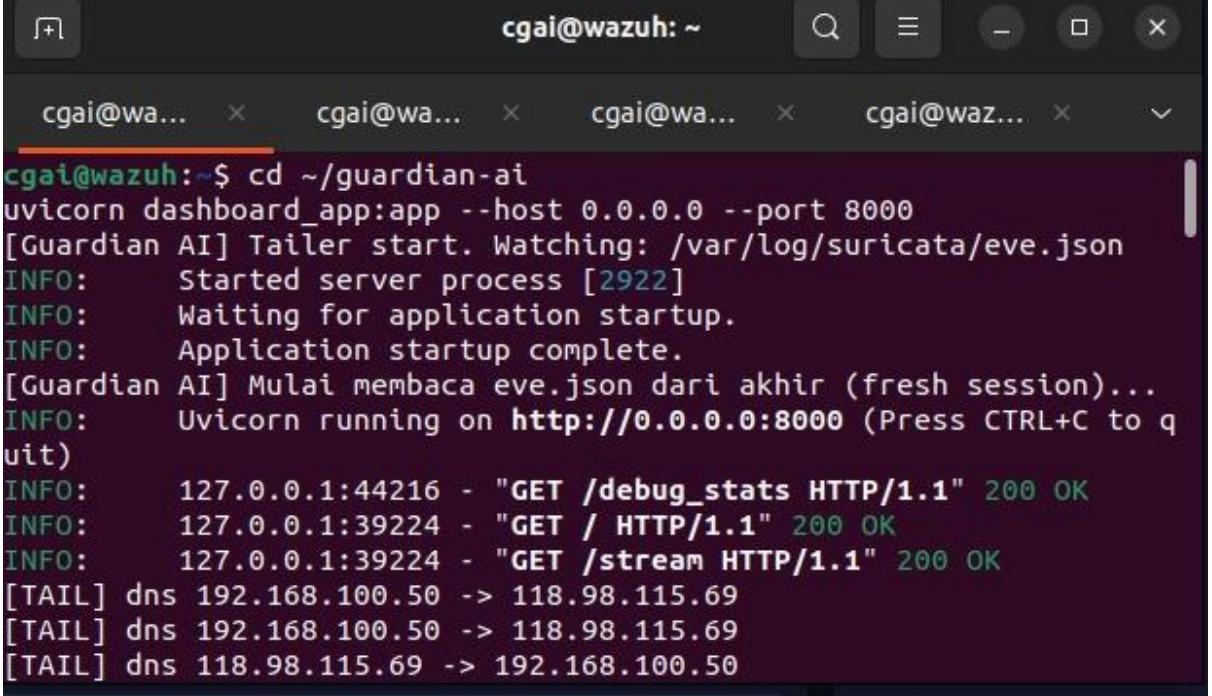
Beberapa fungsi pendukung:

- Reset Counter → menghapus data untuk pengujian ulang
- Live Mode ON/OFF
- ON = semua *anomaly*
- OFF = hanya *anomaly* berat
- Export CSV → mengambil semua alert untuk laporan

BAB III

IMPLEMENTASI, HASIL, DAN ANALISIS

3.1. Aktivasi Guardian AI



The screenshot shows a terminal window titled "cgai@wazuh: ~". It contains four tabs, all labeled "cgai@wa...". The active tab displays the command "cd ~/guardian-ai" followed by the output of the "uvicorn dashboard_app:app --host 0.0.0.0 --port 8000" command. The output includes logs from the "Guardian AI" Tailer, which starts a server process (2922), waits for startup, and then indicates application startup complete. It also shows a fresh session reading "eve.json" and Uvicorn running on "http://0.0.0.0:8000". Subsequent log entries show various HTTP requests (GET /debug_stats, GET /, GET /stream) being handled successfully (200 OK). The Tailer log shows DNS queries between 192.168.100.50 and 118.98.115.69.

```
cgai@wazuh:~$ cd ~/guardian-ai
uvicorn dashboard_app:app --host 0.0.0.0 --port 8000
[Guardian AI] Tailer start. Watching: /var/log/suricata/eve.json
INFO:     Started server process [2922]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
[Guardian AI] Mulai membaca eve.json dari akhir (fresh session)...
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     127.0.0.1:44216 - "GET /debug_stats HTTP/1.1" 200 OK
INFO:     127.0.0.1:39224 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:39224 - "GET /stream HTTP/1.1" 200 OK
[TAIL] dns 192.168.100.50 -> 118.98.115.69
[TAIL] dns 192.168.100.50 -> 118.98.115.69
[TAIL] dns 118.98.115.69 -> 192.168.100.50
```

Screenshot ini bisa kamu pakai sebagai bukti implementasi bahwa:

- Server Guardian AI berhasil berjalan.
- Integrasi dengan Suricata dan dashboard web sudah menyatu.
- Mekanisme real-time log streaming ([TAIL]) berjalan seperti desain pada dua dokumen yang kita analisis.

