LAPORAN TUGAS BESAR KEAMANAN JARINGAN

PENGEMBANGAN SISTEM MONITORING DAN DETEKSI SERANGAN SIBER MENGGUNAKAN HONEYPOT, WAZUH SIEM, DAN WIRESHARK

Dosen Pengampu:

Wilfridus Handaya. ST., M. Cs.

Kelompok 4:

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

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ATMA JAYA YOGYAKARTA

TAHUN 2025/2026

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi membuat jaringan komputer menjadi fondasi utama dalam operasional organisasi modern, baik dalam lingkungan akademik maupun bisnis. Hampir seluruh layanan penting seperti autentikasi pengguna, layanan web, manajemen data, hingga sistem transaksi digital sangat bergantung pada stabilitas dan keamanan jaringan. Ketergantungan ini menghadirkan risiko yang sama besarnya karena penyerang dapat memanfaatkan celah untuk meluncurkan serangan otomatis. Jenis serangan yang umum terjadi seperti *scanning*, *brute force*, dan *Distributed Denial of Service (DDoS)* dapat menyebabkan gangguan layanan jika tidak dideteksi dengan baik. Oleh karena itu, pemahaman mengenai pola serangan menjadi hal krusial dalam meningkatkan keamanan jaringan secara menyeluruh.

Salah satu pendekatan efektif untuk mempelajari perilaku penyerang adalah penggunaan honeypot, yaitu sistem umpan yang didesain untuk merekam seluruh aktivitas mencurigakan. Honeypot memungkinkan administrator menganalisis metode serangan tanpa membahayakan sistem utama karena perangkat ini tidak memiliki nilai operasional selain sebagai objek serangan. Cowrie merupakan salah satu honeypot yang banyak digunakan karena dapat memerangkap percobaan *login, command intruder,* hingga file transfer yang dicoba oleh penyerang. Selain honeypot, platform *Security Information and Event Management (SIEM)* seperti Wazuh dapat memberikan analisis lanjutan berupa deteksi, korelasi *log*, hingga pembuatan peringatan otomatis berdasarkan *rule* tertentu. Ketiga komponen ini diperkuat dengan alat analisis *traffic* seperti Wireshark yang mampu menampilkan paket jaringan secara *real time.*

Melalui kombinasi Cowrie, Wazuh, dan Wireshark, analisis serangan menjadi lebih komprehensif karena mencakup sudut pandang *endpoint, rule-based detection,* dan *packet-level*. Pendekatan ini dianggap sangat relevan karena memberikan gambaran nyata bagaimana serangan dilakukan, bagaimana jaringan merespons, serta bagaimana sistem monitoring mendeteksinya. Sebagaimana dinyatakan oleh Heluka dkk. [1], integrasi honeypot dan SIEM terbukti meningkatkan visibilitas serta akurasi deteksi terhadap aktivitas berbahaya. Dengan demikian, penelitian ini penting untuk memberikan gambaran utuh tentang cara kerja serangan umum serta strategi pertahanan yang sesuai.

1.2 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. Melakukan implementasi lengkap dan pengujian terhadap perilaku tiga jenis serangan siber menggunakan Cowrie, Wazuh, dan Wireshark.
2. Memberikan pemahaman mendalam mengenai pola serangan serta bagaimana serangan tersebut terdeteksi oleh sistem monitoring.
3. Menyusun dokumentasi komprehensif mengenai hasil pengujian, mulai dari instalasi perangkat, pelaksanaan uji serangan, hingga analisis perbandingan pola.
4. Mengevaluasi efektivitas Wazuh dalam menghasilkan *alert* dari *log* yang dikirim oleh honeypot.
5. Menghubungkan hasil analisis *log* dan *packet capture* untuk mendapatkan gambaran utuh dari serangan.

1.3 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah:

1. Bagaimana perilaku serangan *scanning, brute force,* dan *DDoS* ketika diarahkan ke sistem honeypot Cowrie.
2. Bagaimana Wazuh SIEM memproses, mendeteksi, dan memberikan *alert* terhadap ketiga jenis serangan tersebut.
3. Bagaimana pola paket serangan tercermin dalam analisis *traffic* menggunakan Wireshark.

4. Bagaimana perbandingan antara ketiga serangan ditinjau dari sudut pandang honeypot, sistem deteksi, dan *paket capture.*

1.4 Batasan Penelitian

Penelitian ini dibatasi pada:

1. Sistem dibangun dalam jaringan lokal berbasis virtualisasi (VirtualBox).
2. Honeypot yang digunakan dibatasi pada Cowrie dan tidak mencakup tipe honeypot lain, seperti Dionaea atau Kippo.
3. Jenis serangan yang diuji terbatas pada *scanning, brute force,* dan *DDoS.*
4. Wazuh digunakan sebatas SIEM dasar tanpa automasi respons lanjutan.
5. Wireshark hanya digunakan pada segmen jaringan tertentu untuk mendukung analisis serangan.

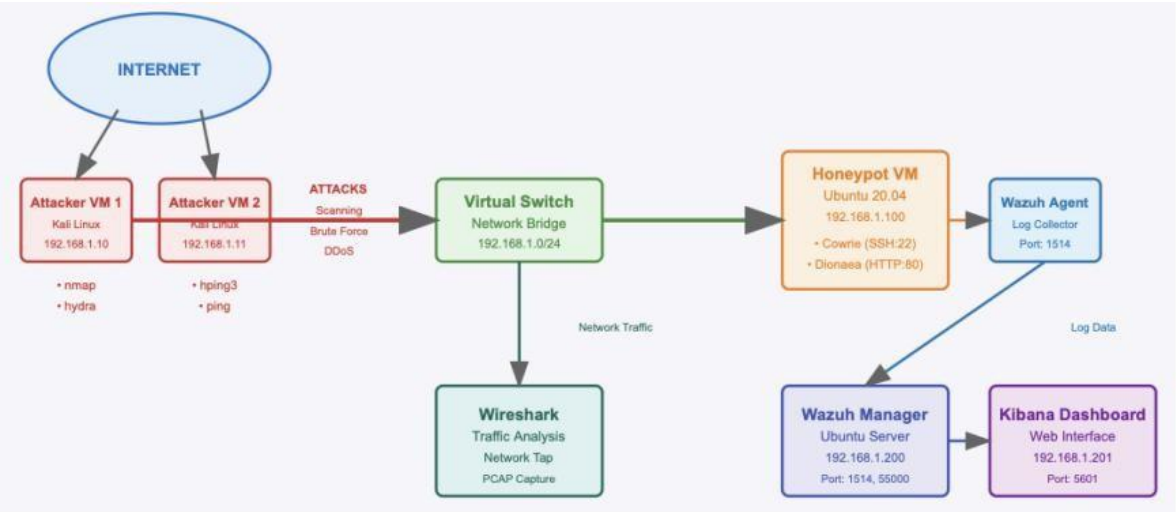
BAB II

METODOLOGI

2.1 Gambaran Umum Metode

Metode penelitian yang digunakan dalam tugas ini adalah metode eksperimen di lingkungan laboratorium virtual. Metode ini dipilih karena memberikan fleksibilitas dalam mensimulasikan berbagai jenis serangan jaringan tanpa mengganggu infrastruktur nyata. Dalam pendekatan ini, setiap komponen keamanan seperti Cowrie Honeypot, Wazuh SIEM, dan Wireshark dipasang dalam mesin virtual yang terpisah, sehingga interaksi antara *attacker* dan target dapat diamati dengan jelas. Penelitian dilakukan dengan merancang skenario serangan tertentu, menjalankannya di lingkungan yang terkontrol, kemudian mengamati dan mencatat hasilnya dari tiga sudut pandang yaitu sistem honeypot, sistem deteksi, dan paket jaringan. Selain itu, proses eksperimen dilakukan secara berulang untuk memastikan keakuratan data serta mendapatkan pola yang konsisten dari masing-masing jenis serangan.

2.2 Topologi Sistem



Topologi terdiri dari empat komponen utama:

- *Attacker* → menjalankan *scanning, brute force,* atau *DDoS.*
- Honeypo*t* Cowrie → target serangan.
- Wazuh Server → menerima *log* dari honeypot melalui *agent.*
- Wireshark → menangkap *traffic* antara *attacker* dan honeypot.

Setiap mesin ditempatkan dalam jaringan internal VirtualBox sehingga seluruh interaksi hanya terjadi di dalam lingkungan lab virtual, bukan pada jaringan publik. Mesin *attacker* menggunakan Kali Linux yang memiliki berbagai *tools* serangan bawaan, sedangkan honeypot menggunakan Ubuntu Server dengan Cowrie yang dipasang sebagai layanan SSH/Telnet palsu. Server Wazuh ditempatkan pada mesin terpisah yang menjalankan Wazuh Manager serta *dashboard* untuk memantau semua aktivitas. Topologi ini dirancang agar alur serangan dari *attacker* menuju honeypot dapat dikumpulkan dan dianalisis secara menyeluruh oleh Wazuh dan Wireshark.

2.3 *Tools* dan *Software* yang Digunakan

Tugas ini memanfaatkan beberapa tools dan software untuk mendukung implementasi dan pengujian.

- VirtualBox sebagai platform virtualisasi untuk membuat lingkungan laboratorium yang terisolasi dan aman untuk uji serangan.
- Honeypot Cowrie menggunakan Ubuntu Server dan berperan sebagai target serangan karena mampu mencatat aktivitas *login, command,* dan interaksi penyerang dengan sangat detail.
- Wazuh Agent dipasang di honeypot VM untuk mengirim log ke Wazuh Manager dan memicu *alert* otomatis untuk aktivitas mencurigakan.
- Wazuh Manager + *Dashboard* sebagai pusat monitoring dan analisis, serta visualisasi *alert* dan statistik serangan.
- Wireshark untuk *capture* semua *traffic network,* memahami pola serangan, dan *packet analysis.*

2.4 Langkah Implementasi

Tahap implementasi diawali dengan menyiapkan seluruh mesin virtual yang digunakan dalam tugas ini menggunakan platform VirtualBox. Setiap mesin diberikan konfigurasi jaringan yang sesuai, yaitu menggunakan mode *Internal Network* agar komunikasi antar-mesin berlangsung dalam lingkungan yang aman dan terisolasi. Pengaturan ini memastikan bahwa seluruh serangan yang diuji tidak keluar ke jaringan publik dan tidak mengganggu sistem di luar laboratorium virtual. Selain itu, setiap mesin diberikan spesifikasi yang cukup, terutama pada sisi Wazuh Manager yang membutuhkan kapasitas penyimpanan lebih besar. Pada tahap ini juga dilakukan pengujian awal untuk memastikan semua mesin dapat dikenali oleh VirtualBox tanpa *error* konfigurasi.

Setelah lingkungan virtual siap, proses instalasi Cowrie dilakukan pada mesin honeypot sebagai komponen utama penelitian. Instalasi dimulai dengan pemasangan *dependency* seperti *Python3, pip, virtual environment,* serta pustaka pendukung seperti *libssl-dev* dan *libffi-dev*. Langkah selanjutnya adalah membuat *user* khusus bernama *cowrie* agar layanan honeypot dapat dijalankan secara terpisah dan aman dari *user root*. Setelah *repository* Cowrie di-*clone* dari GitHub, dibentuk virtual *environment* Python tempat seluruh modul Cowrie dipasang agar tidak bercampur dengan paket sistem. Pada tahap akhir, dilakukan konfigurasi layanan SSH dan Telnet palsu serta pengaturan *logging* untuk memastikan Cowrie siap menampung serangan dari *attacker.*

Implementasi dilanjutkan dengan pemasangan Wazuh Manager pada mesin server yang berfungsi sebagai pusat analisis log dan deteksi ancaman. Instalasi dilakukan menggunakan skrip resmi dari Wazuh yang secara otomatis mengatur komponen manager, indexer, dan *dashboard*. Setelah instalasi selesai, dilakukan registrasi Wazuh Agent pada mesin honeypot agar log dari Cowrie dapat dikirimkan ke server secara *real time*. *Dashboard* Wazuh kemudian digunakan untuk memverifikasi bahwa *agent* terhubung dan seluruh *event* baru dapat terbaca dengan benar. Pada tahap ini juga dilakukan pengecekan *rule* untuk memastikan bahwa Wazuh mampu memberikan *alert* sesuai jenis serangan yang akan diuji.

Pada mesin monitoring dipasang aplikasi Wireshark yang digunakan untuk menangkap dan menganalisis paket jaringan. Instalasi dilakukan setelah memastikan *interface* jaringan virtual sudah aktif dan dapat menerima paket dari komunikasi antara *attacker* dan honeypot. Wireshark kemudian diuji melalui *capture* singkat untuk memastikan paket dapat terbaca tanpa *error* dan filter jaringan dapat diterapkan dengan baik. Penggunaan Wireshark sangat penting

karena paket hasil serangan seperti *SYN flood* atau *brute force handshake* hanya dapat dianalisis pada level paket. Dengan demikian, Wireshark memberikan perspektif yang berbeda dibandingkan log Cowrie maupun *alert* Wazuh.

Tahap implementasi ditutup dengan pengujian konektivitas dari mesin *attacker* ke honeypot sebelum skenario serangan dijalankan. Pengujian dilakukan menggunakan perintah *ping* untuk memastikan respon ICMP dapat diterima secara normal dan tidak ada gangguan pada jaringan internal. Selain itu, dilakukan uji akses SSH ke honeypot untuk memastikan Cowrie merespons permintaan *login* palsu sesuai yang diharapkan. Jika layanan tambahan seperti HTTP diaktifkan, dilakukan pula pengecekan awal melalui *browser* atau *curl* untuk memastikan layanan tersebut dapat diakses. Setelah seluruh koneksi berjalan normal dan semua komponen terhubung, lingkungan dinyatakan siap untuk menjalankan skenario serangan yang telah direncanakan.

