**Laporan Akhir Proyek**

**Implementasi IDS (Intrusion Detection System) dengan   
Snort / Suricata**



Dosen Pengampu :   
Dr. Hendra Kurniawan, S.Kom., M.Sc.Eng

Mata Kuliah :

Perancangan Keamanan Sistem dan Jaringan

Disusun Oleh Kelompok 6 :

2201020107 - Aria Dimas Mastur

2201020121 - Ahmad Zeldiyan

2201020058 - Abdul Arafah

2201020120 - Juandienova Abdullah Sofyan

2201020097 - Fauzan Hanif

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK DAN TEKNOLOGI KEMARITIMAN**

**UNIVERSITAS MARITIM RAJA ALI HAJI (UMRAH)**

**2025**

# **DAFTAR ISI**

[**DAFTAR ISI 2**](#_c7rhkegvv746)

[**DAFTAR GAMBAR 4**](#_lg6w4vlfeq6i)

[**DAFTAR TABEL 5**](#_y1oboe3i4itn)

[**BAB I**](#_etgtv8a7p7zg) [**PENDAHULUAN 6**](#_sd3i9x3sdf4)

[1.1 Latar Belakang 6](#_epsg8x1f27bq)

[1.2 Tujuan Proyek 6](#_inqhd6ds5rdb)

[1.3 Ruang Lingkup Proyek 7](#_uvoh55utek38)

[**BAB II**](#_tawqq1prizm4) [**LANDASAN TEORI 8**](#_o8j77lpq7fyy)

[2.1 Konsep Keamanan Jaringan 8](#_uuax71tlvmj6)

[2.2 Intrusion Detection System (IDS) 8](#_vahyj1hjrsv3)

[2.2.1 Tipe-Tipe IDS 8](#_cnbso1kkj9f9)

[2.2.2 Perbedaan IDS dan IPS 9](#_zfgamvxmgisu)

[2.3 Snort 9](#_uqtmsl545afm)

[2.3.1 Arsitektur Snort 9](#_iauu09tkujgz)

[2.3.2 Aturan (Rules) Snort 9](#_796ydhs9f0d9)

[2.4 Skenario Serangan 10](#_y51xwox00p7t)

[2.4.1 Port Scanning (Nmap) 10](#_ojbxygjtr68d)

[2.4.2 Brute Force Login (Hydra / Medusa) 10](#_bp5nd9f5x6a7)

[2.4.3 Denial of Service (DoS) - ICMP Flood 10](#_8yprdnokoqks)

[2.5 Arsitektur Teoretis Implementasi 10](#_yk3bph9mjvev)

[**BAB III**](#_ud0uur3ivezc) [**PERANCANGAN SISTEM 12**](#_7ou1yjdpaq3w)

[3.1 Topologi Jaringan Virtual 12](#_iqa879tzjtqq)

[3.2 Kebutuhan Perangkat Lunak 12](#_3onv6q6ayky5)

[3.3 Alokasi IP dan Konfigurasi Antarmuka 13](#_eflmkggwwou6)

[3.4 Skenario dan Aturan (Rules) Deteksi 14](#_aoq4iqwn6lkx)

[**BAB IV**](#_n3vjnbphix2y) [**IMPLEMENTASI 15**](#_gl4kvkvm3c5j)

[4.1 Konfigurasi Jaringan VirtualBox (Host-only Network) 15](#_lznyisrjkcjn)

[4.2 Konfigurasi IP Address Statis 17](#_mge3fg1smkss)

[**BAB V**](#_2yjs8mvthyve) [**PENGUJIAN & ANALISIS 28**](#_8nkwsfktrzvw)

[5.1 Skenario Pengujian 1: Port Scanning 28](#_blkii8gqbum2)

[5.1.1 Tujuan Pengujian 28](#_zx3w6lxlh8o)

[5.1.2 Metodologi Pengujian 28](#_d0grp31ju6ge)

[5.1.3 Hasil Pengujian 28](#_jh6tndxbnxl4)

[5.1.4 Analisis Hasil 30](#_yw5vfk5dss6r)

[5.2 Skenario Pengujian 2: Brute Force Login 30](#_56fr3dq1bhz6)

[5.2.1 Tujuan Pengujian 30](#_eg3q4p2l46jz)

[5.2.2 Metodologi Pengujian dan Troubleshooting 30](#_rfufuofeee03)

[5.2.2.1 Percobaan 1: Menggunakan Hydra (Gagal) 31](#_ykbmmsh97pcz)

[5.2.2.2 Percobaan 2: Menggunakan Nmap Scripting Engine (Berhasil) 32](#_z1vw3euhbwen)

[5.2.3 Hasil Pengujian 33](#_e31co1fq6wsz)

[5.2.3.1 Sisi Penyerang (Kali Linux) 33](#_x2rjwqa8br0a)

[5.2.3.2 Sisi IDS (Ubuntu Server) 35](#_mpnvy3hym7jn)

[5.2.4 Analisis Hasil 35](#_6jteo4v6jtv8)

[5.3 Implementasi Custom Rule Lanjutan: Deteksi Serangan DoS 36](#_6cnm3ogfrjep)

[5.3.1 Konfigurasi Aturan Deteksi (Custom Rule) 36](#_m67fgc26b6qo)

[5.3.2 Pengujian Variasi 1: Serangan Menggunakan hping3 37](#_rbu44um0wt92)

[5.3.3 Pengujian Variasi 2: Serangan Menggunakan Ping Flood 38](#_ynhqtdhnmr3w)

[5.3.4 Analisis dan Evaluasi Log 40](#_vlyhd5fngazl)

[**KESIMPULAN 41**](#_dhqgadm3zs32)

[**SARAN PENGEMBANGAN 42**](#_u89cu6o34bin)

[**DAFTAR PUSTAKA 44**](#_qml1ey8kp4e)

# **DAFTAR GAMBAR**

[Gambar 1. Diagram Arsitektur Teoretis Implementasi](#_2ifbbf48z334) 13

[Gambar 2. Topologi Jaringan Virtual Implementasi IDS](#_sg9ea88l9xko) 14

[Gambar 3. Konfigurasi Jaringan VM Kali-Linux](#_wbc8keskrcj4) 18

[Gambar 4. Spesifikasi VM Metasploitable](#_n8oze3f7j0ae) 18

[Gambar 5. Spesifikasi VM Ubuntu Server (ids-server-snort-defender)](#_89t5xy10osps) 19

[Gambar 6. IP Statis Kali-Linux Sebelumnya (MAC 08:00:27:a9:38:80)](#_rqxl1uwt0ica) 21

[Gambar 7. IP Statis Kali-Linux Setelah (192.168.56.10)](#_au2bu6hcdhqy) 22

[Gambar 8. IP Statis Metasploitable Sebelumnya (MAC 08:00:27:a9:38:80)](#_xz1tq692np6) 23

[Gambar 9. IP Statis Metasploitable Setelah (192.168.56.20)](#_nhcz41osd7ve) 23

[Gambar 10. IP Statis Ubuntu Server Sebelumnya (MAC 08:00:27:95:89:0b)](#_ck2qpuiiwxkz) 24

[Gambar 11. IP Statis Ubuntu Server Setelah (192.168.56.30)](#_76k0rs7agoab) 24

[Gambar 12. Mengatur Variabel Jaringan (HOME\_NET)](#_2vj5v1hqu26k) 25

[Gambar 13. Mengaktifkan Aturan Deteksi Port Scan](#_3efzavgjnssg) 25

[Gambar 14. Validasi Konfigurasi](#_jbb1j7dfaqap) 27

[Gambar 15. Memasukkan Aturan](#_362cey68v46z) 29

[Gambar 16. Snort successfully validated the configuration](#_fwama5mu6hu3) 29

[Gambar 17. Output Nmap pada Kali Linux menunjukkan daftar port terbuka pada target.](#_66u3xpapqift) 31

[Gambar 18. Log deteksi Snort menunjukkan adanya upaya scanning dan permintaan koneksi yang mencurigakan.](#_6wfwb85utn40) 32

[Gambar 19. Percobaan serangan menggunakan Hydra gagal akibat kesalahan pertukaran kunci (kex error).](#_vallyegy2x16) 34

[Gambar 20. Output Nmap menunjukkan proses percobaan login (brute force) yang masif dan ditemukannya kredensial valid.](#_jtilh4xznm8e) 36

[Gambar 21. Output Nmap menunjukkan proses percobaan login (brute force) yang masif](#_abv551esegbr) 37

[Gambar 22. Output Nmap menunjukkan proses percobaan login (brute force) yang masif](#_syrm8su34dlo) 38

[Gambar 23. Log Snort menampilkan peringatan "BAHAYA: Serangan Brute Force SSH Terdeteksi" sebagai respons terhadap serangan.](#_54ux49okyac9) 38

[Gambar 24. Konfigurasi penambahan aturan deteksi ICMP Flood pada file local.rules](#_f87rspq0q7cl) 40

[Gambar 25. Eksekusi serangan flooding menggunakan hping3 pada Kali Linux](#_3ttwshkprr76) 40

[Gambar 26. Log Snort mendeteksi pola serangan hping3 sebagai aktivitas 'ICMP PING NMAP'](#_snpii5x5kymk) 41

# **DAFTAR TABEL**

[Tabel 1. Kebutuhan Perangkat Lunak](#_lasnbwx2rpx5) 15

[Tabel 2. Alokasi IP dan Konfigurasi Antarmuka](#_8hvrf51hxmdo) 16

[Tabel 3. Skenario dan Aturan (Rules) Deteksi](#_hew9e9ddvcrm) 17

# **BAB I**

# **PENDAHULUAN**

## 1.1 Latar Belakang

Keamanan jaringan merupakan aspek krusial dalam infrastruktur teknologi informasi modern. Seiring dengan meningkatnya jumlah perangkat yang terhubung ke internet, volume dan kompleksitas ancaman siber juga turut meningkat secara signifikan [1]. Serangan seperti *Denial-of-Service* (DoS), pencurian data, dan akses tidak sah menjadi risiko harian yang dihadapi oleh berbagai organisasi. Oleh karena itu, mekanisme pertahanan proaktif sangat diperlukan untuk melindungi integritas, kerahasiaan, dan ketersediaan data.