Pada gambar terminal tersebut terlihat bahwa perintah pertama yang dijalankan adalah `cd ~/guardian-ai` yang menunjukkan bahwa pengguna masuk ke direktori proyek utama sistem Guardian AI. Setelah itu, perintah `uvicorn dashboard_app:app --host 0.0.0.0 --port 8000` digunakan untuk menjalankan web server FastAPI menggunakan Uvicorn. Perintah tersebut memuat aplikasi `dashboard_app:app`, mengaktifkannya pada seluruh alamat IP yang tersedia (0.0.0.0), dan membuka layanan pada port 8000. Proses ini memastikan bahwa dashboard Guardian AI dapat diakses tidak hanya dari localhost tetapi juga dari perangkat lain dalam jaringan yang sama. Dengan demikian, tahap ini menandai dimulainya komponen Dashboard & Observability yang menjadi antarmuka utama pengguna untuk memantau event keamanan secara real time.

Setelah server dijalankan, terminal menampilkan berbagai pesan log awal yang menjadi indikasi bahwa pipeline Guardian AI telah aktif. Log [Guardian AI] Tailer start. Watching: /var/log/suricata/eve.json menunjukkan bahwa thread tailer mulai bekerja dengan memantau file log eve.json yang berasal dari Suricata. Kemudian muncul pesan seperti Started server process [2922], Waiting for application startup, dan Application startup complete yang mengonfirmasi bahwa Uvicorn berhasil memulai servernya tanpa kendala. Pesan-pesan ini menandakan bahwa seluruh komponen backend web telah terinisialisasi dengan sempurna dan siap menerima koneksi dari dashboard pengguna.

Salah satu bagian penting dari proses ini adalah log dengan pesan [Guardian AI] Mulai membaca eve.json dari akhir (fresh session).... Pesan ini menunjukkan bahwa tailer Guardian AI tidak membaca seluruh isi file eve.json dari awal, tetapi melakukan *seek* ke bagian paling akhir file. Dengan cara ini, sistem mengabaikan log lama dan hanya memproses event baru setelah aplikasi dijalankan. Desain ini merupakan implementasi pipeline real-time dari alur Suricata → Guardian AI → Dashboard, di mana setiap event baru langsung diklasifikasikan, diproses oleh model AI, dan kemudian dikirimkan ke dashboard tanpa menunggu batch processing atau refresh manual. Pendekatan ini juga mencegah misleading data akibat pembacaan ulang log lama yang tidak relevan.

Setelah server berjalan, terminal menampilkan log koneksi HTTP yang berasal dari browser pengguna. Baris GET /debug_stats menunjukkan bahwa browser mengakses endpoint debugging yang menampilkan statistik internal Guardian AI secara langsung dari memori. Baris GET / mengonfirmasi bahwa halaman utama dashboard telah berhasil dimuat, yaitu halaman yang berisi panel Total Events, Rule Alerts, AI Anomalies, AI Summary, tabel Live Alerts, dan grafik Chart.js. Sementara itu, GET /stream menandakan bahwa browser membuka koneksi Server-Sent Events (SSE). Melalui koneksi SSE inilah Guardian AI mengirimkan data update statistik dan alert secara kontinu tanpa perlu reload. Keberadaan ketiga GET request ini membuktikan bahwa dashboard memang sedang berjalan dan sedang diakses melalui localhost.

Pada bagian bawah terminal, terdapat log tambahan seperti [TAIL] dns 192.168.100.50 -> 118.98.115.69. Log tersebut berasal dari fungsi tailer yang berada di dalam kode dashboard_app.py. Setiap event yang dicatat Suricata di eve.json akan ditampilkan dalam bentuk ringkasan singkat oleh tailer. Label dns menunjukkan bahwa event yang diterima adalah transaksi DNS, sedangkan dua alamat IP yang muncul menggambarkan sumber dan tujuan query DNS tersebut. Keberadaan log DNS ini membuktikan bahwa Suricata sedang menulis

log secara aktif, dan Guardian AI berhasil membacanya secara real time. Meskipun event DNS tidak menambah counter utama seperti alert atau AI anomalies, pergerakan log ini menandakan bahwa pipeline log monitoring berjalan dengan benar.