2.5 Skenario Pengujian

Skenario pengujian dalam tugas ini dirancang untuk mensimulasikan tiga jenis serangan jaringan paling umum yang sering terjadi pada layanan server, yaitu *Scanning, Brute Force,* dan *Distributed Denial of Service (DDoS).* Setiap skenario disusun untuk menggambarkan kondisi nyata yang biasanya ditemui pada serangan terhadap layanan SSH maupun *port* lain yang terbuka. Tujuan dari penyusunan skenario ini adalah untuk mengamati bagaimana honeypot Cowrie mencatat aktivitas serangan, bagaimana Wazuh memberikan *alert* berdasarkan *rule* yang berlaku, serta bagaimana Wireshark memperlihatkan pola paket pada setiap skenario. Dengan merancang tiga serangan yang berbeda ini, kita dapat membandingkan karakteristik masing-masing serangan baik dari sisi intensitas trafik, pola log, maupun tingkat risiko. Seluruh pengujian dilakukan secara bertahap untuk memastikan bahwa setiap serangan dapat diamati dengan jelas tanpa mengganggu proses analisis selanjutnya.

a) Scanning (Reconnaissance)

Skenario pertama adalah *Scanning (Reconnaissance)* yang dijalankan menggunakan *tools* nmap dari mesin *attacker.* Serangan *scanning* digunakan untuk mengidentifikasi *port* yang terbuka, layanan yang berjalan, hingga versi layanan yang digunakan pada honeypot Cowrie. Aktivitas ini menjadi tahap awal yang biasa dilakukan penyerang untuk mengumpulkan informasi sebelum meluncurkan serangan lebih berbahaya. Hasil dari skenario ini akan dianalisis melalui log Cowrie yang

mencatat setiap koneksi singkat dan dari Wazuh yang mengeluarkan *alert* terkait *port scanning.*

b) Brute Force (Credential Attack)

Skenario kedua adalah *Brute Force Attack*, yaitu percobaan *login* SSH berulang yang dijalankan menggunakan *tools* Hydra. Pada skenario ini, *attacker* mencoba melakukan *login* ke layanan SSH honeypot dengan kombinasi *username* dan *password* secara otomatis menggunakan *wordlist* tertentu. Serangan jenis ini sangat umum terjadi pada server yang mengekspos layanan SSH karena penyerang berharap menemukan kredensial yang lemah. Pengujian *brute force* dijalankan dalam durasi tertentu untuk memunculkan pola *login* gagal dalam jumlah besar sehingga dapat terlihat jelas pada log Cowrie dan *alert* Wazuh. Wireshark juga akan menangkap pola *handshake* SSH yang muncul berulang kali sebagai indikasi terjadinya percobaan autentikasi berlebih.

c) DDoS (Availability Attack)

Skenario ketiga adalah *DDoS (Availability Attack)* yang dijalankan menggunakan *tools* hping3 dan ping flood dari mesin *attacker.* Serangan ini bertujuan membanjiri honeypot dengan paket dalam jumlah besar sehingga membuat layanan menjadi lambat atau tidak merespons. Hping3 digunakan untuk melakukan *SYN flood*, yaitu pengiriman paket SYN terus menerus tanpa melanjutkan proses *handshake* TCP, sehingga server menerima banyak permintaan palsu dalam waktu singkat. Sementara itu, *ping flood* digunakan untuk menghasilkan trafik ICMP yang sangat besar ke IP honeypot, sehingga konektivitas menjadi tidak stabil. Pada skenario ini, Wireshark diharapkan menampilkan lonjakan trafik yang signifikan, sedangkan Wazuh memberikan *alert* dengan tingkat tinggi yang menunjukkan adanya indikasi DDoS.

BAB III

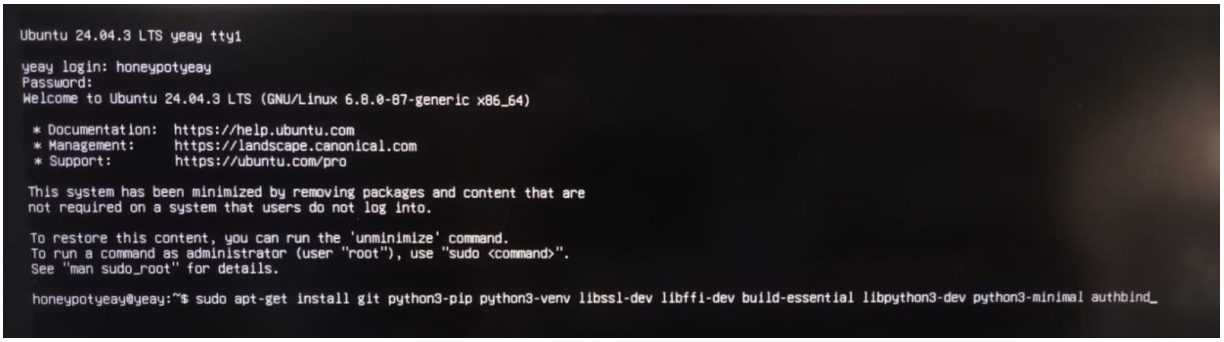
IMPLEMENTASI, HASIL, DAN ANALISIS

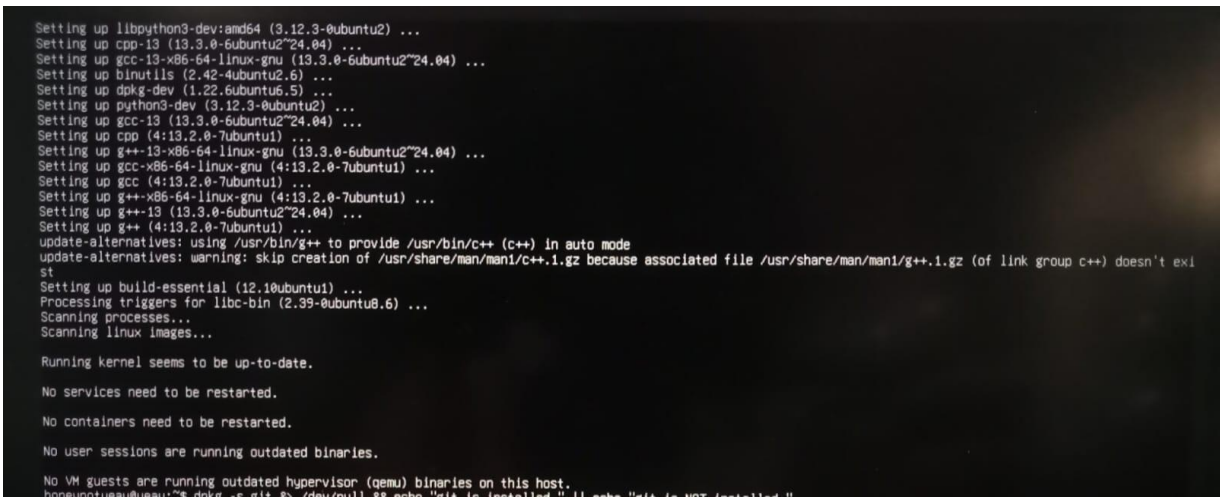
3.1 Setup Honeypot (Cowrie)

Langkah 1: Instal dependensi sistem

Tujuannya adalah menyediakan paket-paket sistem dan *library* yang diperlukan agar Cowrie dan modul Python yang diperlukan bisa dikompilasi dan dijalankan dengan benar.. Untuk sistem berbasis Debian, seperti Ubuntu dan Debian Bookworm, jalankan perintah:

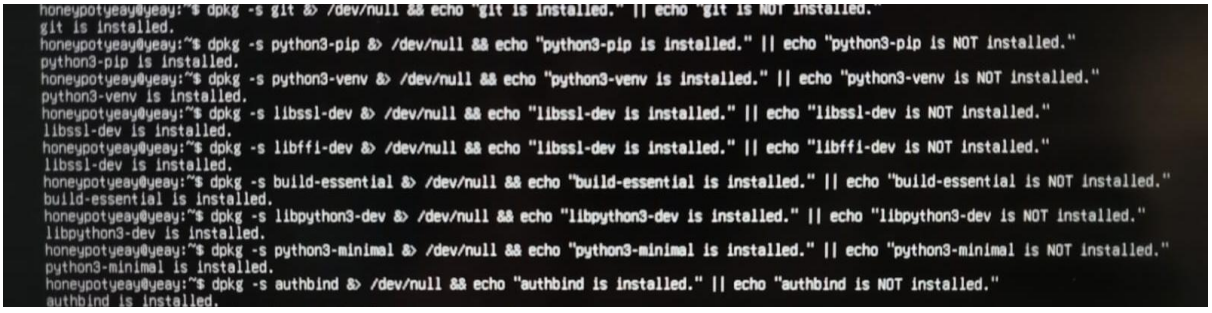
```
sudo apt-get install git python3-pip python3-venv libssl-dev
libffi-dev build-essential libpython3-dev python3-minimal authbind
```





Paket seperti *libssl-dev, libffi-dev,* dan *build-essential* dibutuhkan untuk membangun modul Python yang berinteraksi dengan *TLS/cryptography* dan komponen *native*. *python3-venv* dan *python3-pip* dibutuhkan untuk membuat virtual *environment* dan mengelola paket Python. *authbind* berguna jika ingin agar proses *non-root* dapat *bind ke port <1024* (opsional). Setelah instalasi, lakukan pengecekan paket untuk memastikan tidak ada dependensi yang terlewat, jalankan perintah berikut:

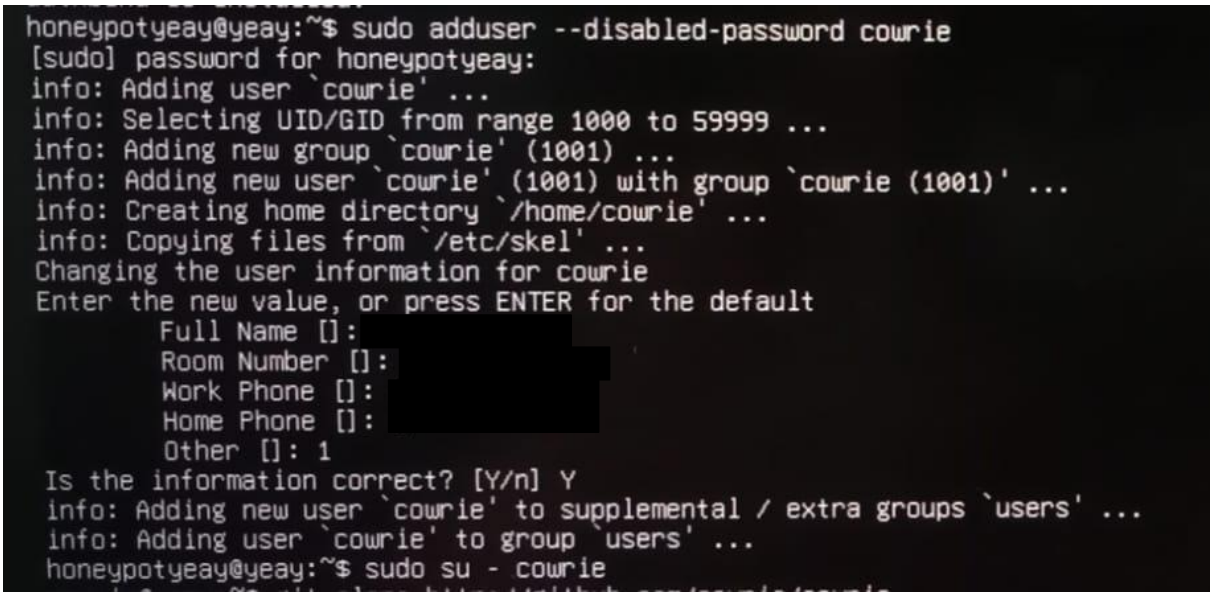
```
dpkg -s git &> /dev/null && echo "git is installed." || echo "git
is NOT installed."

dpkg -s python3-pip &> /dev/null && echo "python3-pip is
installed." || echo "python3-pip is NOT installed."

dpkg -s python3-venv &> /dev/null && echo "python3-venv is
installed." || echo "python3-venv is NOT installed."

dpkg -s libssl-dev &> /dev/null && echo "libssl-dev is installed."
|| echo "libssl-dev is NOT installed."

dpkg -s libffi-dev &> /dev/null && echo "libffi-dev is installed."
|| echo "libffi-dev is NOT installed."

dpkg -s build-essential &> /dev/null && echo "build-essential is
installed." || echo "build-essential is NOT installed."

dpkg -s libpython3-dev &> /dev/null && echo "libpython3-dev is
installed." || echo "libpython3-dev is NOT installed."

dpkg -s python3-minimal &> /dev/null && echo "python3-minimal is
installed." || echo "python3-minimal is NOT installed."

dpkg -s authbind &> /dev/null && echo "authbind is installed." ||
echo "authbind is NOT installed."
```



Langkah 2: Membuat *user* khusus untuk Cowrie

Tujuannya adalah menjalankan honeypot dengan *privilege* rendah (bukan *root*) untuk mengurangi risiko jika honeypot dikompromikan, jalankan perintah:

```
sudo adduser --disabled-password cowrie

sudo su - cowrie
```

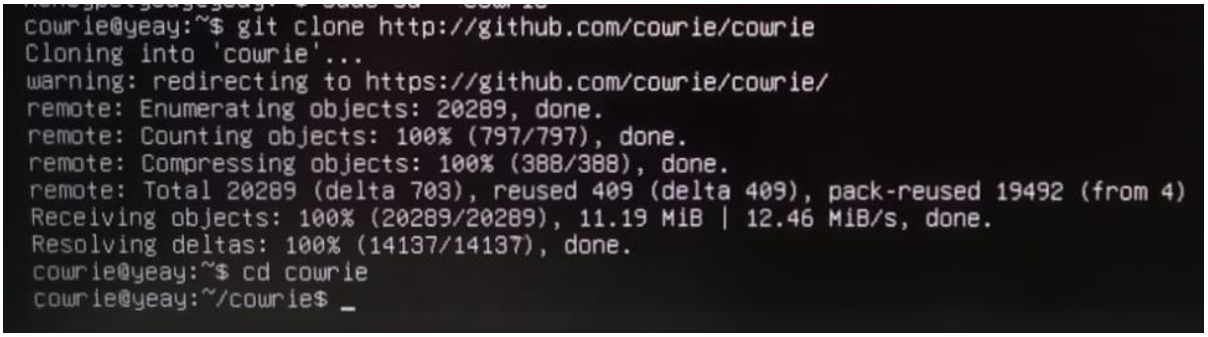


User cowrie dibuat tanpa *password login* interaktif. Semua langkah instalasi dan eksekusi Cowrie selanjutnya dilakukan dari akun ini sehingga file, proses, dan log berada dalam *scope user* terbatas.