Sistem pertahanan tradisional seperti *firewall* berfungsi sebagai garda terdepan dengan menyaring lalu lintas berdasarkan aturan statis, namun seringkali tidak cukup untuk mendeteksi ancaman yang lebih canggih yang berhasil lolos. Di sinilah peran *Intrusion Detection System* (IDS) menjadi sangat penting. IDS adalah perangkat lunak atau perangkat keras yang dirancang untuk memantau lalu lintas jaringan atau aktivitas sistem untuk mencari tanda-tanda aktivitas berbahaya atau pelanggaran kebijakan [2].

Snort adalah salah satu platform IDS berbasis *open-source* yang paling banyak digunakan di dunia. Keunggulan utamanya terletak pada fleksibilitasnya dalam mendeteksi ancaman menggunakan seperangkat aturan (rules) yang dapat disesuaikan [3]. Snort mampu melakukan analisis lalu lintas secara *real-time* dan mencatat paket data untuk mendeteksi berbagai macam pola serangan.

Dalam proyek akhir ini, implementasi IDS Snort akan dilakukan dalam lingkungan jaringan virtual. Proyek ini akan berfokus pada konfigurasi Snort untuk mengenali pola serangan spesifik yang umum terjadi, yaitu *port scanning* (pemindaian port) dan *brute force login*, yang sering menjadi langkah awal bagi peretas sebelum melakukan intrusi lebih jauh [4].

## 1.2 Tujuan Proyek

Berdasarkan latar belakang yang telah diuraikan, tujuan dari pengerjaan proyek ini adalah sebagai berikut:

1. Mengimplementasikan server IDS menggunakan Snort pada sistem operasi Linux Ubuntu di dalam lingkungan jaringan virtual (VirtualBox).
2. Melakukan konfigurasi *interface* jaringan dan aturan dasar (*basic rules*) pada Snort agar dapat memonitor lalu lintas jaringan.
3. Melakukan simulasi skenario serangan, khususnya *port scanning* menggunakan Nmap dan *brute force login* menggunakan Hydra/Medusa.
4. Menganalisis file *log* yang dihasilkan oleh Snort untuk memvalidasi bahwa sistem berhasil mendeteksi simulasi serangan yang dilakukan.

## 1.3 Ruang Lingkup Proyek

Untuk memastikan pengerjaan proyek tetap fokus dan sesuai dengan tujuan, ruang lingkup pekerjaan dibatasi sebagai berikut:

Batasan Pekerjaan (In-Scope) :

1. Lingkungan implementasi terbatas pada jaringan virtual di VirtualBox yang terdiri dari tiga mesin: mesin penyerang (Kali Linux), mesin target (Metasploitable2), dan mesin IDS (Ubuntu Server).
2. Perangkat lunak IDS yang digunakan adalah Snort (bukan Suricata).
3. Skenario serangan yang diuji terbatas pada *port scanning* dan *brute force login* sesuai panduan proyek.
4. Fokus utama adalah analisis *log* hasil deteksi untuk membuktikan keberhasilan IDS.

Di Luar Batasan Pekerjaan (Out-of-Scope) :

1. Proyek ini hanya berfokus pada *Intrusion Detection* (Deteksi), bukan *Intrusion Prevention* (Pencegahan). Snort tidak dikonfigurasi dalam mode *inline* (IPS) untuk memblokir serangan secara aktif.
2. Proyek tidak mencakup penulisan *custom rule* yang kompleks dari awal, melainkan penyesuaian *rule* dasar yang ada untuk skenario pengujian.
3. Proyek tidak menguji semua jenis serangan keamanan jaringan, melainkan hanya yang telah ditentukan.

# **BAB II**

# **LANDASAN TEORI**

## 2.1 Konsep Keamanan Jaringan

Keamanan jaringan adalah serangkaian praktik dan teknologi yang dirancang untuk melindungi jaringan komputer dan data dari akses tidak sah, penyalahgunaan, atau kerusakan. Tujuan utamanya adalah untuk menjamin tiga prinsip dasar yang dikenal sebagai CIA Triad (Confidentiality, Integrity, Availability).

1. Confidentiality (Kerahasiaan): Memastikan bahwa data hanya dapat diakses oleh pihak yang berwenang.
2. Integrity (Integritas): Menjamin bahwa data tetap akurat dan utuh, serta tidak diubah oleh pihak yang tidak sah.
3. Availability (Ketersediaan): Memastikan bahwa data dan layanan jaringan selalu tersedia dan dapat diakses oleh pengguna yang sah saat dibutuhkan.

Dalam praktiknya, keamanan jaringan bersifat berlapis (*Defense in Depth*), di mana tidak ada satu alat tunggal yang dapat melindungi dari semua ancaman. Sebuah *firewall* mungkin menjadi lapisan pertahanan pertama, namun sistem seperti IDS diperlukan untuk mendeteksi ancaman yang berhasil melewati lapisan tersebut [5].

## 2.2 Intrusion Detection System (IDS)

*Intrusion Detection System* (IDS) adalah sebuah sistem, baik berupa perangkat keras maupun perangkat lunak, yang bertugas memantau aktivitas jaringan atau sistem untuk mendeteksi aktivitas berbahaya atau pelanggaran kebijakan keamanan. Berbeda dengan *firewall* yang secara aktif memblokir lalu lintas berdasarkan aturan, IDS pada dasarnya bersifat pasif tugas utamanya adalah mendeteksi dan memberikan peringatan (alert) kepada administrator sistem ketika potensi intrusi teridentifikasi [2].

### 2.2.1 Tipe-Tipe IDS

Berdasarkan lingkup pemantauannya, IDS secara umum dibagi menjadi dua kategori utama:

1. Network-based Intrusion Detection System (NIDS): NIDS memantau seluruh lalu lintas data yang mengalir pada suatu segmen jaringan. NIDS bekerja dengan menganalisis paket-paket data secara *real-time* untuk mencari pola yang cocok dengan *signature* serangan. Implementasi Snort dalam proyek ini termasuk dalam kategori NIDS.
2. Host-based Intrusion Detection System (HIDS): HIDS berjalan pada satu *host* (komputer) individual dan memantau aktivitas internal pada *host* tersebut, seperti *log file*, modifikasi file sistem kritis, dan panggilan sistem (*system calls*).

### 2.2.2 Perbedaan IDS dan IPS

Seringkali, IDS dikaitkan dengan *Intrusion Prevention System* (IPS). Perbedaan mendasar keduanya terletak pada respons terhadap ancaman:

1. IDS (Deteksi): Bekerja secara *out-of-band* (tidak berada di jalur data utama). IDS hanya membaca salinan lalu lintas dan mengirimkan *alert* jika menemukan ancaman.
2. IPS (Pencegahan): Bekerja secara *in-line* (berada di jalur data utama). IPS dapat secara aktif memblokir paket data yang dianggap berbahaya sebelum mencapai target.

Proyek ini berfokus pada implementasi IDS untuk deteksi dan analisis log.

## 2.3 Snort

Snort adalah sebuah NIDS *open-source* yang sangat populer dan kuat. Snort menggunakan mekanisme deteksi berbasis aturan (*rule-based*) untuk menganalisis lalu lintas jaringan. Ia mampu mendeteksi berbagai jenis serangan, seperti *port scanning*, *buffer overflows*, serangan CGI, dan banyak lagi, dengan mencocokkan lalu lintas terhadap *database* aturan yang telah ditentukan [3].

### 2.3.1 Arsitektur Snort

Secara sederhana, Snort bekerja melalui beberapa komponen utama :

1. Packet Decoder: Mengambil paket dari berbagai jenis antarmuka jaringan dan mempersiapkannya untuk dianalisis.
2. Preprocessors: Komponen *plug-in* yang mengatur dan menormalkan data paket sebelum dianalisis oleh mesin deteksi (misalnya, menata ulang fragmen paket TCP).
3. Detection Engine: Komponen inti yang menerapkan aturan (*rules*) Snort terhadap paket data. Ini adalah tempat di mana deteksi intrusi terjadi.
4. Logging and Alerting System: Komponen yang bertanggung jawab untuk mencatat hasil deteksi (membuat *log file* ) dan/atau mengirimkan peringatan.

### 2.3.2 Aturan (Rules) Snort

Kekuatan utama Snort terletak pada aturannya. Setiap aturan terdiri dari dua bagian logis: *Rule Header* dan *Rule Options*.