Dari keseluruhan output tersebut dapat disimpulkan bahwa ketiga lapisan arsitektur Guardian AI bekerja secara berkesinambungan. Lapisan pertama adalah Suricata, yang memonitor interface jaringan dan menulis seluruh aktivitas ke dalam eve.json. Lapisan kedua adalah Guardian AI Engine yang menjalankan proses tailing untuk membaca log secara langsung, mengklasifikasikan event (alert, flow, dns, ai_test), menerapkan model anomaly detection seperti Isolation Forest untuk event flow, serta mengonversi event ke format yang dikirimkan ke dashboard. Lapisan ketiga adalah Dashboard & Observability yang dijalankan oleh Uvicorn melalui port 8000 dan menampilkan event secara visual melalui koneksi SSE. Dengan demikian, informasi pada terminal ini membuktikan bahwa seluruh pipeline deteksi ancaman berjalan stabil, mulai dari sumber data IDS, proses analitik AI, hingga pelaporan realtime melalui dashboard.

3.2. Menjalankan Suricata

```
cgai@wazuh:~/guardian-ai$ sudo suricata -c /etc/suricata/suricata.yaml -i enp0s3
t: suricata: This is Suricata version 8.0.2 RELEASE running in SYSTEM mode
t: threads: Threads created -> W: 2 FM: 1 FR: 1   Engine started.
```

Screenshot ini bisa kamu pakai sebagai bukti implementasi bahwa **lapisan IDS Suricata sudah benar-benar aktif dan berjalan**. Pada gambar terminal tersebut terlihat bahwa kamu berada di direktori proyek [~/guardian-ai](#) lalu menjalankan perintah `sudo suricata -c /etc/suricata/suricata.yaml -i enp0s3`, yang artinya Suricata dijalankan menggunakan file konfigurasi utama `/etc/suricata/suricata.yaml` dan mengawasi trafik pada interface jaringan `enp0s3`. Output `suricata: This is Suricata version 8.0.2 RELEASE running in SYSTEM mode` menunjukkan versi Suricata yang digunakan serta mode operasi sistemnya, sedangkan baris `threads: Threads created -> W: 2 FM: 1 FR: 1 Engine started` menegaskan bahwa thread-thread worker telah berhasil dibuat dan **engine Suricata sudah start** sehingga siap menangkap dan menganalisis paket lalu menuliskannya ke `eve.json`. Dengan demikian, screenshot ini melengkapi bukti bahwa sumber data untuk Cyber Guardian AI (log Suricata) sudah berjalan sesuai desain arsitektur.

3.3. Log Traffic pada Suricata

Screenshot ini bisa kamu pakai sebagai bukti implementasi bahwa Suricata benar-benar menulis log trafik ke file `/var/log/suricata/eve.json` dalam format JSON yang siap dibaca oleh Cyber Guardian AI.

Pada gambar terminal tersebut terlihat bahwa kamu menjalankan perintah `sudo tail -n 20 /var/log/suricata/eve.json`, lalu muncul deretan objek JSON dengan field seperti "timestamp", "event_type": "dns", "src_ip": "192.168.100.50", "dest_ip": "118.98.115.69", "proto": "UDP", serta detail kueri DNS ("rrname", "rrtype", "rdata" yang berisi domain seperti www.bing.com dan host e86033.dscx.akamaiedge.net). Hal ini menunjukkan bahwa engine Suricata yang sebelumnya kamu jalankan pada interface `enp0s3` aktif menangkap paket DNS dari host 192.168.100.50 menuju internet, mengurai isinya, dan mencatatnya sebagai event `dns` di `eve.json`; dengan kata lain, pipeline pada layer IDS sudah berfungsi dan sumber data yang akan *di-tail* oleh Guardian AI telah terisi trafik nyata dari jaringanmu.