Langkah 3: Meng-*clone* kode sumber Cowrie

Tujuannya adalah mengambil kode sumber Cowrie dari repositori resmi sehingga memiliki versi lengkap aplikasi dan konfigurasi, jalankan perintah:

```
git clone http://github.com/cowrie/cowrie

cd cowrie
```

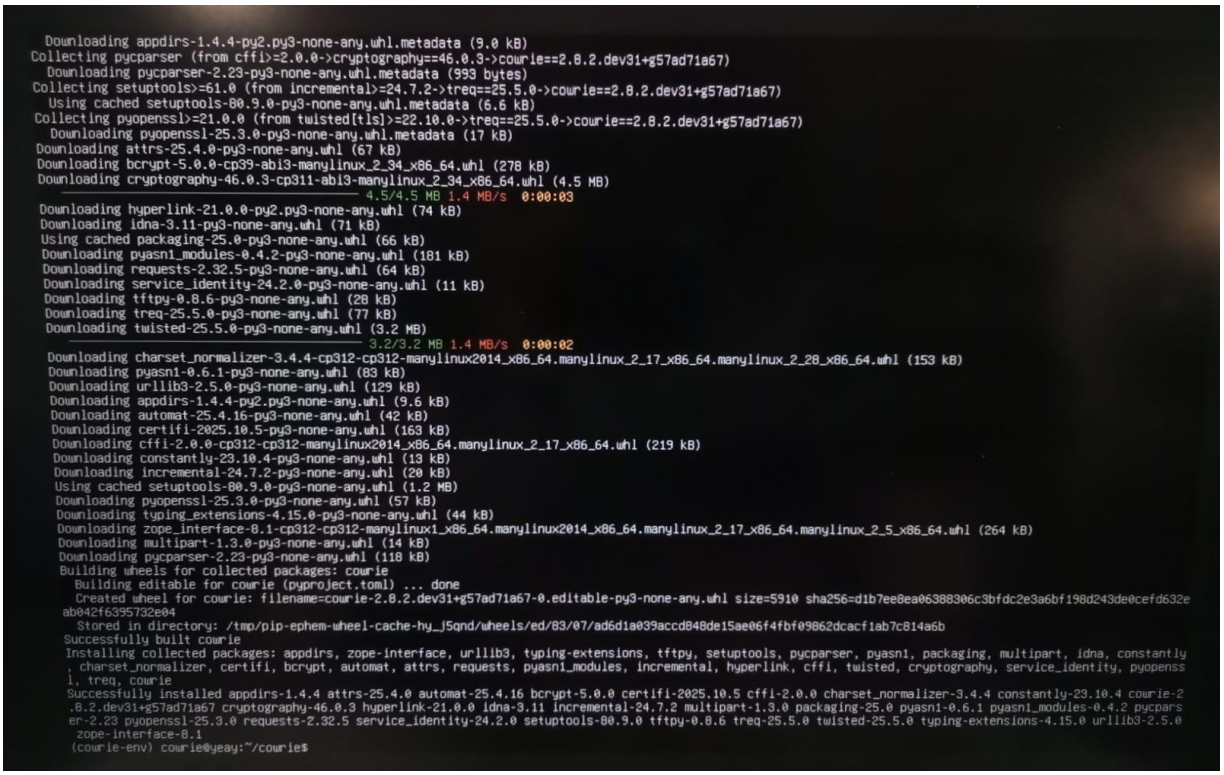


Dengan men-*clone repository*, kamu mendapatkan struktur *project* (bin/, etc/, var/, cowrie/) dan file contoh konfigurasi. Semua modifikasi konfigurasi dan instalasi paket Python dilakukan pada direktori hasil *clone* ini.

Langkah 4: Buat dan aktifkan Python virtual *environment*

Tujuannya adalah menjalankan Cowrie di lingkungan Python terisolasi agar dependensi tidak bertabrakan dengan paket sistem, jalankan perintah:

```
python3 -m venv cowrie-env

source cowrie-env/bin/activate

python -m pip install --upgrade pip

python -m pip install -e .
```



Virtualenv *(cowrie-env*) dibuat untuk mengisolasi paket-paket Python. Perintah *pip install -e* . menginstal *package* Cowrie dalam mode *editable*, memungkinkan pengembangan/perubahan tanpa *reinstall.*

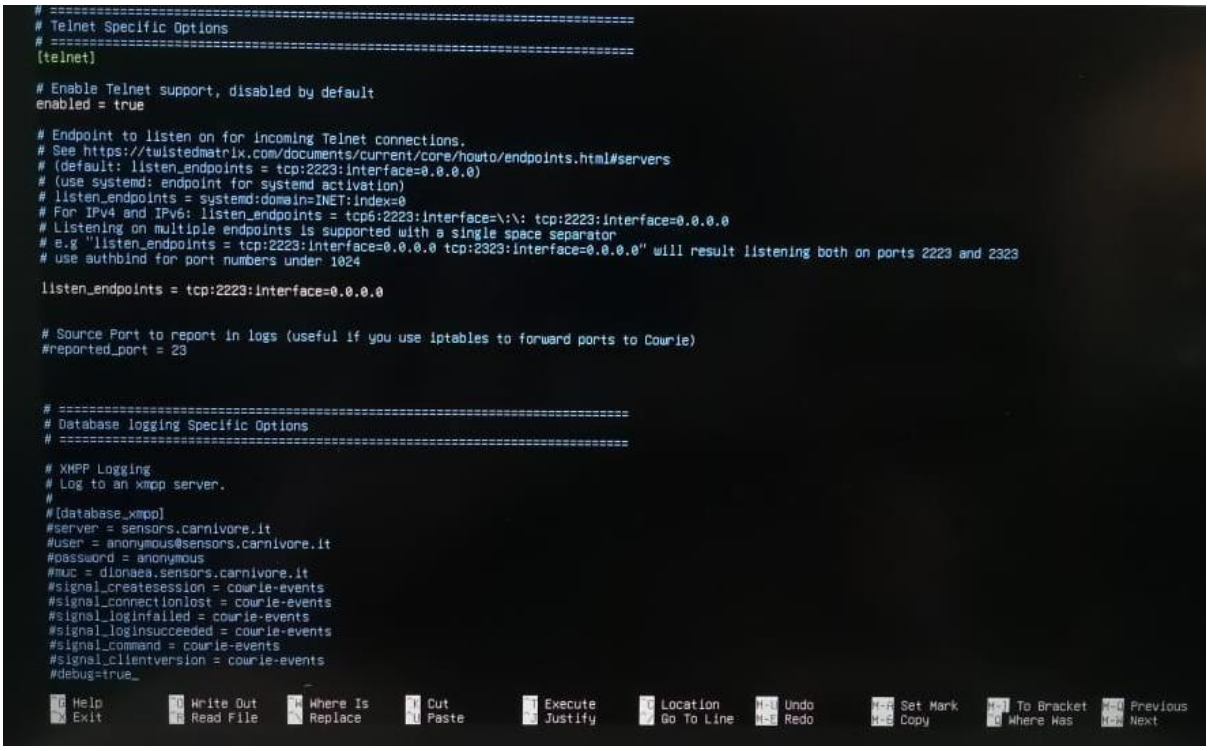
Langkah 5: Instal konfigurasi file

File konfigurasi terletak di cowrie/etc/ dengan dua versi:

- cowrie.cfg.dist → *default* (bisa ditimpa saat *update*)
- cowrie.cfg → konfigurasi personal (tidak ditimpa *update*)

Untuk mengaktifkan Telnet, buat cowrie.cfg dengan isi:

```
[telnet]

enabled = true
```



Langkah 6: Menjalankan Cowrie

Tujuannya adalah mengaktifkan honeypot dan mulai merekam interaksi penyerang, jalankan perintah:

```
cd ~/cowrie

source cowrie-env/bin/activate

cowrie start
```

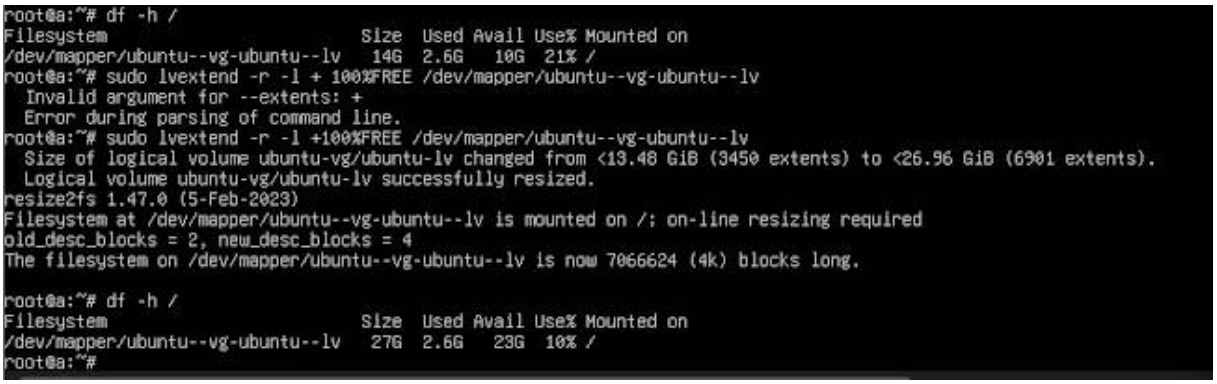
Binary bin/cowrie (atau perintah twistd pada versi lama) mengontrol daemon Cowrie. Setelah *start,* honeypot akan mendengarkan *port* yang dikonfigurasi dan mulai merekam *session,* perintah yang dikirim *attacker*, serta file transfer.

3.2 Konfigurasi Wazuh

Langkah 1: Persiapan sistem

Tujuannya adalah memastikan server siap menjalankan komponen Wazuh dengan sumber daya yang memadai. Proses dimulai dengan memperbarui paket sistem dan menyiapkan kapasitas disk yang cukup, jalankan perintah:

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



Pada lingkungan virtualisasi, kadang perlu memperbesar *logical volume* agar indeks dan data Wazuh memiliki ruang yang cukup. Tampak perintah *df -h* dan *lvextend* digunakan untuk memperbesar *logical volume* dan me-*resize filesystem* secara otomatis /dev/mapper/ubuntu--vg-ubuntu—lv, jalankan perintah:

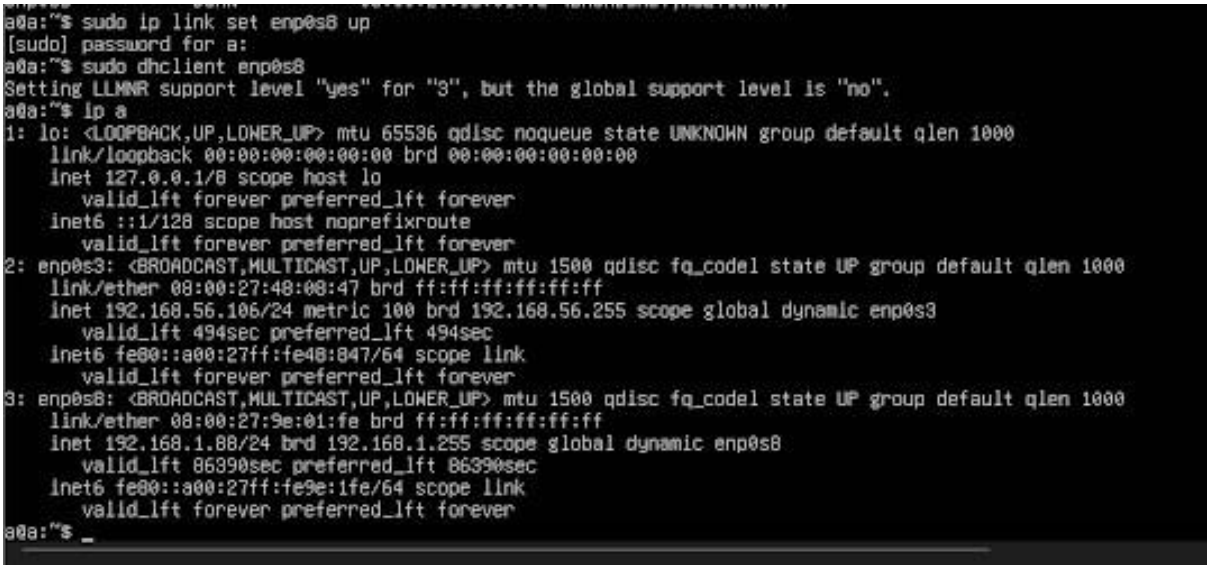
```
sudo lvextend -r -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu-
-lv
```

Output menunjukkan *filesystem* diperluas menjadi 27 GB dengan 23 GB ruang kosong tersedia.