1. Rule Header: Mendefinisikan aksi aturan (misal: alert), protokol (TCP/UDP/ICMP), alamat IP sumber dan tujuan, serta *port* sumber dan tujuan.
2. Rule Options: Berisi bagian yang lebih spesifik tentang apa yang harus dicari di dalam paket (misal: *payload* data tertentu, pesan, atau *flag* TCP) untuk memicu aturan tersebut.

## 2.4 Skenario Serangan

Proyek ini akan menguji efektivitas Snort dalam mendeteksi dua jenis aktivitas berbahaya yang umum dilakukan pada tahap pengintaian dan serangan awal:

### 2.4.1 Port Scanning (Nmap)

*Port scanning* adalah teknik yang digunakan penyerang untuk mengidentifikasi *port* (pintu) apa saja yang terbuka pada sebuah sistem target. Dengan mengetahui *port* yang terbuka, penyerang dapat mengetahui layanan apa yang berjalan (misal: Web Server di *port* 80, SSH di *port* 22) dan merencanakan serangan lebih lanjut. Nmap adalah alat yang paling umum digunakan untuk melakukan *port scanning*. Snort memiliki aturan bawaan untuk mendeteksi berbagai jenis teknik *scanning* yang dilakukan oleh Nmap.

### 2.4.2 Brute Force Login (Hydra / Medusa)

Serangan *brute force* adalah upaya untuk mendapatkan akses tidak sah ke sebuah akun (misalnya SSH atau login web) dengan cara mencoba semua kemungkinan kombinasi *username* dan *password* secara sistematis dan berulang kali [4]. Hydra dan Medusa adalah alat populer yang mengotomatisasi serangan ini. IDS yang dikonfigurasi dengan baik dapat mendeteksi lonjakan upaya *login* yang gagal dalam waktu singkat dari satu alamat IP, yang merupakan indikator kuat dari serangan *brute force* [7].

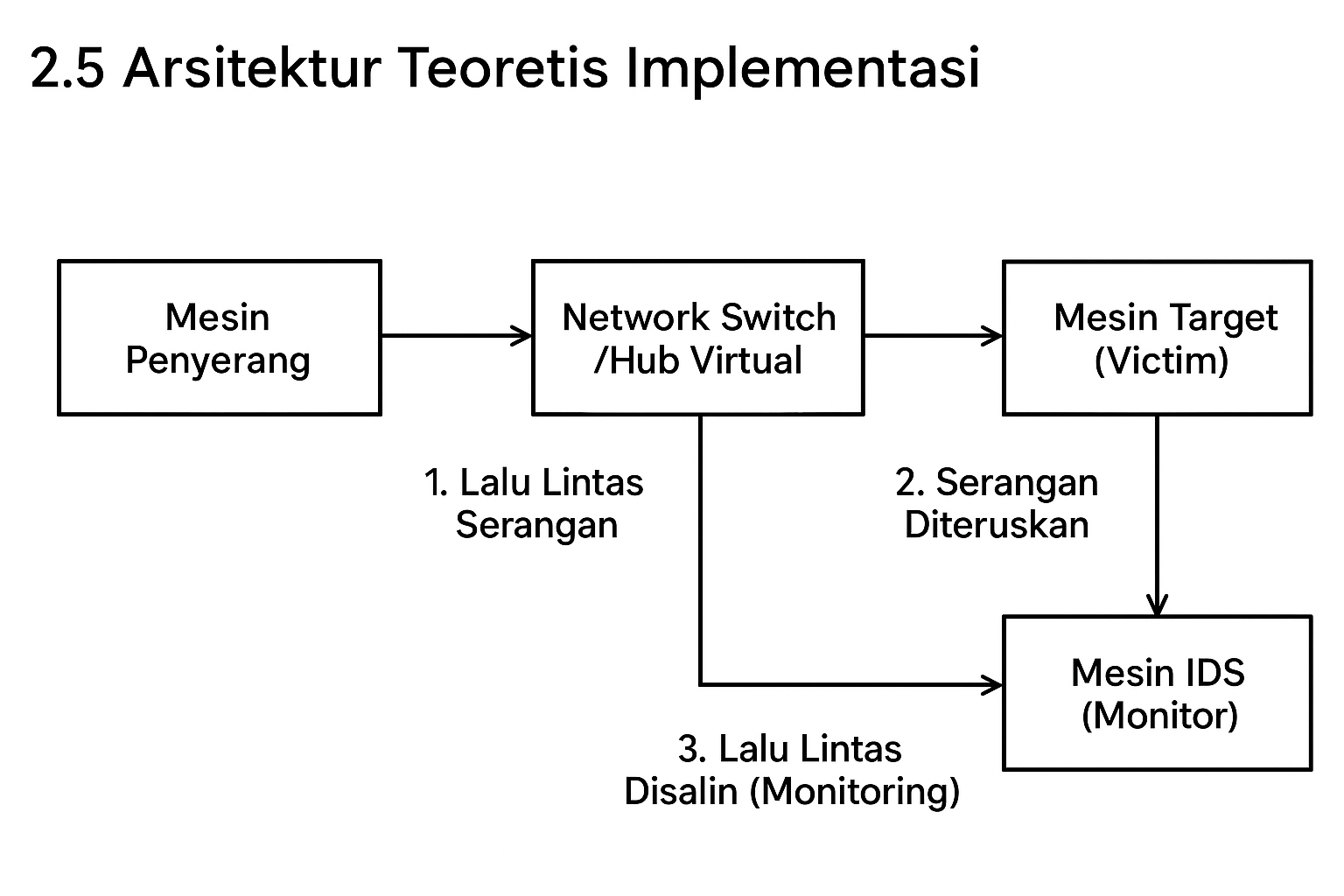
### 2.4.3 Denial of Service (DoS) - ICMP Flood

Serangan *Denial of Service* (DoS) bertujuan untuk membuat sumber daya sistem atau jaringan tidak tersedia bagi pengguna yang sah. ICMP Flood (atau *Ping Flood*) adalah teknik DoS di mana penyerang membanjiri target dengan paket *ICMP Echo Request* dalam jumlah masif dan cepat. Tujuannya adalah menghabiskan *bandwidth* jaringan atau membebani CPU target sehingga tidak mampu merespons permintaan lain. Alat seperti hping3 atau perintah ping flood digunakan untuk mensimulasikan serangan volumetrik ini.

## 2.5 Arsitektur Teoretis Implementasi

Untuk memenuhi tujuan proyek, arsitektur jaringan virtual akan dirancang untuk mensimulasikan skenario serangan dan deteksi. Arsitektur ini terdiri dari tiga komponen utama yang terisolasi dalam lingkungan VirtualBox/VMware:

1. Mesin Penyerang (Attacker): Menggunakan OS Kali Linux, yang berisi alat-alat serangan seperti Nmap dan Hydra.
2. Mesin Target (Victim): Menggunakan OS Metasploitable2, sebuah sistem operasi yang sengaja dibuat rentan untuk tujuan pengujian keamanan.
3. Mesin IDS (Defender): Menggunakan OS Ubuntu Server yang telah diinstal Snort. Mesin ini akan dikonfigurasi untuk memonitor lalu lintas yang ditujukan ke Mesin Target.



##### Gambar 1. Diagram Arsitektur Teoretis Implementasi

Diagram di atas mengilustrasikan bagaimana server IDS (Snort) akan ditempatkan pada jaringan untuk memonitor lalu lintas antara Penyerang dan Target. Seluruh lalu lintas dari penyerang akan disalin dan dianalisis oleh Snort secara bersamaan saat lalu lintas tersebut dikirimkan ke target, memungkinkan Snort untuk mencatat aktivitas berbahaya sesuai aturan yang berlaku.