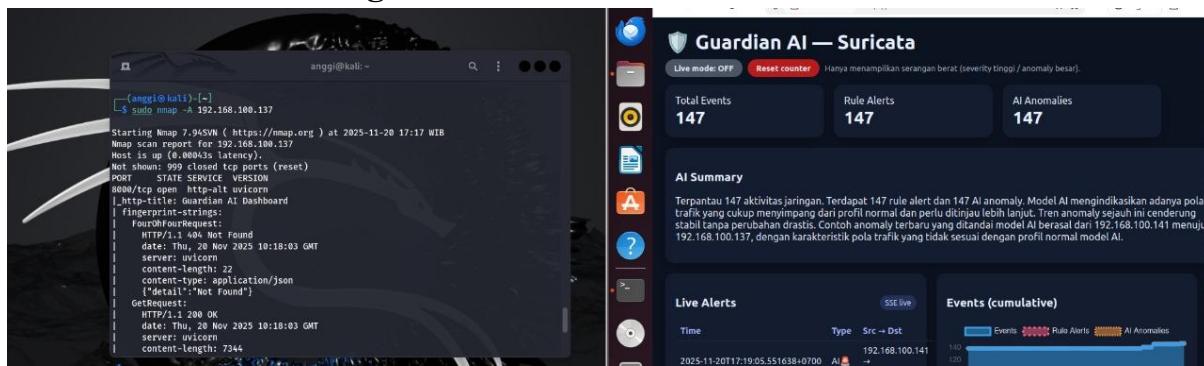
3.4. Uji Serangan pada Guardian Ai

```
cgai@wazuh: ~ × cgai@  
cgai@wazuh:~/guardian-ai$ curl http://testmyids.com  
curl http://testmyids.com  
curl http://testmyids.org  
uid=0(root) gid=0(root) groups=0(root)  
uid=0(root) gid=0(root) groups=0(root)  
uid=0(root) gid=0(root) groups=0(root)  
cgai@wazuh:~/guardian-ai$ █
```

Screenshot ini bisa kamu pakai sebagai bukti implementasi bahwa skenario serangan uji (RULE-based) berhasil kamu jalankan untuk memicu alert Suricata dan Guardian AI. Pada gambar terminal tersebut terlihat kamu berada di direktori `~/guardian-ai` lalu mengeksekusi beberapa kali perintah `curl http://testmyids.com` dan `curl http://testmyids.org`.

Domain *testmyids* memang sengaja disediakan untuk menguji sistem IDS: setiap kali di-*curl*, server akan mengirimkan respon khusus (di screenshot terlihat keluaran **uid=0(root)** **gid=0(root)** **groups=0(root)** berulang) yang dikenali oleh rule Suricata sebagai pola serangan/payload uji. Akibatnya, Suricata akan menuliskan event **alert** ke file **eve.json**, yang kemudian dibaca oleh modul Guardian AI sebagai Rule Alert dan ikut menambah nilai Total Events serta Rule Alerts di dashboard. Dengan demikian, screenshot ini membuktikan bahwa kamu tidak hanya menangkap trafik pasif (seperti DNS), tetapi juga benar-benar melakukan *trigger* serangan uji yang memverifikasi bahwa jalur trafik → Suricata → eve.json → Guardian AI → Dashboard bekerja sesuai rancangan.

3.5. Simulasi Serangan



Pada screenshot ini terlihat simulasi serangan dan respons sistem secara utuh dari dua sisi. Di sisi kiri, pada mesin Kali Linux kamu menjalankan perintah `sudo nmap -A 192.168.100.137`, yaitu *aggressive scan* terhadap host 192.168.100.137. Hasil Nmap menunjukkan bahwa host tersebut hidup dan memiliki port 8000/tcp dalam keadaan open dengan service `http-alt` yang di-*fingerprint* sebagai unicorn, bahkan terdeteksi pula *HTTP title* “Guardian AI Dashboard” beserta respon HTTP 404 dan 200 dari web server FastAPI yang menjalankan dashboard. Ini menggambarkan peran mesin Kali sebagai penyerang/pelaku pemindaian yang mencoba mengidentifikasi layanan pada target.

Di sisi kanan, tampak tampilan web Guardian AI Suricata pada host 192.168.100.137 dengan Live mode: *OFF* dan *counter* yang menunjukkan Total Events 147, Rule Alerts 147, AI Anomalies 147. AI Summary di bawahnya menjelaskan bahwa terdapat 147 aktivitas jaringan, seluruhnya terkласifikasi baik sebagai rule alert maupun anomaly AI, dan menyebut contoh anomaly terbaru berasal dari IP penyerang (192.168.100.141) menuju IP target (192.168.100.137) dengan pola trafik yang menyimpang dari profil normal. Artinya, pemindaian Nmap dari mesin Kali tidak hanya terdeteksi oleh rule-based Suricata (sehingga Rule Alerts naik), tetapi juga dianggap anomali oleh model Isolation Forest (AI Anomalies ikut

naik), dan semua itu direkam serta diringkas secara real-time di dashboard. Dengan demikian, satu gambar ini menjadi bukti kuat bahwa skenario serangan *port scanning/aggressive scan* dari luar jaringan berhasil dipantau dan dianalisis oleh seluruh arsitektur Cyber Guardian AI, mulai dari IDS Suricata, modul AI, hingga tampilan dashboard.