Langkah 2: Konfigurasi jaringan

Wazuh memerlukan alamat IP yang stabil agar *agent* dapat mendaftar ke manager, oleh karena itu, pastikan *interface* jaringan aktif dan memperoleh IP yang benar, jalankan perintah:

```
sudo ip link set enp0s8 up
sudo dhclient enp0s8
ip a
```

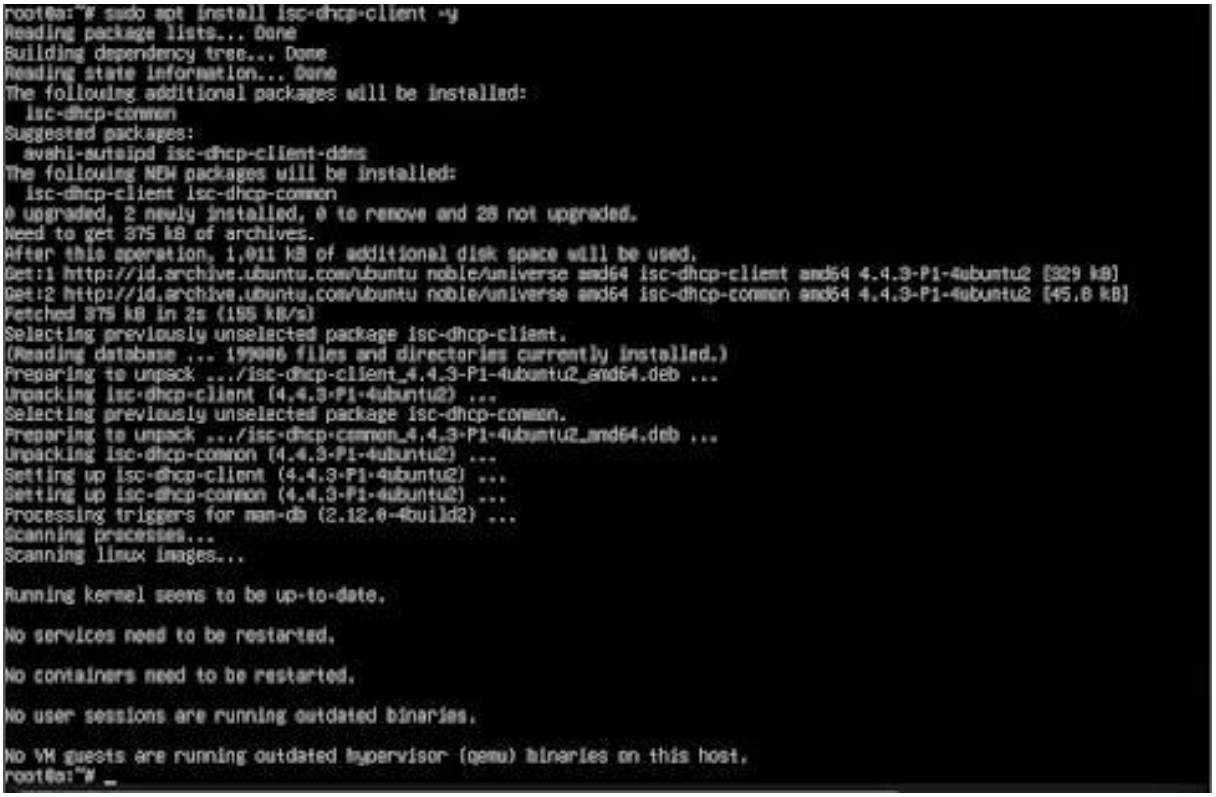


Pada gambar di atas, server Wazuh menggunakan IP 192.168.56.107 (enp0s3) sementara enp0s8 berada di segmen lain (192.168.1.8). Hal tersebut memastikan manajer dapat diakses dari jaringan yang relevan dan memudahkan proses registrasi *agent* nantinya.

Langkah 3: Instalasi paket pendukung

Sebelum menjalankan *installer* Wazuh, pasang paket pendukung dasar yang diperlukan sistem, seperti DHCP client atau alat lain yang relevan, jalankan perintah:

```
sudo apt install isc-dhcp-client -y
```

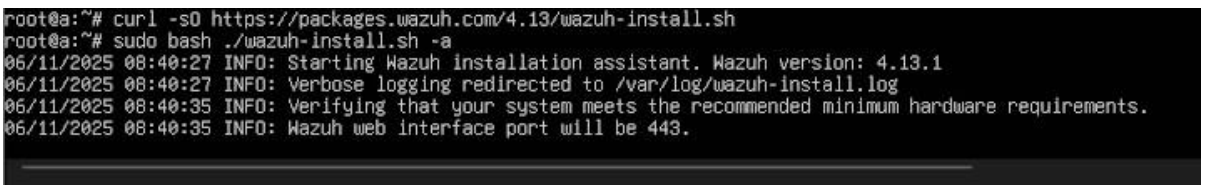


Paket pendukung ini memastikan server dapat berkomunikasi di jaringan dinamik dan menyelesaikan proses konfigurasi jaringan bila diperlukan. Proses instalasi paket harus berjalan tanpa *error*, pesan seperti *"Setting up isc-dhcp-client"* menandakan pemasangan sukses. Hal tersebut bersifat administratif tetapi penting untuk memastikan sistem tidak menemui hambatan jaringan selama atau setelah instalasi Wazuh.

Langkah 4: Menjalankan instalasi Wazuh Manager

Wazuh menyediakan skrip instalasi otomatis yang dapat mengatur komponen inti (manager, indexer, *dashboard*) secara cepat, jalankan perintah:

```
curl -sO https://packages.wazuh.com/4.13/wazuh-install.sh
sudo bash ./wazuh-install.sh -a
```



Installer akan memverifikasi bahwa hardware dan konfigurasi memenuhi rekomendasi (RAM/CPU/disk) dan akan menyiapkan Wazuh Web *Interface (default* di *port* 443). Pada output akan muncul versi (mis. 4.13.1) dan konfirmasi bahwa antarmuka web akan tersedia di

HTTPS *port* 443. Jika instalasi sukses, komponen Wazuh (manager, indexer*, dashboard*) akan terpasang dan service terkait berjalan otomatis.

Langkah 5: Verifikasi konfigurasi kernel dan sysctl

Setelah pemasangan selesai, Wazuh menerapkan sejumlah parameter kernel/sysctl yang direkomendasikan untuk keamanan dan stabilitas operasi (mis. kernel.kptr_restrict, fs.protected_hardlinks, vm.max_map_count) melalui file konfigurasi di /etc/sysctl.d/99-wazuh.conf. Terapkan dan periksa konfigurasi ini dengan, jalankan perintah:

```
sudo sysctl --system
```

```
root@a:~# sudo sysctl --system
* Applying /usr/lib/sysctl.d/10-apparmor.conf ...
* Applying /etc/sysctl.d/10-bufferbloat.conf ...
* Applying /etc/sysctl.d/10-console-messages.conf ...
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
* Applying /etc/sysctl.d/10-map-count.conf ...
* Applying /etc/sysctl.d/10-network-security.conf ...
* Applying /etc/sysctl.d/10-ptrace.conf ...
* Applying /etc/sysctl.d/10-zeropage.conf ...
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/99-wazuh.conf ...
* Applying /etc/sysctl.conf ...
kernel.apparmor_restrict_unprivileged_userns = 1
net.core.default_qdisc = fq_codel
kernel.printk = 4 4 1 7
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
kernel.kptr_restrict = 1
kernel.sysrq = 176
vm.max_map_count = 1048576
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.all.rp_filter = 2
kernel.yama.ptrace_scope = 1
vm.mmap_min_addr = 65536
kernel.pid_max = 4194304
fs.protected_fifos = 1
fs.protected_hardlinks = 1
fs.protected_regular = 2
fs.protected_symlinks = 1
vm.max_map_count = 262144
root@a:~# _
```

Contoh parameter yang umum diterapkan oleh installer :

- kernel.kptr_restrict=1
- fs.protected_hardlinks=1
- fs.protected_symlinks=1
- vm.max_map_count=262144.

Semua menunjukkan sistem telah dikonfigurasi aman.

Langkah 6: Kredensial default dan keamanan awal

Setelah instalasi selesai, *installer* menampilkan ringkasan yang berisi info cara akses Wazuh Web dan kredensial sementara. Contoh output yang muncul pada terminal:

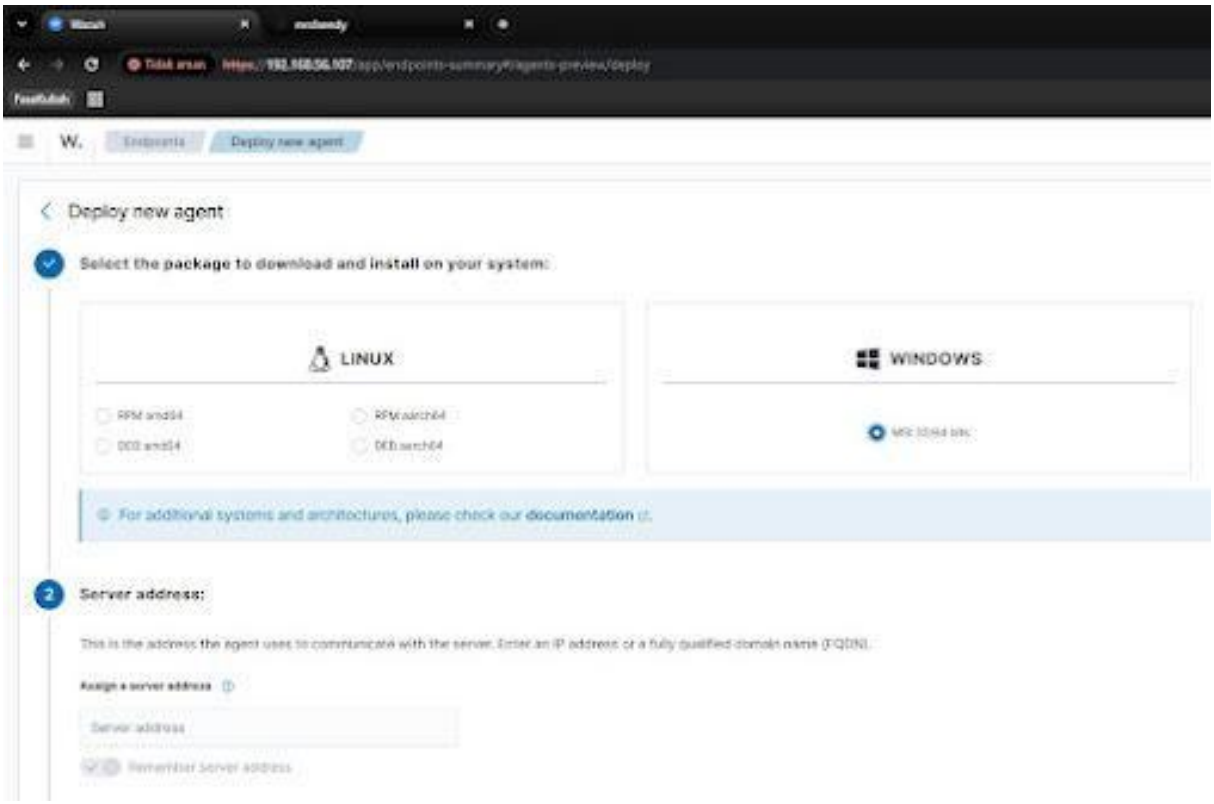
```
You can access the web interface https://<wazuh-dashboard-ip>:443
User: admin
Password: ........
Installation finished.
```

Catat kredensial ini dan segera lakukan tindakan keamanan awal ubah *password* admin, aktifkan HTTPS yang valid, dan batasi akses *dashboard* ke jaringan manajemen internal. Mengganti *password default* dan mengamankan akses web mengurangi risiko akses tidak sah ke antarmuka manajemen.

Langkah 7: Akses Wazuh web *interface*

Setelah komponen terpasang dan service berjalan, antarmuka web dapat diakses melalui *browser:*

```
https://<wazuh-dashboard-ip>

https://192.168.56.107
```



Saat pertama buka, *dashboard* akan menampilkan status awal seperti *"No agents registered"* dan tombol untuk *"Deploy new agent"*. Hal ini menandakan Wazuh Manager siap menerima *agent* dan konfigurasi lebih lanjut dapat dilakukan lewat UI.

Langkah 8: *Deploy* dan instalasi Wazuh Agent

Untuk menghubungkan *endpoint* ke manager, pilih menu *Deploy new agent* pada *dashboard*. Administrator memilih OS target (Linux/Windows) dan memasukkan alamat server.

Contoh:

```
Server address: 192.168.56.107
```

Dashboard menyediakan installer agent sesuai arsitektur:

- Linux → `.deb` atau `.rpm`
- Windows → `.msi`

Setelah *agent* diinstal di endpoint, agent akan muncul di tab *Endpoints → Agents.*

Langkah 9: Verifikasi status *agent* dan *troubleshooting*

Setelah *agent* diinstal, periksa status koneksi di *dashboard*. Jika *agent* menampilkan status *"never connected",* hal ini biasanya akibat masalah jaringan*, firewall*, atau konfigurasi agen. Pastikan *port* berikut diizinkan antara *agent* dan manager:

- 1514/tcp (data)
- 1515/tcp *(registration)*

Pada *endpoint,* jalankan perintah:

```
sudo systemctl status wazuh-agent
```

Perintah di atas untuk melihat proses *agent*. Pastikan status aktif

3.3 Persiapan Tools Serangan

Pada tahap ini, mesin *attacker* dipersiapkan dengan beberapa *tools* yang digunakan untuk melakukan simulasi serangan terhadap honeypot Cowrie. Pemilihan *tools* ini didasarkan pada skenario serangan yang harus diuji, sehingga seluruh perangkat yang digunakan merupakan alat standar dalam kegiatan *penetration testing*. Dengan menggunakan *tools* yang relevan terhadap skenario, pola serangan dapat direpresentasikan secara realistis dan mendekati kondisi serangan yang biasa terjadi di dunia nyata. Selain itu, persiapan *tools* dilakukan secara hati-hati untuk memastikan bahwa setiap perintah dapat dijalankan tanpa kendala teknis selama proses pengujian berlangsung. Tahap persiapan ini sangat penting karena kualitas data yang diperoleh bergantung pada keberhasilan eksekusi setiap tools yang digunakan dalam penelitian.

Tools pertama yang dipersiapkan adalah Nmap, yaitu utilitas pemindaian jaringan yang digunakan untuk melakukan *scanning* terhadap *port* dan layanan yang terbuka pada honeypot. Nmap dipilih karena kemampuannya yang sangat kuat dalam melakukan *reconnaissance,* seperti mengidentifikasi *port,* layanan, dan bahkan versi aplikasi yang berjalan pada target. *Tools* ini menghasilkan pola pengiriman paket SYN atau *probe* TCP ke berbagai *port* sehingga memberikan gambaran jelas tentang aktivitas *scanning* yang umum dilakukan penyerang sebelum melakukan eksploitasi lanjutan. Honeypot Cowrie diharapkan mencatat seluruh koneksi singkat tersebut, sementara Wazuh di sisi lain akan mendeteksi bahwa aktivitas pemindaian *port* terjadi dalam waktu singkat. Dengan demikian, penggunaan Nmap memberikan fondasi awal bagi analisis serangan *reconnaissance* yang lengkap.