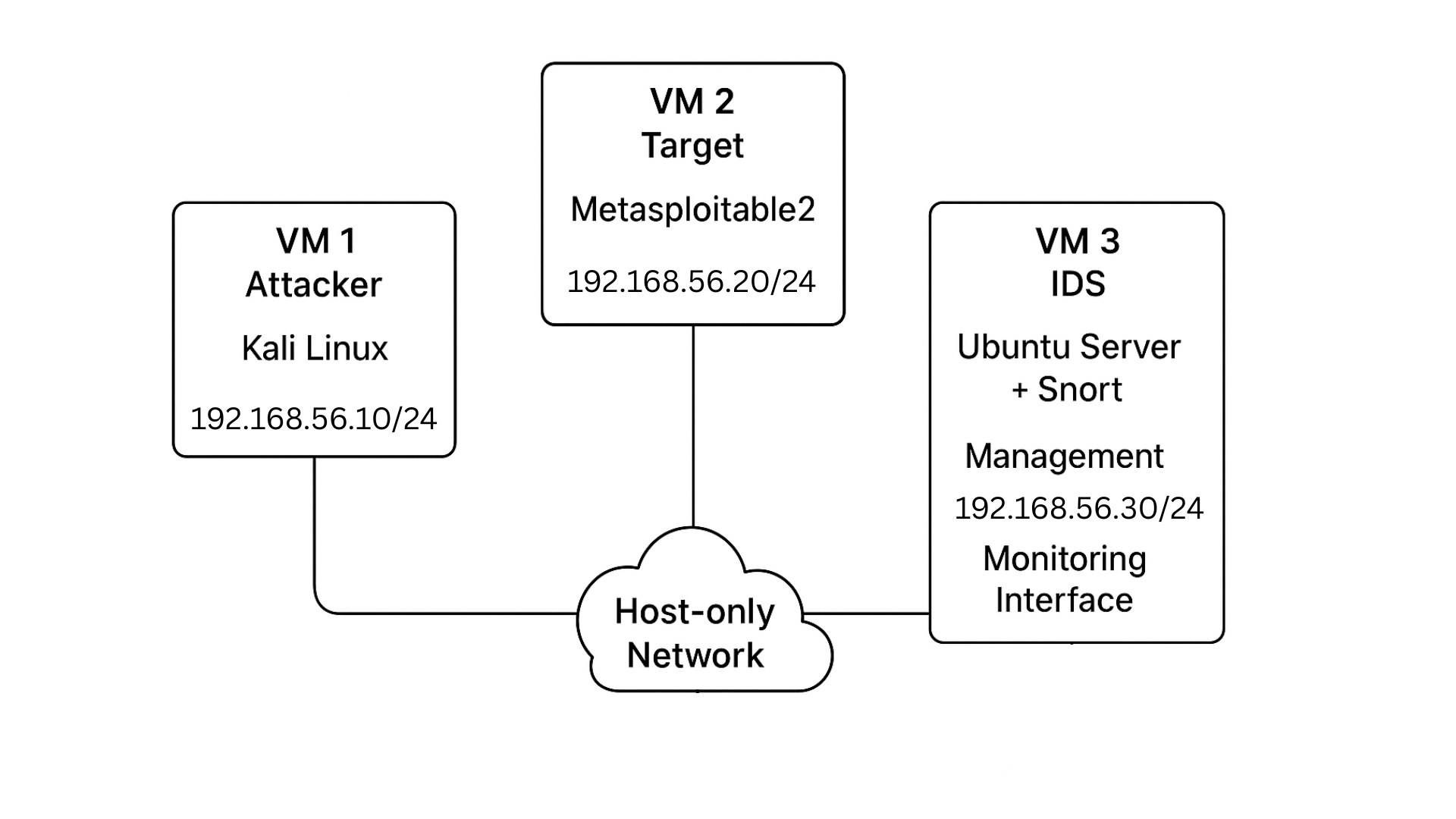
# **BAB III**

# **PERANCANGAN SISTEM**

## 3.1 Topologi Jaringan Virtual

Topologi jaringan yang dirancang untuk proyek ini mengacu pada arsitektur teoretis yang dijelaskan dalam BAB 2 . Seluruh implementasi dilakukan dalam lingkungan virtual menggunakan Oracle VirtualBox. Tiga mesin virtual (VM) dihubungkan ke satu jaringan internal yang sama, yaitu "Host-only Network". Penggunaan jaringan *host-only* memastikan bahwa ketiga VM dapat saling berkomunikasi dalam jaringan yang terisolasi, sekaligus memungkinkan server IDS untuk memonitor lalu lintas di antara mereka.

Diagram topologi jaringan yang digunakan adalah sebagai berikut :



##### Gambar 2. Topologi Jaringan Virtual Implementasi IDS

## 3.2 Kebutuhan Perangkat Lunak

Untuk membangun lingkungan pengujian sesuai topologi di atas, perangkat lunak berikut digunakan:

|  |  |  |
| --- | --- | --- |
| Kategori | Perangkat Lunak | Peran / Keterangan |
| Virtualisasi | Oracle VirtualBox | Platform untuk menjalankan semua mesin virtual. |
| OS Penyerang | Kali Linux | Berisi alat uji (Nmap, Hydra). |
| OS Target | Metasploitable2 | Sistem target yang memiliki kerentanan. |
| OS IDS | Ubuntu Server 24.04 LTS | Sistem operasi untuk instalasi Snort. |
| Software IDS | Snort | Perangkat lunak NIDS yang akan dikonfigurasi. |

##### Tabel 1. Kebutuhan Perangkat Lunak

## 3.3 Alokasi IP dan Konfigurasi Antarmuka

Pengalamatan IP dan konfigurasi antarmuka jaringan untuk setiap VM dirancang secara spesifik untuk memenuhi kebutuhan fungsionalnya. Jaringan yang digunakan adalah 192.168.56.0/24 (ini adalah *range* IP umum untuk "Host-only Network" di VirtualBox).

| VM (Hostname) | Peran | OS | Adapter VirtualBox | Nama Interface (di OS) | Alamat IP / Netmask | Keterangan |
| --- | --- | --- | --- | --- | --- | --- |
| VM 1 (Kali) | Penyerang | Kali Linux | Host-only Adapter | eth0 | http://192.168.56.10/24 | IP statis untuk mesin penyerang. |
| VM 2 (Metasploitable) | Target | Metasploitable2 | Host-only Adapter | eth0 | http://192.168.56.20/24 | IP statis untuk mesin target. |
| VM 3 (ids-server-snort-defender) | IDS Server | Ubuntu Server | Adapter 1 (Host-only) | enp0s3 | http://192.168.56.30/24 | Antarmuka untuk manajemen (via SSH). |
|  |  |  | Adapter 2 (Host-only) | enp0s8 | - | Antarmuka untuk monitoring (mode promiscuous). |

##### Tabel 2. Alokasi IP dan Konfigurasi Antarmuka

Catatan Penting untuk VM 3 (IDS Server):

1. Server IDS sengaja dikonfigurasi dengan dua antarmuka jaringan.
2. Adapter 1 (enp0s3) digunakan untuk manajemen server (seperti login SSH dari komputer *host*).
3. Adapter 2 (enp0s8) akan dikonfigurasi dalam mode *promiscuous* (mode campur). Antarmuka ini tidak memiliki alamat IP dan hanya bertugas "mengendus" atau mendengarkan semua lalu lintas paket yang lewat di "Host-only Network" . Snort akan dijalankan untuk memonitor antarmuka ini.

## 3.4 Skenario dan Aturan (Rules) Deteksi

Perancangan ini berfokus pada dua skenario pengujian utama sesuai panduan proyek.

##### Tabel 3. Skenario dan Aturan (Rules) Deteksi

| NO | Skenario Pengujian | Deskripsi Skenario | Aturan/Konfigurasi Snort yang Digunakan |
| --- | --- | --- | --- |
| 1 | Port Scanning (Nmap) | Penyerang (VM 1) menjalankan Nmap untuk memindai port yang terbuka pada mesin Target (VM 2). | Mengaktifkan preprocessor deteksi port scan bawaan Snort untuk mendeteksi berbagai jenis pemindaian |
| 2 | Brute Force Login (Hydra) | Penyerang (VM 1) menjalankan Hydra untuk melakukan serangan brute force (mencoba banyak password) ke layanan SSH (port 22) di mesin Target (VM 2). | Membuat aturan kustom di file local.rules yang akan memicu alert jika terjadi lebih dari 5 upaya koneksi gagal ke port 22 dari IP yang sama dalam 60 detik. |
| 3 | Denial of Service (DoS) - ICMP Flood | Penyerang (VM 1) membanjiri jaringan Target dengan paket ICMP (Ping) secara masif menggunakan alat hping3 atau ping flood. | Membuat aturan kustom dengan parameter threshold di file local.rules. Alert akan muncul jika terdeteksi lebih dari 100 paket ICMP dari satu sumber dalam waktu 10 detik. |