Tools kedua yang dipersiapkan adalah Hydra, yaitu alat serangan *brute force* yang digunakan untuk melakukan percobaan *login* berulang pada layanan SSH yang disediakan oleh Cowrie. Hydra mampu mencoba kombinasi *username* dan *password* secara otomatis melalui *wordlist* sehingga sangat sesuai untuk mensimulasikan serangan *credential attack*. Pola serangan *brute force* ditandai dengan banyaknya entri *login* gagal yang terjadi secara berurutan dan cepat, sehingga mudah dikenali baik oleh Cowrie maupun Wazuh. Cowrie akan mencatat seluruh percobaan *login* termasuk kombinasi *username* dan *password* yang dicoba, sedangkan Wazuh akan memicu *rule brute force* yang menghasilkan *alert* tingkat menengah hingga tinggi. Dengan kemampuan ini, Hydra memberikan gambaran yang jelas mengenai ancaman *login* berlebih yang sering terjadi pada layanan SSH publik.

Tools selanjutnya yang digunakan adalah Hping3, yaitu utilitas jaringan yang digunakan untuk melakukan serangan DDoS tipe SYN *flood* ke honeypot. Hping3 mampu mengirimkan ribuan paket SYN dalam waktu singkat sehingga membuat *resource* target menjadi tinggi akibat terlalu banyak koneksi setengah jadi. Serangan ini tidak menyelesaikan proses *handshake* TCP, sehingga server menjadi terbebani oleh banyaknya permintaan palsu yang tidak pernah diselesaikan. Selain Hping3, digunakan pula *ping flood* melalui opsi ping -f, yang mengirimkan paket ICMP dalam jumlah sangat besar untuk menyerang ketersediaan layanan. Aktivitas DDoS ini menghasilkan trafik ekstrem yang akan terlihat jelas pada Wireshark, sementara Wazuh akan memberikan alert berlevel tinggi sebagai indikator adanya serangan terhadap ketersediaan. Dengan kombinasi kedua jenis DDoS ini, pola serangan dapat diamati dari dua jenis protokol sekaligus, yaitu TCP dan ICMP.

3.4 Testing Konektivitas

Sebelum skenario serangan dijalankan, dilakukan tahap pengujian konektivitas untuk memastikan bahwa seluruh komponen dalam topologi mesin *attacker*, honeypot Cowrie, dan Wazuh Manager dapat saling berkomunikasi tanpa hambatan. Tahap pengujian ini sangat penting karena keberhasilan serangan dalam memberikan data log yang lengkap sepenuhnya bergantung pada kestabilan konektivitas antar mesin. Jika terdapat gangguan jaringan atau konfigurasi IP yang tidak tepat, maka log serangan tidak akan tercatat secara benar di honeypot maupun tidak akan dikirimkan ke Wazuh untuk dianalisis. Oleh karena itu, verifikasi konektivitas dilakukan secara menyeluruh untuk menghindari kegagalan saat pengujian utama berlangsung. Dengan memastikan seluruh mesin saling terhubung, penelitian dapat berlanjut pada tahap eksekusi serangan secara optimal.

Pengujian pertama dilakukan dengan menjalankan perintah *ping* dari mesin *attacker* menuju alamat IP honeypot. Perintah *ping* digunakan untuk mengetahui apakah server honeypot dapat memberikan respons terhadap paket ICMP *(Internet Control Message Protocol)* yang dikirim *attacker.* Respons *ping* yang konsisten menunjukkan bahwa jaringan internal telah tersambung dengan baik dan tidak terdapat hambatan pada *routing* atau konfigurasi alamat IP. Selain itu, nilai *latency* yang stabil menunjukkan bahwa kondisi jaringan berada dalam keadaan normal dan siap digunakan untuk serangan dengan intensitas trafik yang lebih tinggi. Jika *ping* gagal, maka dapat dipastikan bahwa terdapat kesalahan konfigurasi jaringan yang harus diperbaiki sebelum pengujian dilanjutkan.

Setelah konektivitas dasar dipastikan, dilakukan pengujian akses SSH normal ke honeypot menggunakan kredensial yang valid atau yang memang disiapkan dalam konfigurasi Cowrie. Tahap ini bertujuan untuk memastikan bahwa layanan SSH palsu yang disediakan oleh Cowrie berjalan dengan benar dan dapat menerima koneksi masuk. Melalui pengujian ini juga dapat dipastikan bahwa Cowrie mencatat aktivitas autentikasi, baik yang berhasil maupun yang gagal, sehingga dapat digunakan sebagai dasar untuk menganalisis serangan *brute force* nantinya. Selain itu, aktivitas *login* normal ini membantu memastikan bahwa log dari honeypot benar-benar dapat diteruskan ke Wazuh melalui *agent* yang telah dipasang. Jika *login* tidak tercatat di Wazuh, maka berarti terdapat kesalahan integrasi yang harus diperbaiki sebelum uji serangan dimulai.

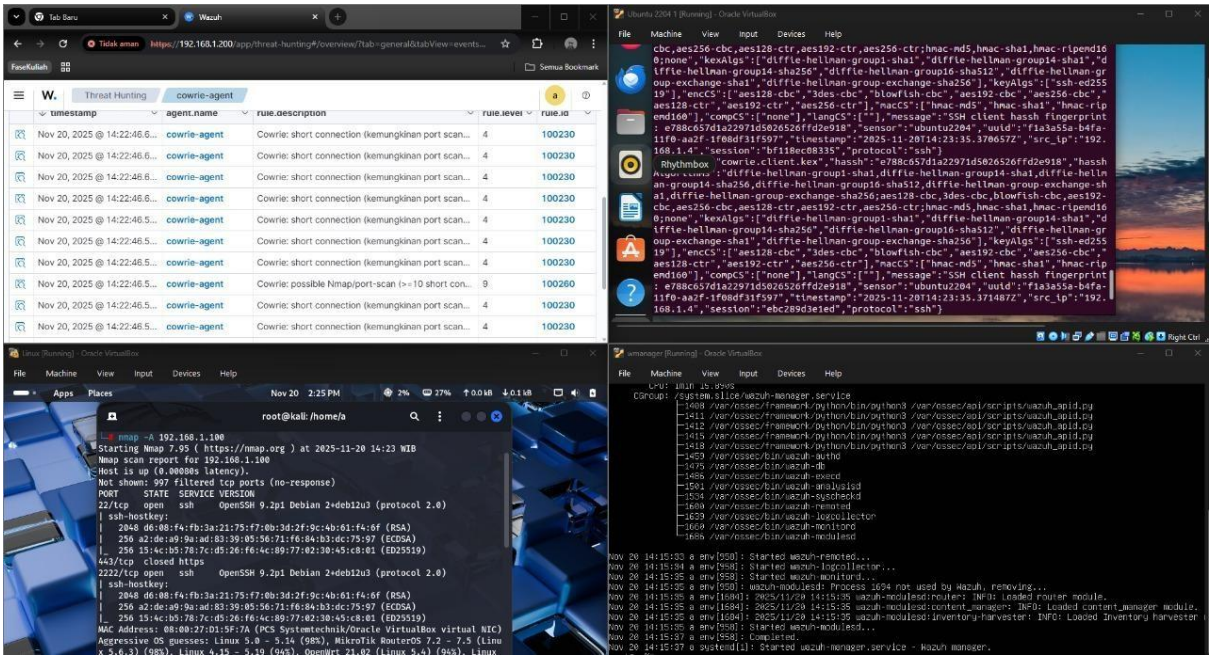
Tahap berikutnya adalah pengujian akses HTTP ke layanan web dari honeypot, apabila layanan tersebut tersedia dalam konfigurasi. Pengujian dilakukan menggunakan *browser* atau melalui perintah seperti *curl* untuk memastikan bahwa server web berjalan dengan baik dan dapat menerima permintaan dari *attacker.* Tujuan pengujian ini bukan untuk memicu serangan web, melainkan untuk memastikan bahwa layanan HTTP dapat dicapai dengan baik dan permintaan HTTP normal muncul di log server. Aktivitas akses normal seperti ini tidak dianggap sebagai aktivitas berbahaya oleh Wazuh karena tidak menunjukkan pola serangan, sehingga tidak akan menghasilkan *alert.* Namun, keberadaan akses ini penting sebagai pembanding terhadap pola trafik yang dihasilkan pada serangan web atau serangan berbasis HTTP lainnya.

Berdasarkan hasil keseluruhan dari uji konektivitas, aktivitas *ping, login* SSH normal, dan akses HTTP berhasil muncul pada log honeypot dan juga dapat terlihat melalui *dashboard* Wazuh tanpa memicu peringatan apa pun. Hal ini menunjukkan bahwa integrasi antara Cowrie dan Wazuh berfungsi dengan baik, di mana aktivitas normal dapat dipantau namun tidak dianggap sebagai ancaman. Tidak adanya alert dari Wazuh pada tahap ini mengonfirmasi bahwa *rule engine* Wazuh bekerja sesuai desain, yaitu hanya mendeteksi aktivitas yang bersifat abnormal, mencurigakan, atau berpotensi membahayakan sistem. Dengan lulusnya seluruh tahap ini, sistem berada dalam kondisi stabil dan siap untuk menjalankan tiga skenario serangan yang telah dirancang sebelumnya. Tahap konektivitas ini menjadi dasar penting untuk memastikan bahwa hasil pengujian selanjutnya valid dan dapat dianalisis secara akurat.

3.5 Screenshot dan Dokumentasi Setiap Serangan

Menyajikan hasil dokumentasi dari tiga jenis serangan *scanning, brute force*, dan *DDoS* yang dilakukan dari mesin *attacker* menuju honeypot Cowrie. Setiap serangan menghasilkan pola interaksi yang berbeda, baik di level log honeypot, alert Wazuh, maupun *traffic* jaringan. Dokumentasi ini nantinya digunakan sebagai dasar analisis pada bagian berikutnya, sesuai pendekatan yang juga dilakukan oleh penelitian SIEM sebelumnya

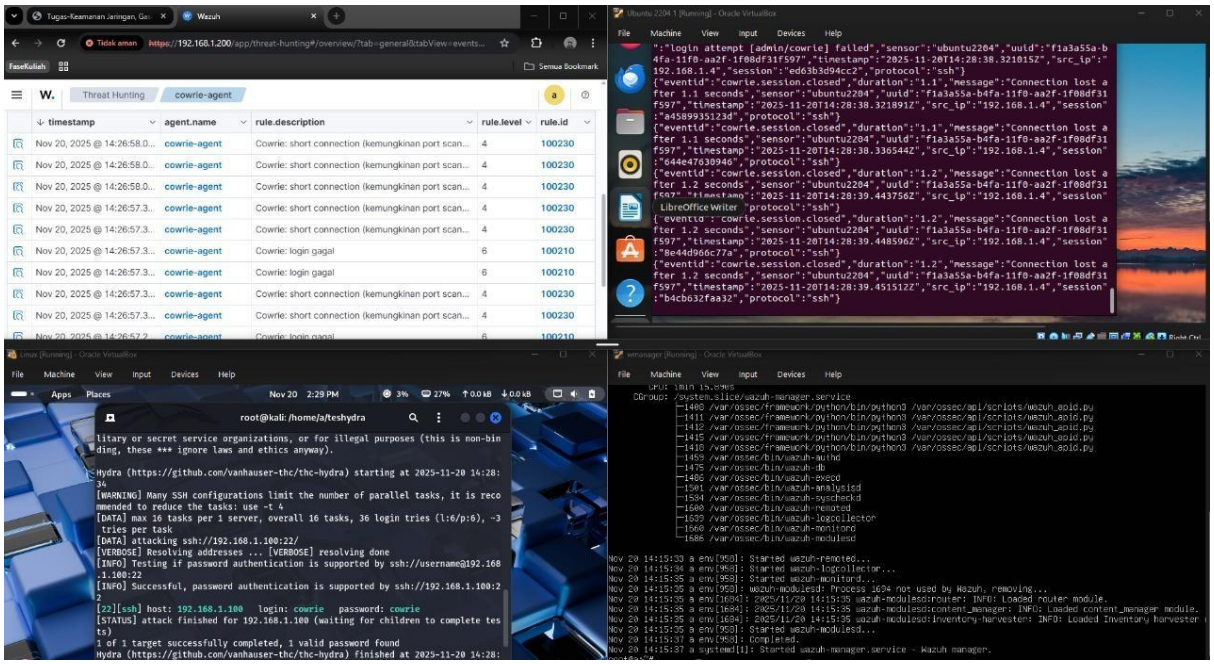
A. *Scanning* (Nmap)



Pada *screenshot* Nmap, terlihat bahwa *attacker* melakukan pemindaian terhadap alamat IP honeypot untuk mengidentifikasi *port* yang terbuka. Output Nmap menampilkan daftar *port open, closed*, atau *filtered*, beserta jenis layanan yang berjalan.