# **BAB IV**

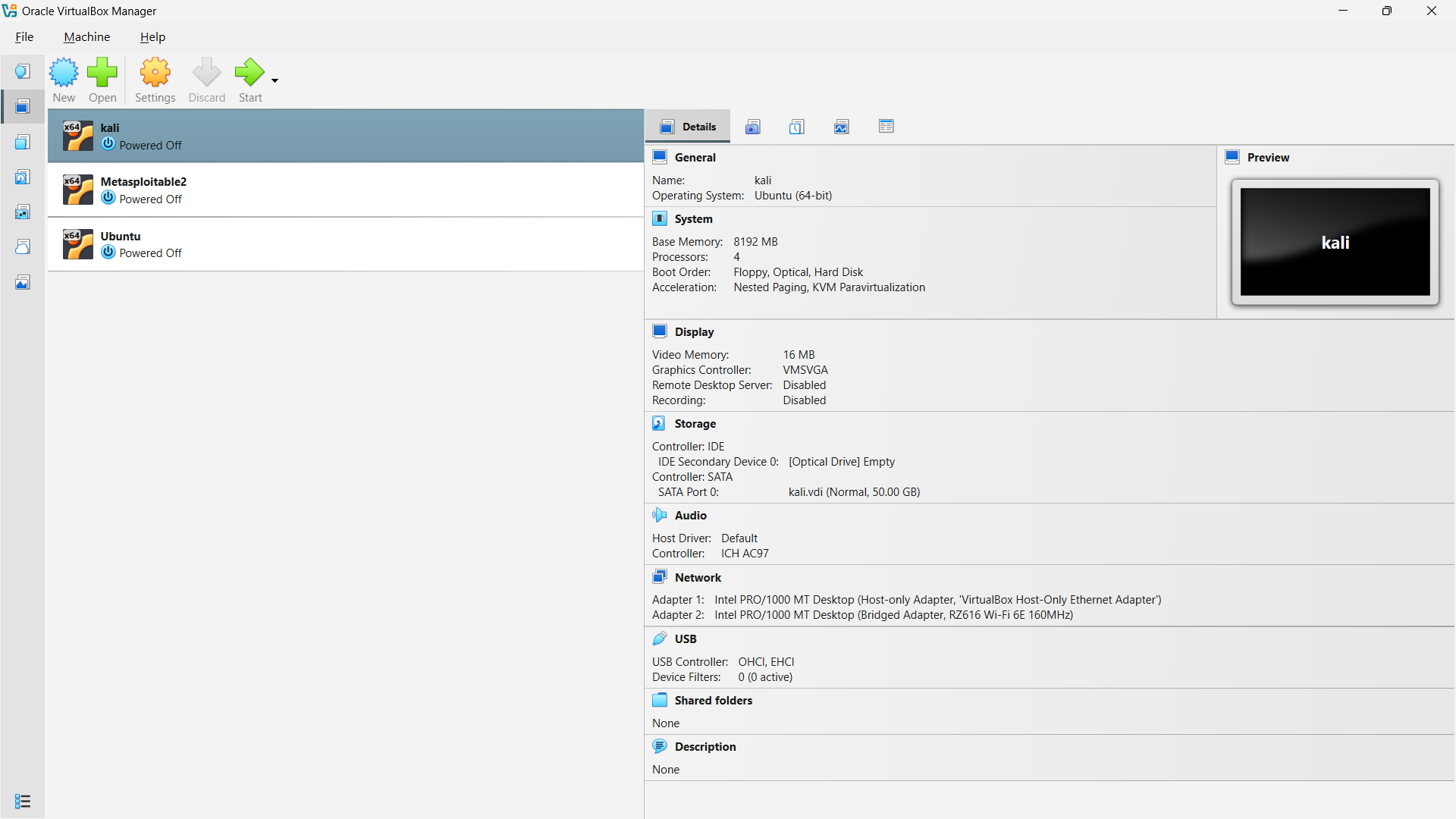
# **IMPLEMENTASI**

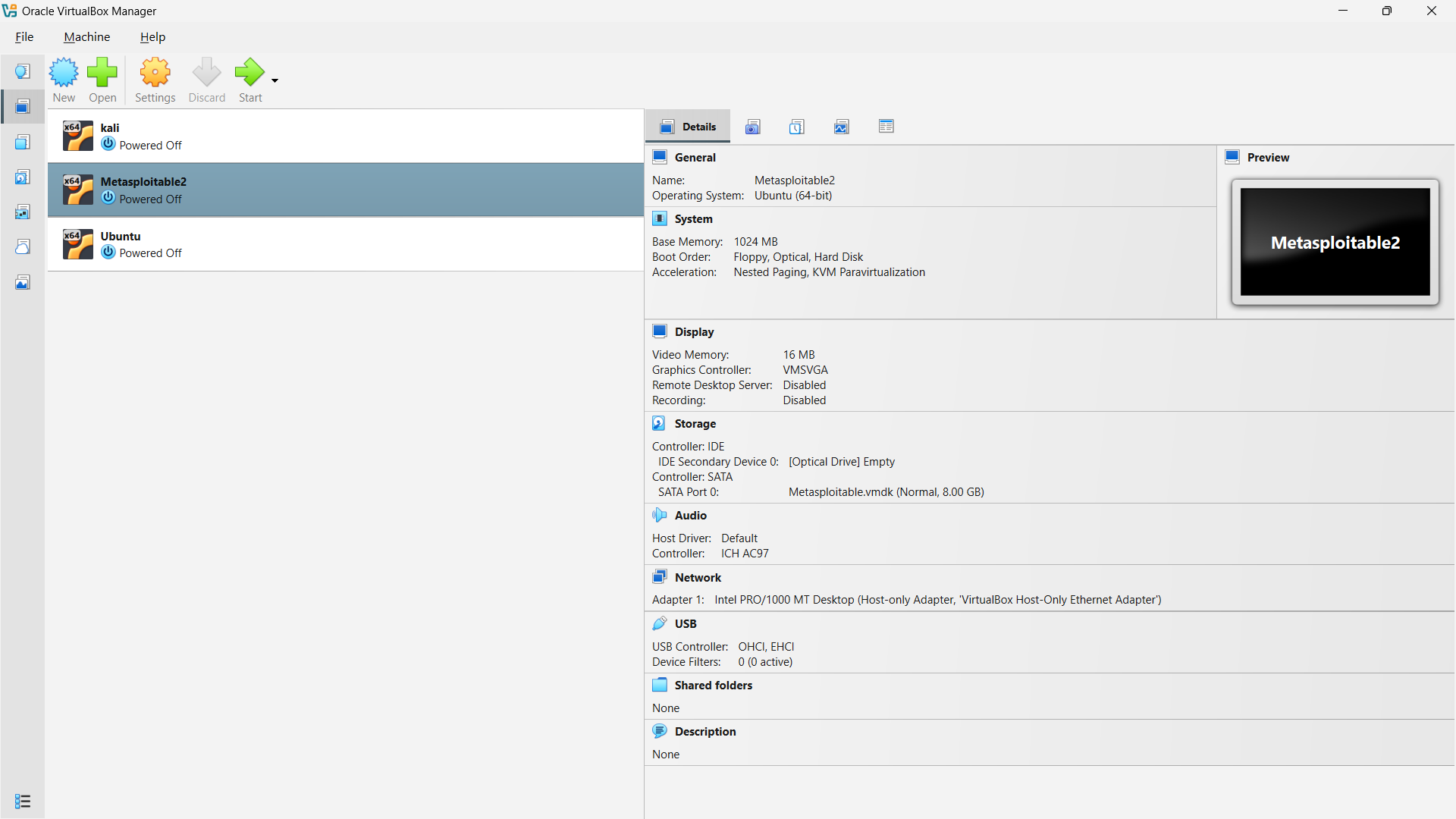
## 4.1 Konfigurasi Jaringan VirtualBox (Host-only Network)

Langkah pertama adalah memastikan ketiga VM terhubung ke "kabel LAN virtual" yang sama. Kita akan menggunakan "Host-only Adapter" bawaan VirtualBox (vboxnet0) yang sesuai dengan rancangan 192.168.56.0/24.

1. Pengaturan VM 1 (Kali Linux) & VM 2 (Metasploitable2):
   1. Matikan kedua VM.
   2. Pilih VM, klik Settings > Network.
   3. Pada Adapter 1, ubah "Attached to" menjadi Host-only Adapter.
   4. Pilih nama adapter VirtualBox Host-Only Ethernet Adapter.
   5. Buka bagian Advanced, set "Promiscuous Mode" menjadi Allow All. Ini penting agar *switch* virtual meneruskan semua paket.

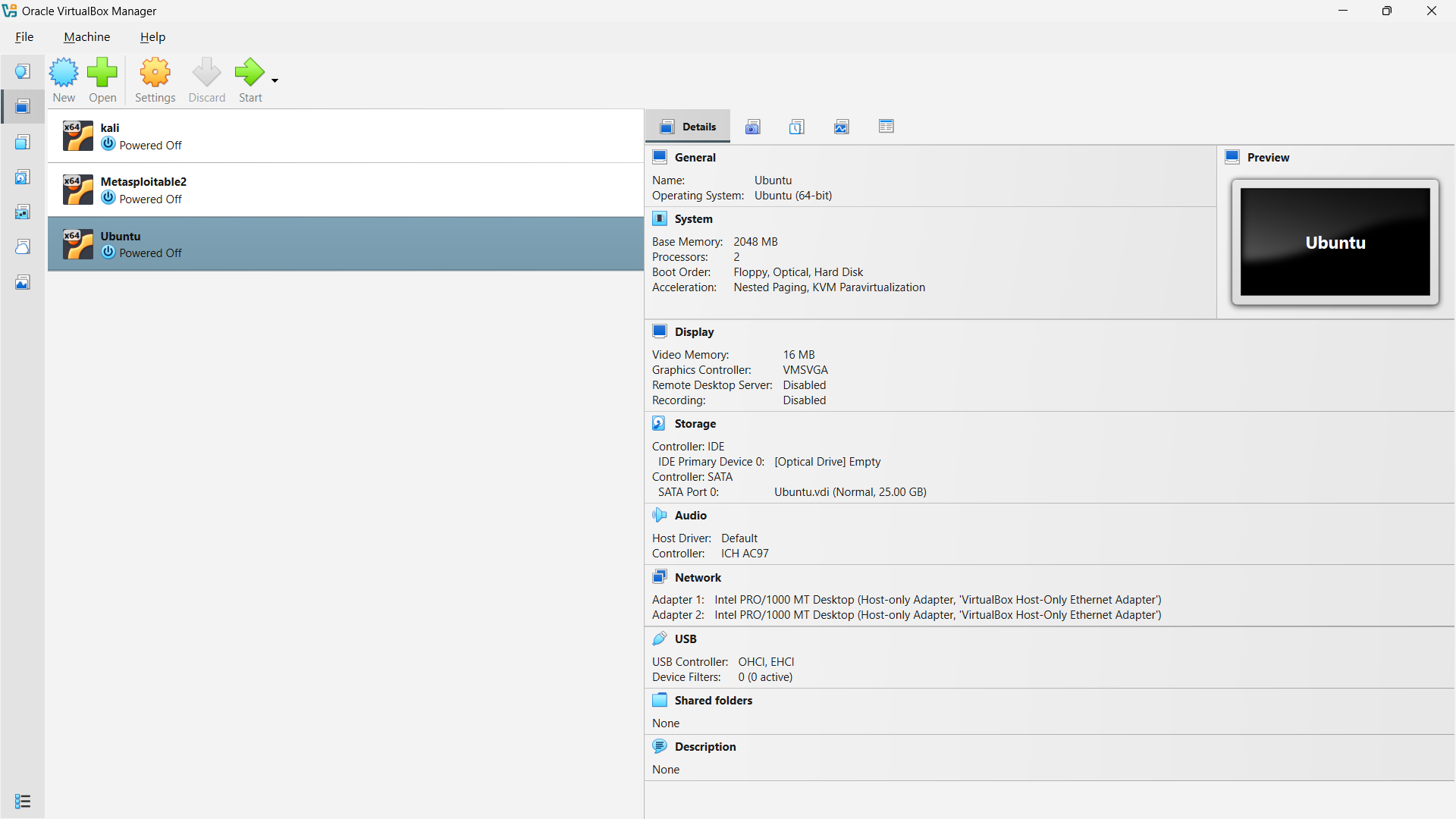
##### Gambar 3. Konfigurasi Jaringan VM Kali-Linux