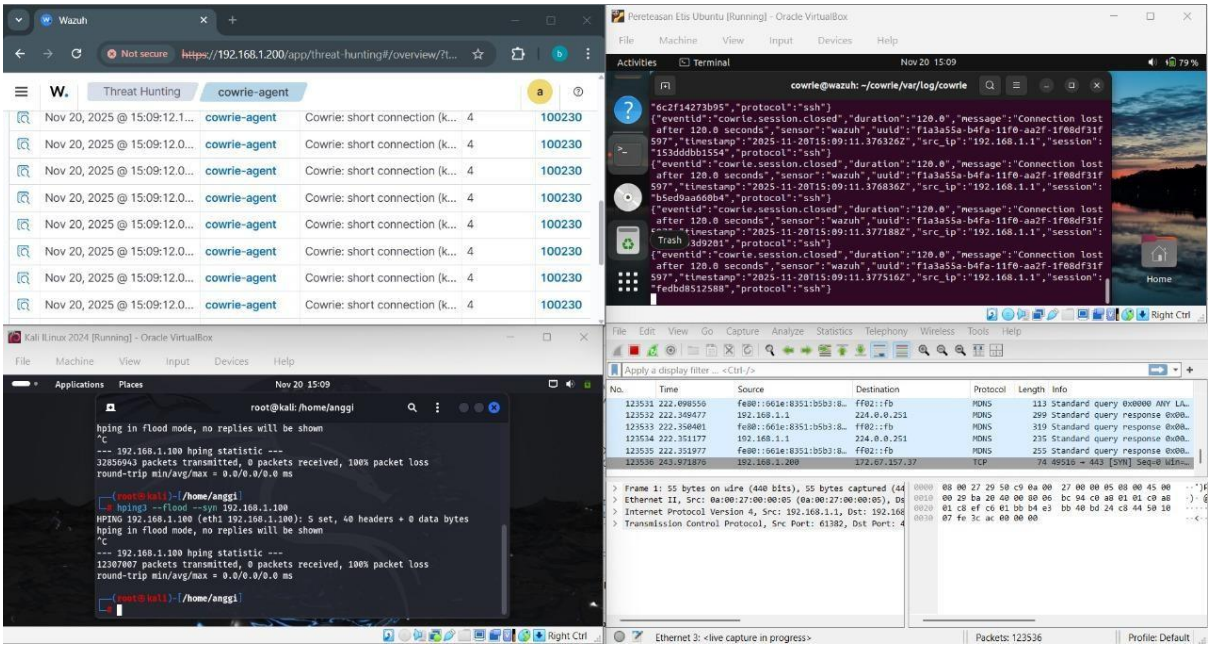
Pemindaian ini merupakan tahap awal *reconnaissance* yang sering digunakan penyerang untuk memetakan target sebelum melakukan eksploitasi lanjutan. Hasil ini sesuai dengan yang dijelaskan oleh Sofiyan Hadi [2], bahwa aktivitas *port scanning* menghasilkan komunikasi singkat pada banyak *port* secara berurutan, sehingga dapat menjadi indikator awal adanya *probing* terhadap server.

B. *Brute Force*



Screenshot Hydra menunjukkan proses percobaan *login* berulang menggunakan kombinasi *username* dan *password.* Terlihat bahwa Hydra mengirim ratusan permintaan autentikasi dalam waktu singkat ke *port* SSH honeypot. Sebagian besar respon yang muncul adalah *"failed login",* yang menandakan serangan *brute force* belum berhasil mendapatkan akses. Pola seperti ini banyak dibahas pada penelitian Wahid dkk. [3], yang menyebutkan bahwa *event failed password* secara berturut-turut merupakan salah satu pola paling umum yang dipantau oleh Wazuh karena identik dengan upaya pengambilalihan sistem.

C. *DDoS*

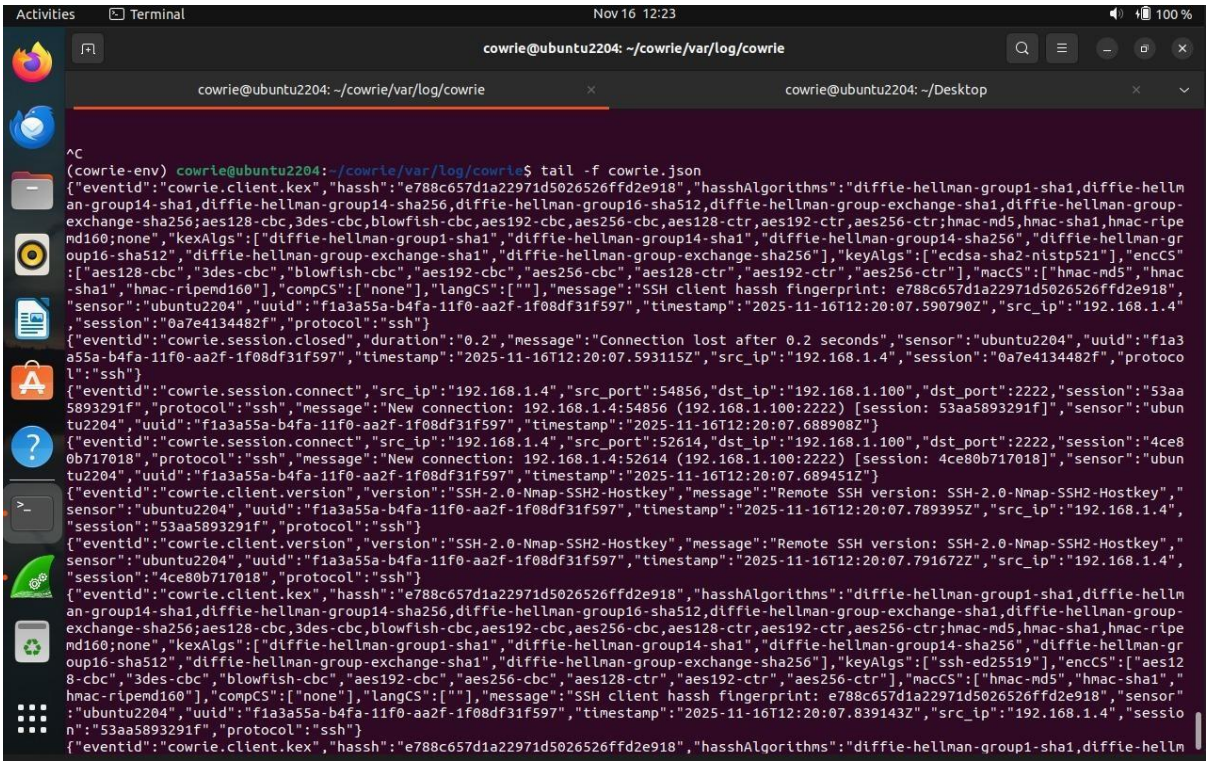


Screenshot untuk SYN *flood* menunjukkan bahwa *attacker* mengirimkan ribuan paket SYN dalam waktu yang sangat cepat. Tidak ada *payload* lanjutan atau *handshake,* sehingga server hanya menerima permintaan koneksi palsu secara terus-menerus. Efeknya, log honeypot menjadi penuh dengan entri *"connection attempt"* tanpa adanya sesi yang valid. Penelitian Heluka dan Sulistyo [1] juga menemukan bahwa serangan volumetrik semacam ini dapat dibaca sebagai lonjakan anomali *connection attempt,* meskipun honeypot seperti Cowrie tidak akan *"down",* tetapi tetap mencatat semua permintaan tersebut.

3.6 Analisis Log Honeypot

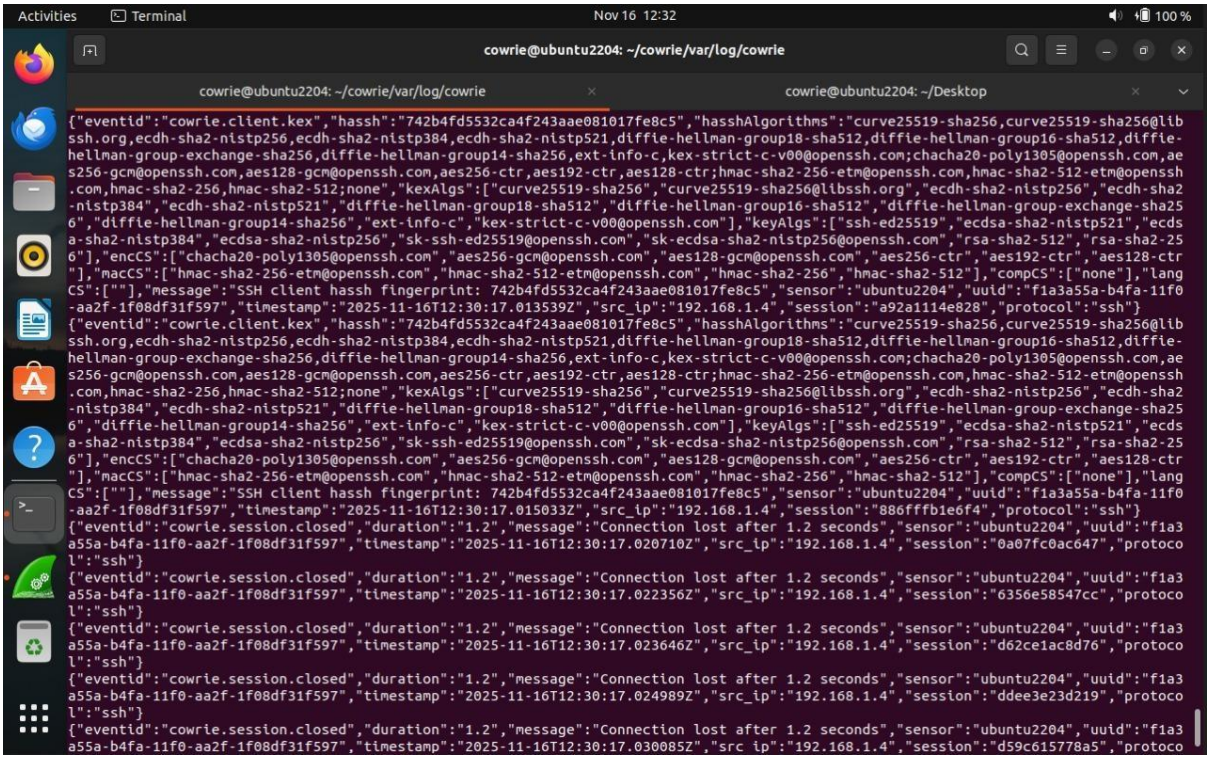
Analisis dilakukan berdasarkan log pada folder *cowrie/var/log* yang merekam semua interaksi *attacker* dengan honeypot. Cowrie menyimpan detail seperti IP penyerang, *command* yang diketik, koneksi yang dibuka, hingga *duration* dari setiap sesi.

A. *Scanning* (Nmap)



Pada log Nmap terlihat banyak entri *connection closed* dan *connection attempt* yang terjadi sangat cepat dalam interval milidetik. Setiap koneksi hanya terdiri dari *handshake* awal tanpa ada *payload* lanjutan. Cowrie menandai koneksi seperti ini sebagai interaksi minimal yang umum terjadi ketika dilakukan *port scanning*. Pola koneksi singkat ini sesuai dengan temuan penelitian SIEM [1], yang menyatakan bahwa *port scanning* menghasilkan *"short-lived connections"* yang menjadi indikator awal *reconnaissance.*

B. *Brute Force*

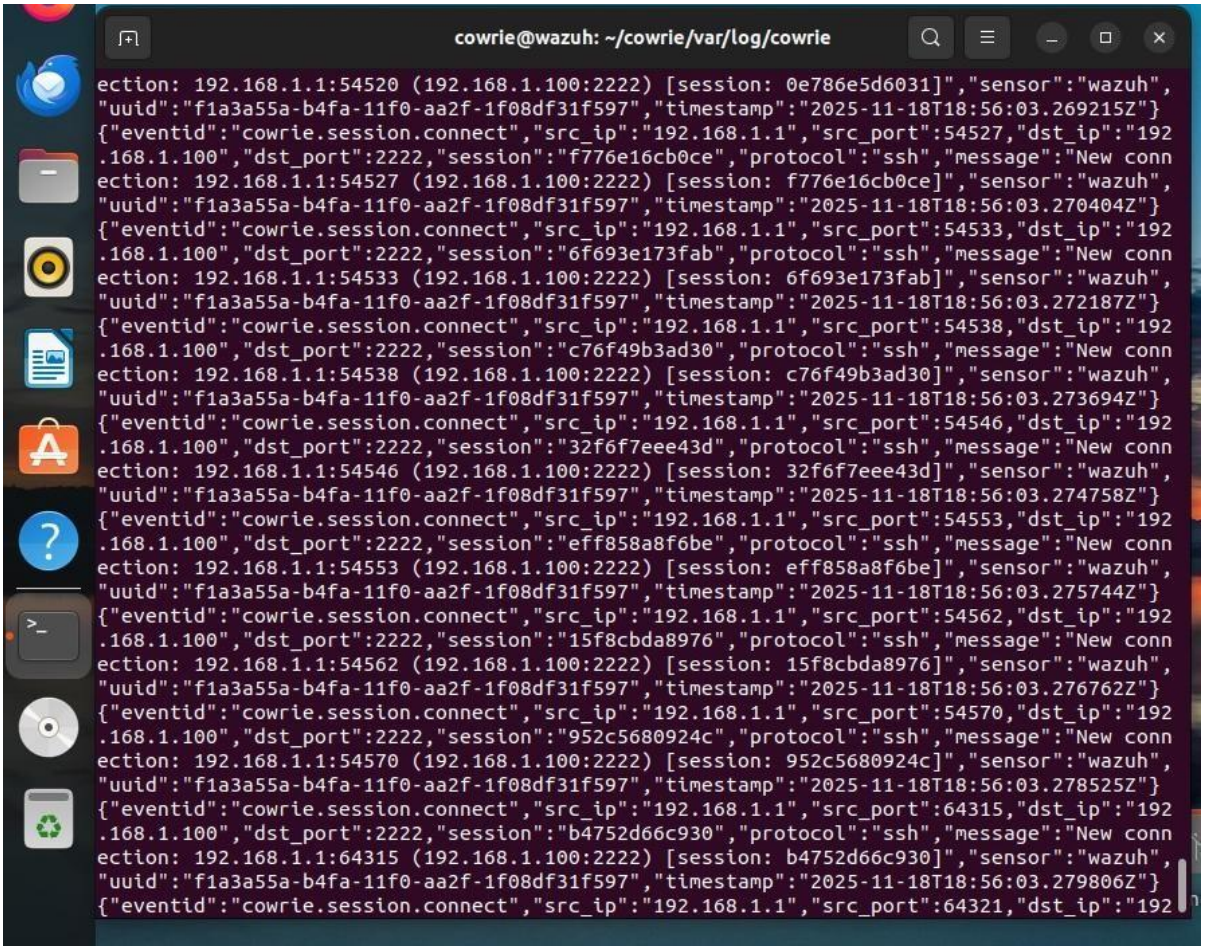


Log *brute force* menampilkan entri seperti:

- *"login attempt [username]: incorrect password"*
- *"failed authentication"*
- *"connection closed after X seconds"*

Terlihat bahwa percobaan *login* dilakukan dalam jumlah besar dan berulang. Honeypot mencatat setiap percobaan, termasuk metode yang digunakan *attacker*. Durasi koneksi menjadi sedikit lebih lama dibanding *port scanning* karena Hydra melakukan percobaan autentikasi setiap koneksi. Penelitian Wahid dkk. [3] menyatakan bahwa log dengan pola seperti ini merupakan salah satu indikator *brute force* yang paling mudah diidentifikasi oleh SIEM.

C. *DDoS*



Log Cowrie untuk SYN *flood* dipenuhi entri yang seragam:

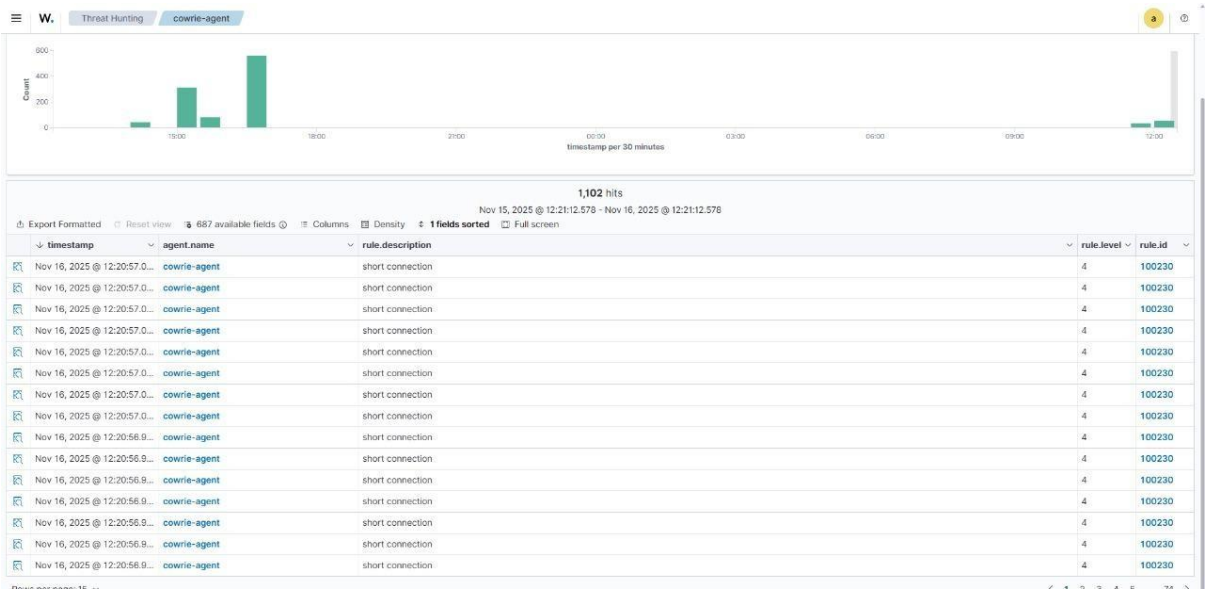
- *"New connection from ..."*
- *Tidak ada aktivitas lanjutan*
- *Tidak ada command*

Entri seperti ini muncul dalam jumlah ribuan dalam waktu sangat singkat. Cowrie mencatatnya sebagai sesi kosong berulang, tanpa adanya *handshake* lanjutan. Hal ini konsisten dengan karakteristik SYN *flood,* yaitu membanjiri server dengan permintaan koneksi palsu tanpa menyelesaikan koneksi TCP. Heluka dkk. [1] menyatakan bahwa pola ini dapat mengakibatkan *overlogging*, tetapi SIEM seperti Wazuh tetap dapat membaca anomali berdasarkan *volume event.*

3.7 Analisis Alert Wazuh

Wazuh menerima log dari honeypot dan memprosesnya menggunakan *rule* yang ada pada Wazuh Manager. Hasil analisis dari dashboard menunjukkan bahwa ketiga serangan menghasilkan *alert* dengan tingkat keparahan yang berbeda.

A. *Scanning* (Nmap)



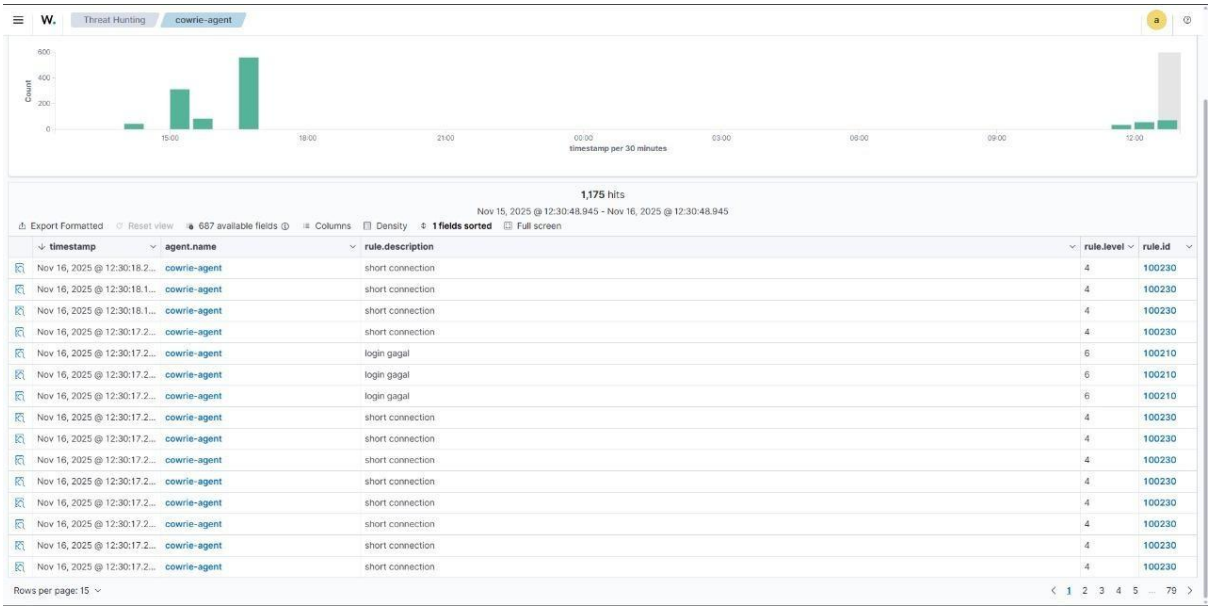
Alert yang muncul biasanya berupa:

- *"Multiple ports scanned"*
- *"Short lived connection detected"*
- *Level: Low – Medium*

Alasan levelnya tidak terlalu tinggi adalah karena *port scanning* dianggap tahap awal serangan dan belum berdampak langsung. Dalam penelitian SIEM [1], *scanning* digolongkan sebagai *reconnaissance-level threat*.

Alert Wazuh juga menunjukkan sumber IP, jumlah *port* yang di-*scan*, dan *timestamp* dari aktivitas tersebut.

B. Brute Force

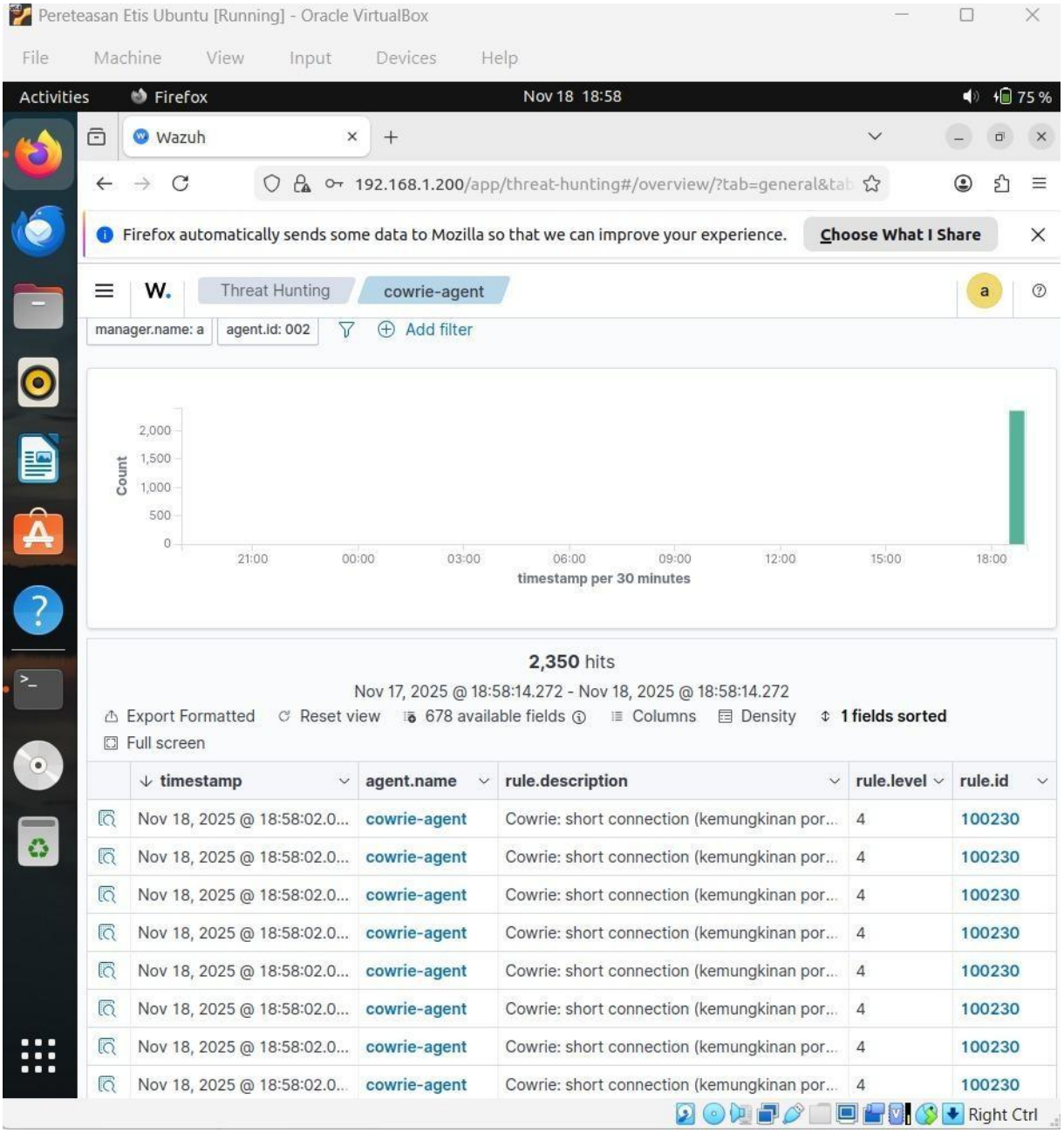


Alert brute force biasanya memiliki level lebih tinggi *(Medium – High*). *Alert* yang muncul antara lain:

- *"Multiple failed login attempts"*
- *"Authentication failure threshold exceeded"*
- *"Possible SSH brute force attack"*

Wazuh melakukan korelasi log berdasarkan jumlah percobaan gagal dalam interval tertentu. Ini sesuai dengan mekanisme deteksi yang dibahas oleh Wahid dkk. [3], bahwa Wazuh dapat mengelompokkan serangan SSH berdasarkan jumlah event berulang dari IP yang sama.

C. *DDoS*



Wazuh memunculkan banyak *alert* seperti:

- *"Large number of connections from single IP"*
- *"High rate of short connections"*

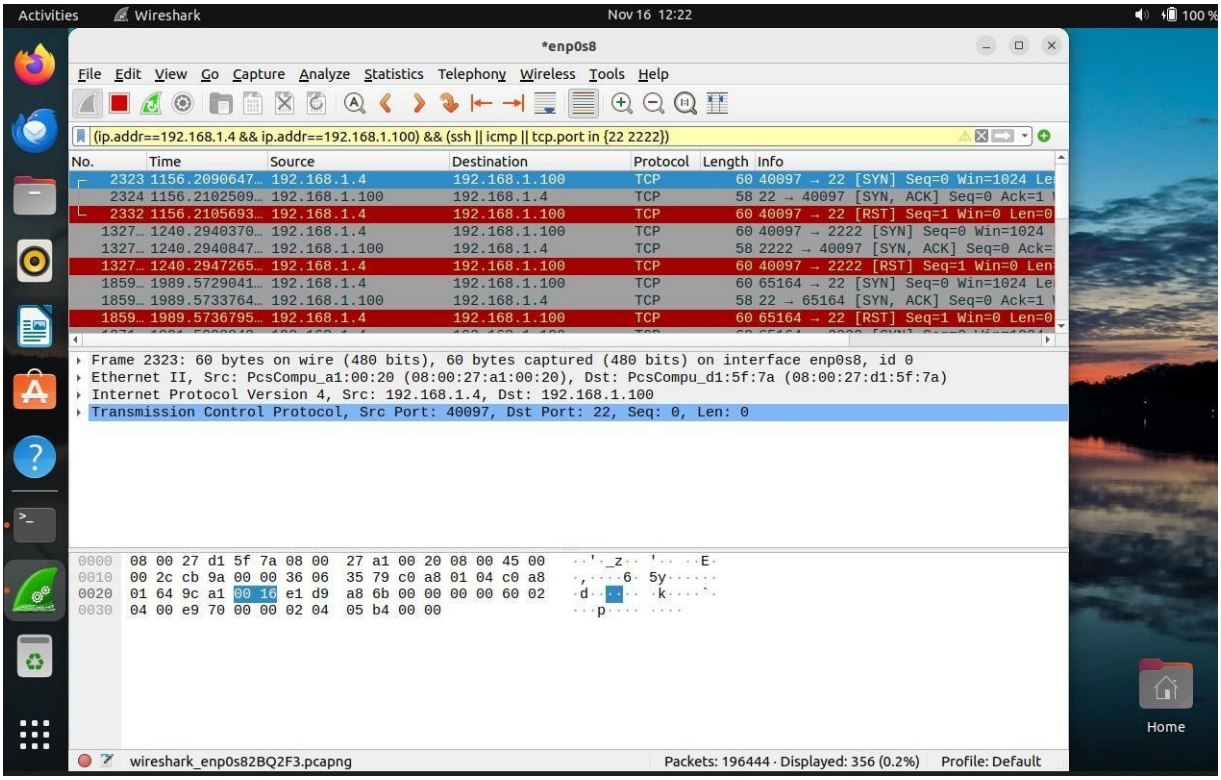
Pada serangan volumetrik, jumlah *alert* dapat menjadi sangat besar. Namun SIEM tetap mampu mengelompokkan *event* berdasarkan frekuensi dan pola waktu.

Penelitian Heluka dan Sulistyo [1] menunjukkan bahwa serangan volumetrik bisa memicu ratusan event SIEM dalam waktu singkat, tetapi tetap terbaca sebagai satu rangkaian serangan.

3.8 Analisis Traffic Wireshark

Wireshark menangkap *traffic* jaringan secara *real-time*, sehingga pola serangan dapat terlihat dari struktur paket yang dikirim *attacker.*

A. *Scanning* (Nmap)

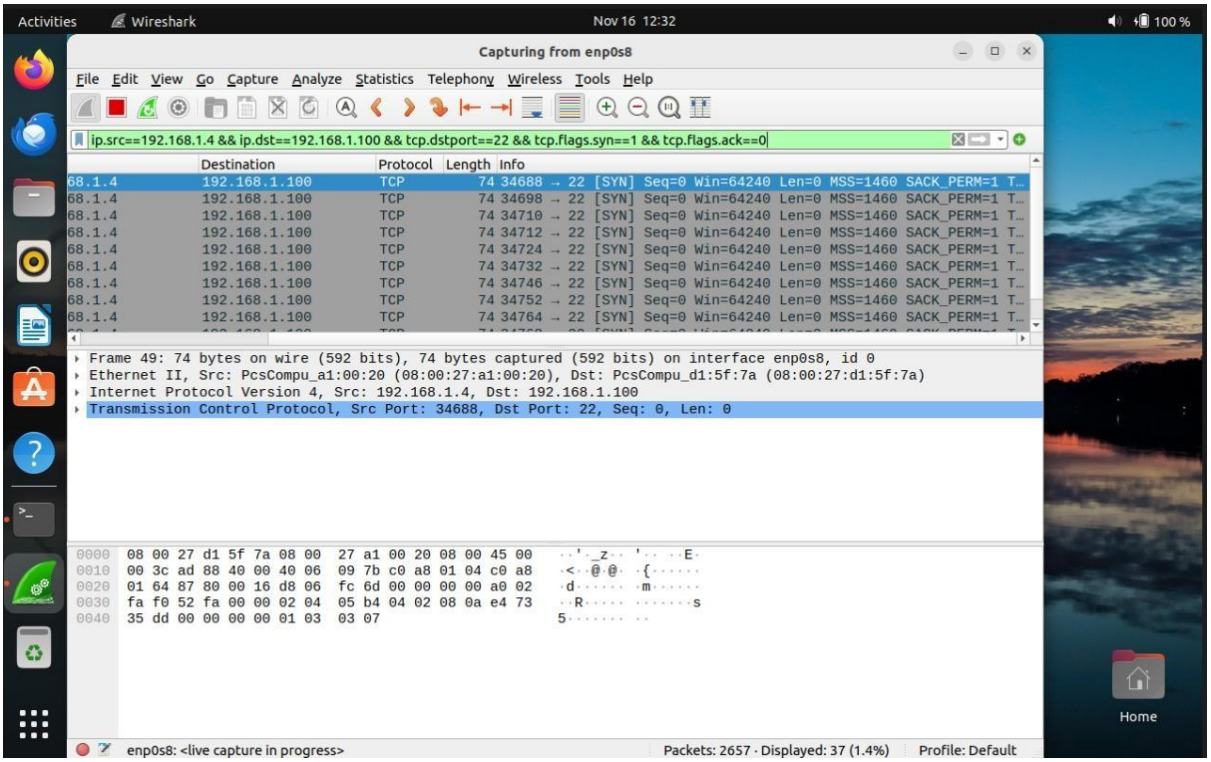


Hasil *capture* menunjukkan:

- Banyak paket SYN ke *port* berbeda.
- Response RST atau SYN-ACK dari honeypot.
- Tidak ada *payload* lanjutan.

Grafik I/O menampilkan puncak kecil tetapi tersebar. Pola *scanning* ini disebut *distributed SYN probing* dan dibahas sebagai tanda reconnaissance awal pada penelitian [1].

B. Brute Force

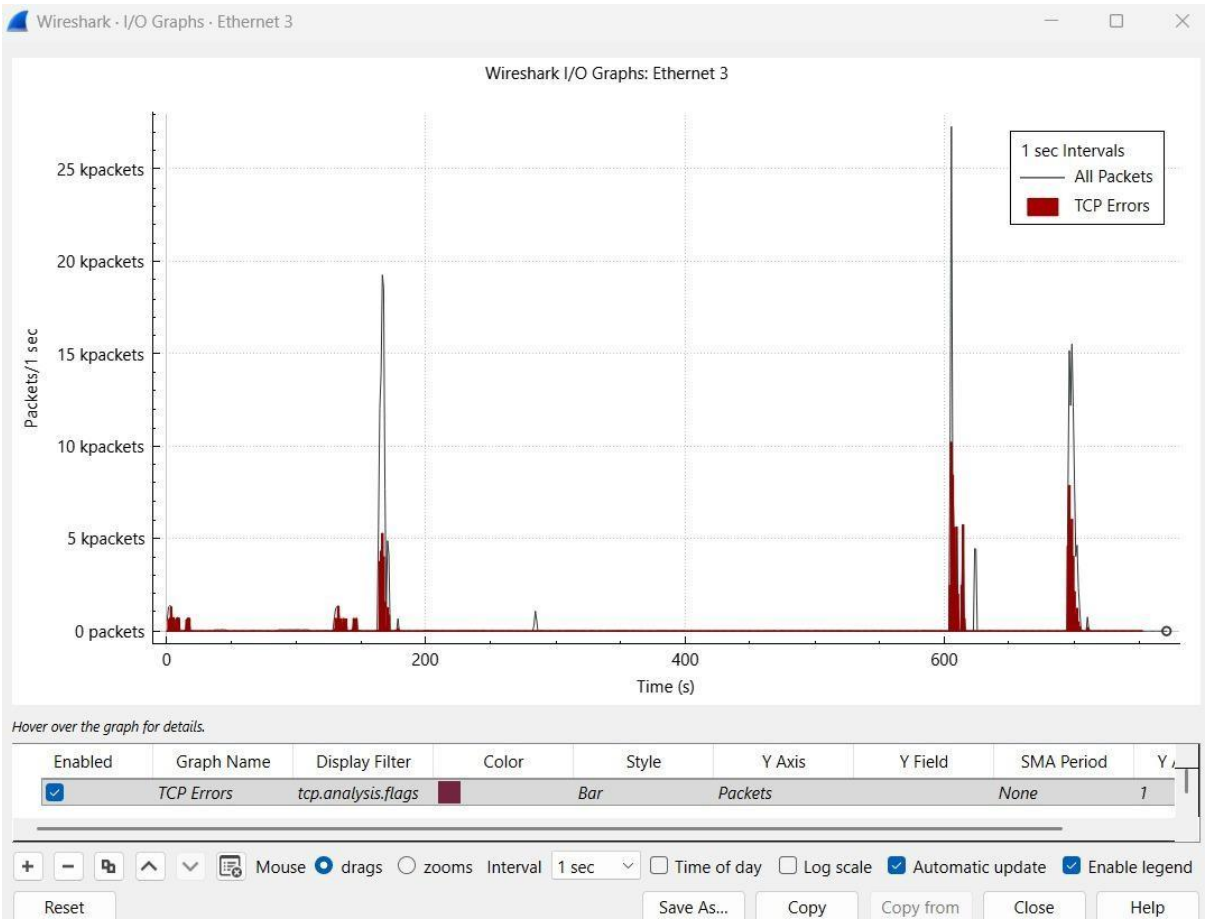


Pola yang terlihat:

- Koneksi berulang ke *port* 22 atau 2222.
- *Handshake* TCP lengkap → diikuti *"encrypted SSH packet"*.
- Terjadi ratusan sesi pendek.

Wireshark menunjukkan bahwa Hydra benar-benar mencoba autentikasi, bukan hanya probing. Setiap koneksi memuat paket SSH seperti *SSH2_MSG_KEXINIT*. Penelitian [2] menjelaskan bahwa *brute force* menghasilkan handshake berulang karena setiap percobaan *login* membutuhkan pembukaan sesi baru.

C. DDoS



Pola yang terlihat pada Wireshark:

- Ribuan paket SYN datang dalam interval mikrodetik.
- Tidak ada ACK dari *attacker (handshake* tidak selesai).
- Grafik I/O menunjukkan spike besar.

Ini sesuai dengan pola serangan volumetrik seperti yang dibahas dalam jurnal [3], di mana SYN *Flood* memanfaatkan eksploitasi state TCP untuk memenuhi *queue* koneksi server.

3.9 Perbandingan Pola Serangan

Ketiga jenis serangan yang diuji yakni *port scanning* menggunakan Nmap, *brute force* SSH menggunakan Hydra, serta serangan DDoS berbasis SYN *Flood* menggunakan Hping3 menunjukkan pola aktivitas yang sangat berbeda pada level log Cowrie, alert Wazuh, maupun *traffic* jaringan di Wireshark. Perbedaan karakteristik ini penting karena setiap jenis serangan memiliki tujuan, teknik, serta dampak yang berbeda pada sistem target. *Port scanning* cenderung menghasilkan koneksi singkat dengan pola *sequential* pada banyak *port,* sementara *brute force* SSH menargetkan satu *port* secara intensif dengan frekuensi *login* yang tinggi. Di sisi lain, SYN *Flood* menghasilkan lalu lintas yang ekstrem karena ribuan paket SYN dikirim dalam waktu yang sangat singkat tanpa melanjutkan proses *handshake*. Analisis komparatif ini memberikan gambaran menyeluruh mengenai bagaimana ketiga *tools* monitoring Cowrie, Wazuh, dan Wireshark saling melengkapi dalam mendeteksi dan memahami karakteristik serangan.

Pada serangan *Port Scanning (Nmap)*, pola serangan terlihat dalam bentuk banyaknya koneksi singkat menuju berbagai *port* yang ada pada honeypot. Cowrie mencatat aktivitas ini sebagai *short connection* karena koneksi dibuka tanpa adanya interaksi lanjutan, sesuai dengan pola scanning khas Nmap. Wazuh, melalui *rule scanning detection*, memberikan *alert* pada tingkat rendah hingga menengah karena aktivitas semacam ini sering muncul dalam fase *reconnaissance* sebelum serangan lebih serius dilakukan. Sementara itu, Wireshark menunjukkan distribusi paket SYN yang tersebar merata pada *port* yang berbeda-beda, namun intensitasnya tidak sepadat serangan *brute force* atau *DDoS*. Melalui pola ini, dapat disimpulkan bahwa *scanning* menampilkan tanda-tanda awal aktivitas penyerangan yang lebih halus dibandingkan dua serangan lainnya.

Pada serangan *Brute Force* SSH (Hydra), pola aktivitas terlihat lebih fokus pada satu *port* tertentu, yaitu port 22 atau *port alternatif* Cowrie seperti 2222. Cowrie mencatat banyak percobaan *login* yang gagal dengan urutan waktu yang sangat berdekatan, menandakan usaha sistematis untuk menebak kredensial autentikasi. Wazuh mengubah entri-entri tersebut menjadi *alert* gagal *login* dan *short connection* dengan level peringatan yang lebih tinggi dibandingkan *port scanning* karena adanya indikasi eksploitasi autentikasi. Wireshark mencatat pola koneksi TCP berulang lengkap dengan tahapan *handshake,* meskipun sering kali ditutup kembali setelah percobaan autentikasi gagal. Secara keseluruhan, brute force memiliki intensitas trafik

yang cukup tinggi namun masih berada dalam pola berurutan dan tidak *sechaotic* serangan DDoS.

Pada serangan SYN *Flood* (Hping3), terlihat pola aktivitas yang paling ekstrem dibandingkan dua jenis serangan lainnya. Cowrie menerima ribuan request koneksi dalam waktu sangat singkat sehingga log dipenuhi dengan entri *session.connect* tanpa adanya *payload* atau interaksi selanjutnya. Wazuh menampilkan lonjakan event terbesar selama seluruh pengujian, dengan dominasi *alert short connection* karena setiap paket SYN dianggap sebagai upaya koneksi yang tidak diselesaikan. Di sisi trafik jaringan, Wireshark memperlihatkan peningkatan tajam pada grafik I/O serta banyaknya paket TCP yang gagal diproses, menandakan adanya tekanan signifikan pada jaringan dan layanan target. Serangan ini secara jelas menggambarkan ciri khas DDoS yang berfokus pada membanjiri server dengan koneksi palsu sehingga menurunkan ketersediaan layanan.

Berdasarkan ketiga pola tersebut, dapat disimpulkan bahwa masing-masing tools yaitu Cowrie, Wazuh, dan Wireshark memberikan sudut pandang yang berbeda namun saling melengkapi dalam menganalisis serangan. Cowrie berperan sebagai sistem pencatat interaksi tingkat aplikasi yang merekam setiap koneksi masuk dengan detail, termasuk command, login attempt, dan session behavior. Wazuh berfungsi sebagai sistem deteksi ancaman yang melakukan korelasi terhadap log tersebut dan mengeluarkan alert berdasarkan rule keamanan yang relevan. Sementara itu, Wireshark memberikan pengamatan langsung terhadap paket jaringan secara real-time sehingga mampu menunjukkan intensitas, volume, dan anomali trafik secara visual. Kombinasi ketiganya memungkinkan analisis serangan dilakukan secara lebih akurat, komprehensif, dan mendalam, memberikan pemahaman yang jelas mengenai karakteristik dan dampak masing-masing jenis serangan.

DAFTAR PUSTAKA

[1] H. Dyan Heluka and W. Sulistyo, “Perancangan Dan Implementasi Security Information and Event Management (SIEM) pada Layanan Virtual Server”.

[2] M. Sofiyan Hadi and D. Afriyantari Puspa Putri, “IMPLEMENTASI SECURITY INFORMATION AND EVENT MANAGEMENT (SIEM) UNTUK DETEKSI DAN ANALISA INSIDEN KEAMANAN PADA WEB SERVER.”

[3] A. Wahid *et al.*, “PKM Pengembangan Sistem Monitoring Keamanan Server Aplikasi SIMLP2M UNM Menggunakan Wazuh”, [Online]. Available: https://journal.diginus.id/index.php/VOKATEK/index