##### Gambar 4. Spesifikasi VM Metasploitable

1. Pengaturan VM 3 (Ubuntu IDS Server): VM ini unik karena memiliki dua adapter.
   1. Matikan VM, klik Settings > Network.
   2. Adapter 1 (Manajemen):
      1. "Attached to": Host-only Adapter
      2. "Name": (Pilih adapter yang sama dengan VM lain)
      3. "Promiscuous Mode": Deny (Adapter ini tidak perlu me-monitoring)
   3. Adapter 2 (Monitoring):
      1. Klik tab Adapter 2, centang Enable Network Adapter.
      2. "Attached to": Host-only Adapter
      3. "Name": (Pilih adapter yang sama dengan VM lain)
      4. "Promiscuous Mode": Allow All (Ini adalah antarmuka yang akan "didengarkan" oleh Snort).



##### Gambar 5. Spesifikasi VM Ubuntu Server (ids-server-snort-defender)

## 4.2 Konfigurasi IP Address Statis

Setelah jaringannya terpasang, kita harus mengatur alamat IP statis untuk setiap VM agar sesuai dengan rancangan. Nyalakan ketiga VM.

4.2.1 VM 1 (Kali Linux - Attacker)

Langkah 1: Buka File Konfigurasi

1. Buka Terminal (kotak hitam).
2. Ketik perintah berikut untuk mengedit file konfigurasi jaringan:  
   Script :   
   sudo nano /etc/network/interfaces
3. Masukkan password Kali-Linux kita

Langkah 2: Edit Konfigurasi

Kita akan melihat tampilan editor teks.

1. Gunakan tombol panah di keyboard untuk turun ke baris paling bawah.
2. Ketikan / atau salin konfigurasi berikut ini di bagian paling bawah file tersebut. Pastikan script-nya sama persis:

# Konfigurasi IP Statis untuk Proyek (Host-only)

auto eth0

iface eth0 inet static

address 192.168.56.10

netmask 255.255.255.0

Catatan: Jangan ubah baris lain yang sudah ada (seperti auto lo atau iface lo inet loopback). Biarkan mereka apa adanya.

Langkah 3: Simpan dan Keluar

1. Tekan tombol Ctrl + O (huruf O, bukan nol) di keyboard untuk menyimpan.
2. Tekan Enter untuk konfirmasi nama file.
3. Tekan Ctrl + X untuk keluar dari editor nano.

Langkah 4: Restart Network

Agar perubahan ini aktif secara permanen dan bersih, cara terbaik adalah me-restart service jaringan atau me-reboot VM. Kita reboot saja untuk membuktikan bahwa IP-nya benar-benar permanen.

Ketik script di terminal:

sudo reboot

Langkah 5: Verifikasi Akhir

Setelah Kali Linux nyala kembali dan kita login:

1. Buka Terminal.

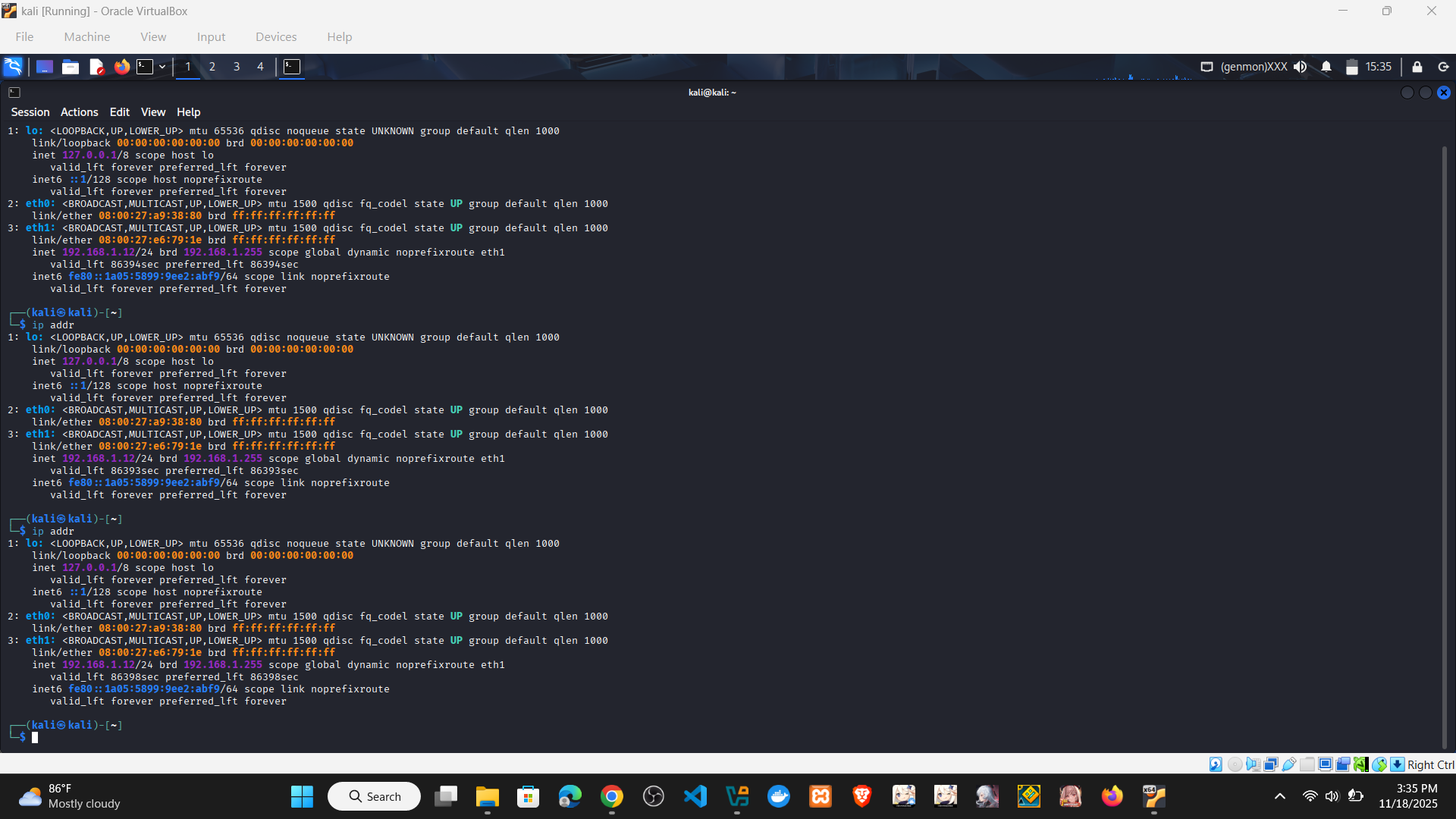
Ketik:

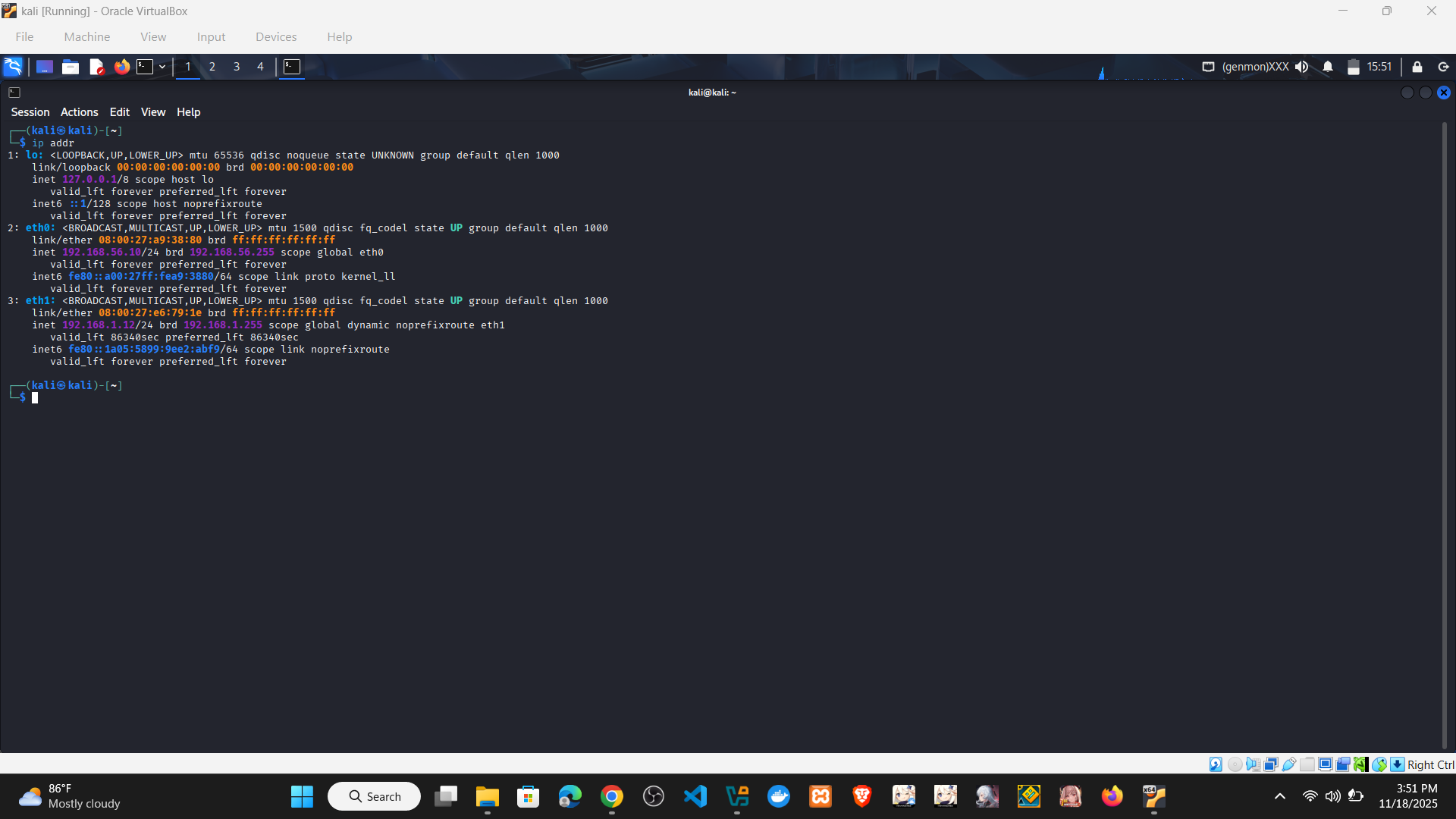
ip addr

1. Cek bagian eth0.

Sekarang, seharusnya kita melihat inet 192.168.56.10/24 terpasang di sana secara otomatis.

##### Gambar 6. IP Statis Kali-Linux Sebelumnya (MAC 08:00:27:a9:38:80)





##### Gambar 7. IP Statis Kali-Linux Setelah (192.168.56.10)

4.2.2 VM 2 (Metasploitable2 - Target)

(Harus via terminal)

1. Login ke Metasploitable2 (default: msfadmin/msfadmin).
2. Buka file konfigurasi *interface*: sudo nano /etc/network/interfaces

Cari bagian auto eth0 dan ubah menjadi seperti ini:

# Konfigurasi IP Statis Adapter Host-only (eth0) Kali-Linux

auto eth0

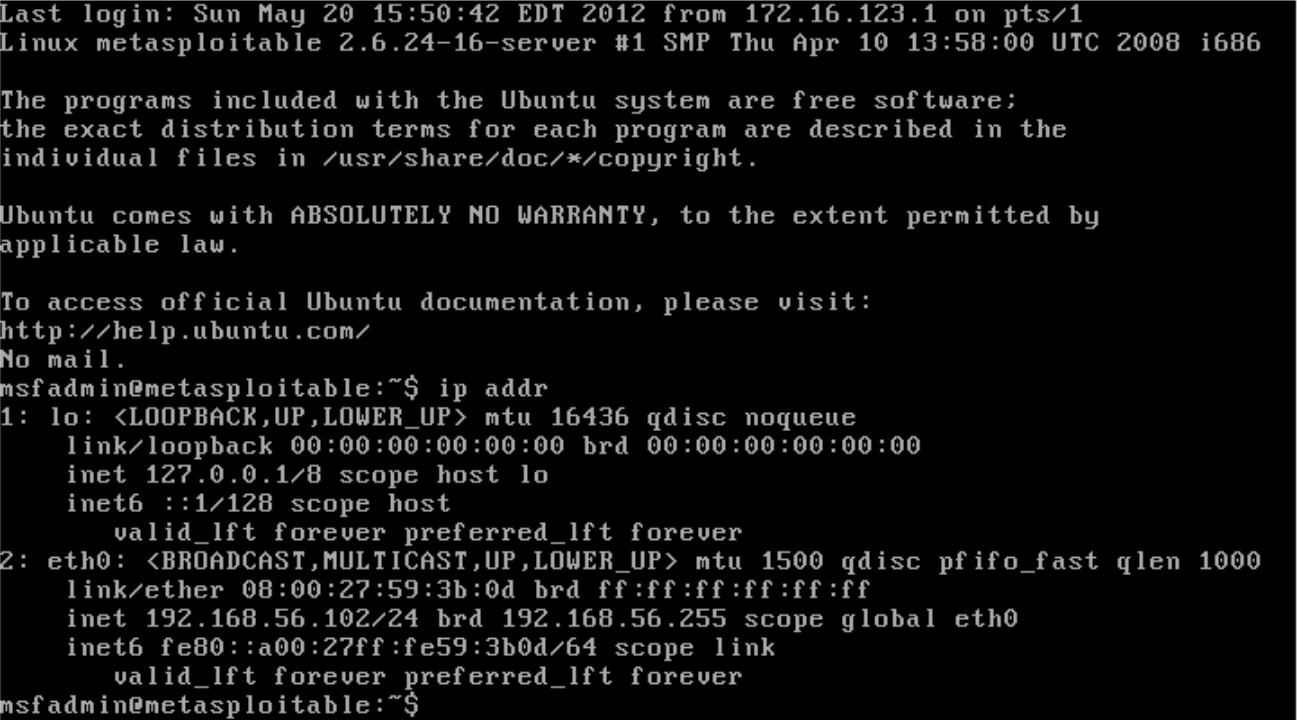
iface eth0 inet static

address 192.168.56.20

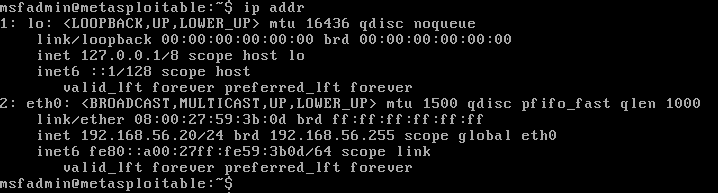
netmask 255.255.255.0

gateway 192.168.56.1

1. Simpan file (Ctrl+O, Enter, Ctrl+X).
2. Restart VM



##### Gambar 8. IP Statis Metasploitable Sebelumnya (MAC 08:00:27:a9:38:80)



##### Gambar 9. IP Statis Metasploitable Setelah (192.168.56.20)

4.2.3 VM 3 (Ubuntu IDS Server)

(Menggunakan Netplan)

1. Login ke server Ubuntu Anda.
2. Edit file konfigurasi Netplan: sudo nano /etc/netplan/00-installer-config.yaml

Ubah isinya agar sesuai dengan rancangan 2 adapter. Adapter 1 (enp0s3) mendapat IP manajemen, Adapter 2 (enp0s8) dibiarkan kosong untuk monitoring.

Format Netplan YAML :

network:

ethernets:

enp0s3:

dhcp4: false

addresses:

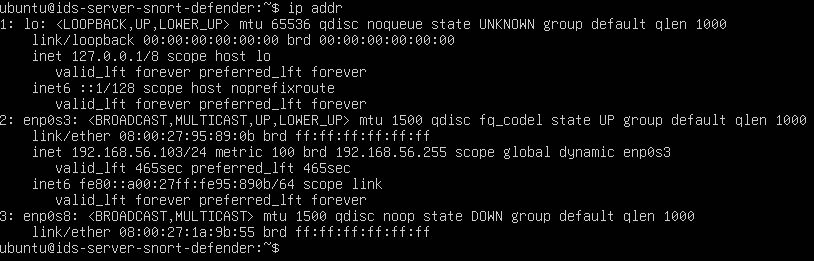
- 192.168.56.30/24

enp0s8:

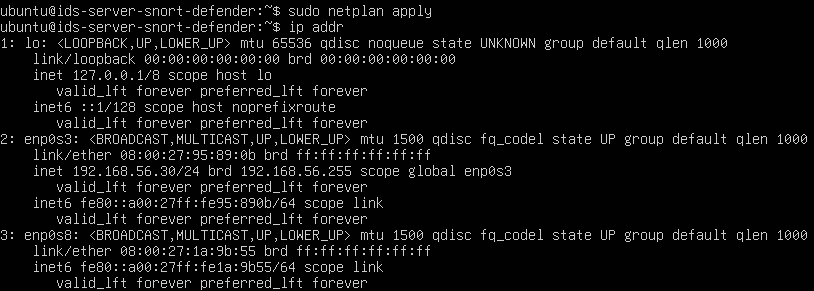
dhcp4: false

version: 2

1. Simpan file.
2. Terapkan konfigurasi: sudo netplan apply



##### Gambar 10. IP Statis Ubuntu Server Sebelumnya (MAC 08:00:27:95:89:0b)

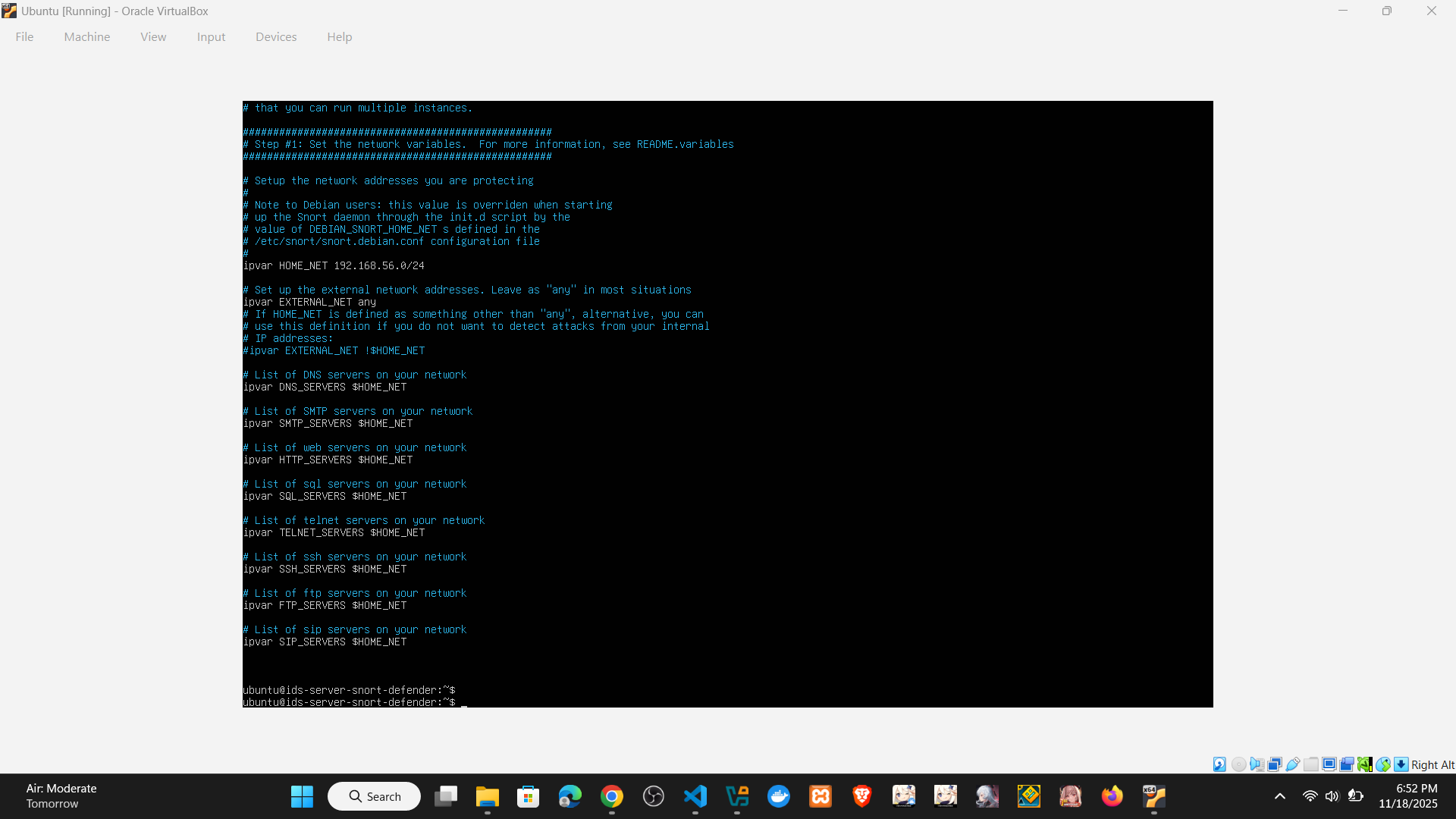


##### Gambar 11. IP Statis Ubuntu Server Setelah (192.168.56.30)

4.3 Konfigurasi Dasar Snort (Basic Rules)

4.3.1 Mengatur Variabel Jaringan (HOME\_NET)

1. Edit file konfigurasi utama Snort: sudo nano /etc/snort/snort.conf
2. Cari baris ipvar HOME\_NET any
3. Ubah any menjadi *range* IP jaringan Anda. Ini memberitahu Snort siapa yang harus dilindungi.
   1. Ubah dari: ipvar HOME\_NET any
   2. Ubah ke: ipvar HOME\_NET 192.168.56.0/24

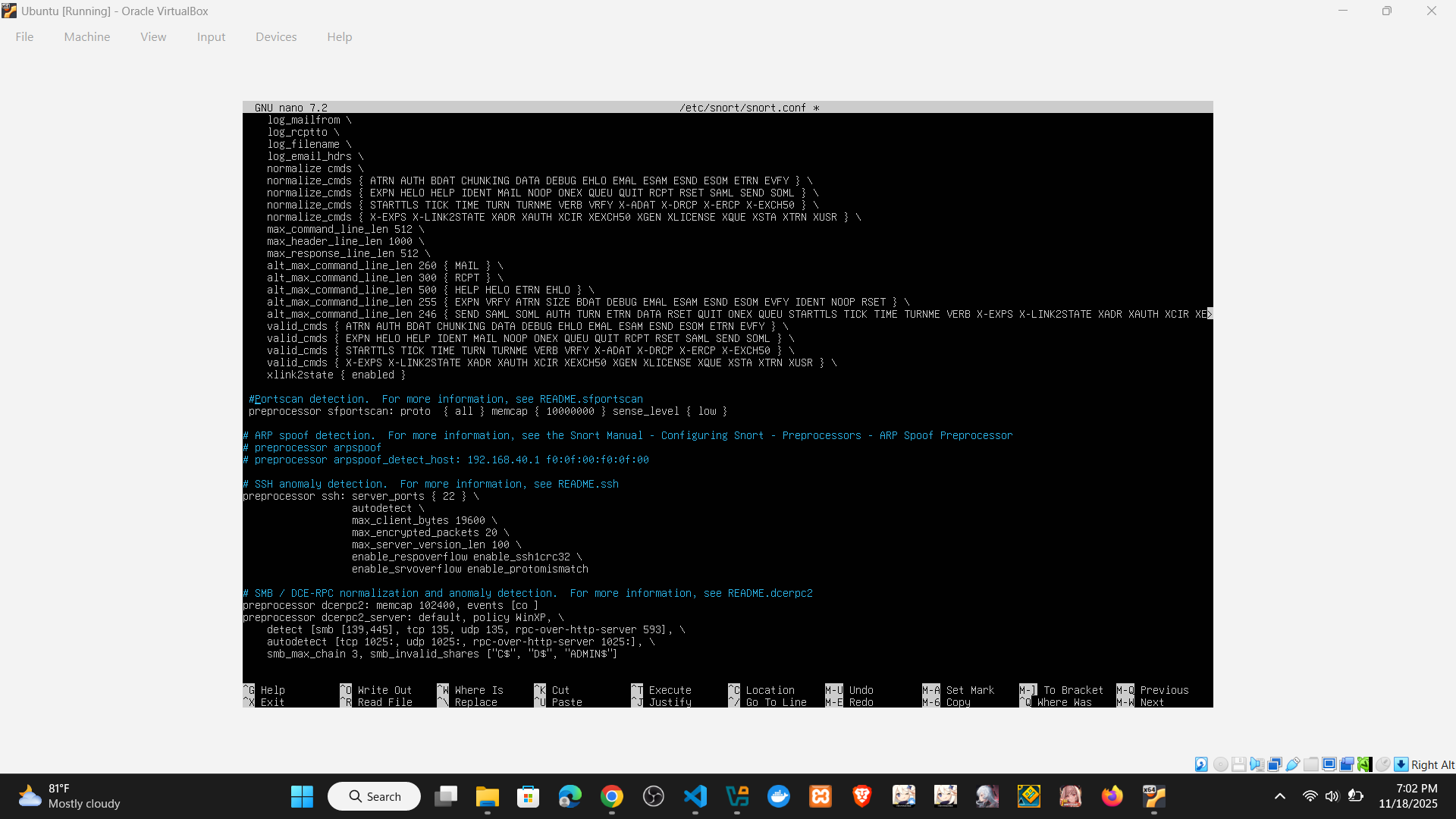


##### Gambar 12. Mengatur Variabel Jaringan (HOME\_NET)

4.3.2 Mengaktifkan Aturan Deteksi Port Scan (sfportscan)

1. Masih di dalam file snort.conf, gulir ke bawah ke bagian *preprocessor*
2. Cari baris yang menyebutkan preprocessor sfportscan.

Pastikan baris tersebut tidak ada tanda pagar (#) di depannya (jika ada, hapus) dan sudah diatur untuk memantau HOME\_NET.  
preprocessor sfportscan: \



##### Gambar 13. Mengaktifkan Aturan Deteksi Port Scan

4.3.3 Membuat Custom Rule (Deteksi Brute Force)

1. Buka file aturan lokal (file ini kosong secara default): sudo nano /etc/snort/rules/local.rules

Tambahkan aturan kustom berikut di baris baru. Aturan ini akan memberi peringatan jika ada lebih dari 5x upaya koneksi ke port 22 (SSH) ke target kita (.20) dalam 60 detik.  
alert tcp any any -> 192.168.56.20 22 (msg:"DETEKSI: Upaya Brute Force SSH"; flow:to\_server; threshold: type threshold, track by\_src, count 5, seconds 60; sid:1000001; rev:1;)

1. Simpan dan tutup file.

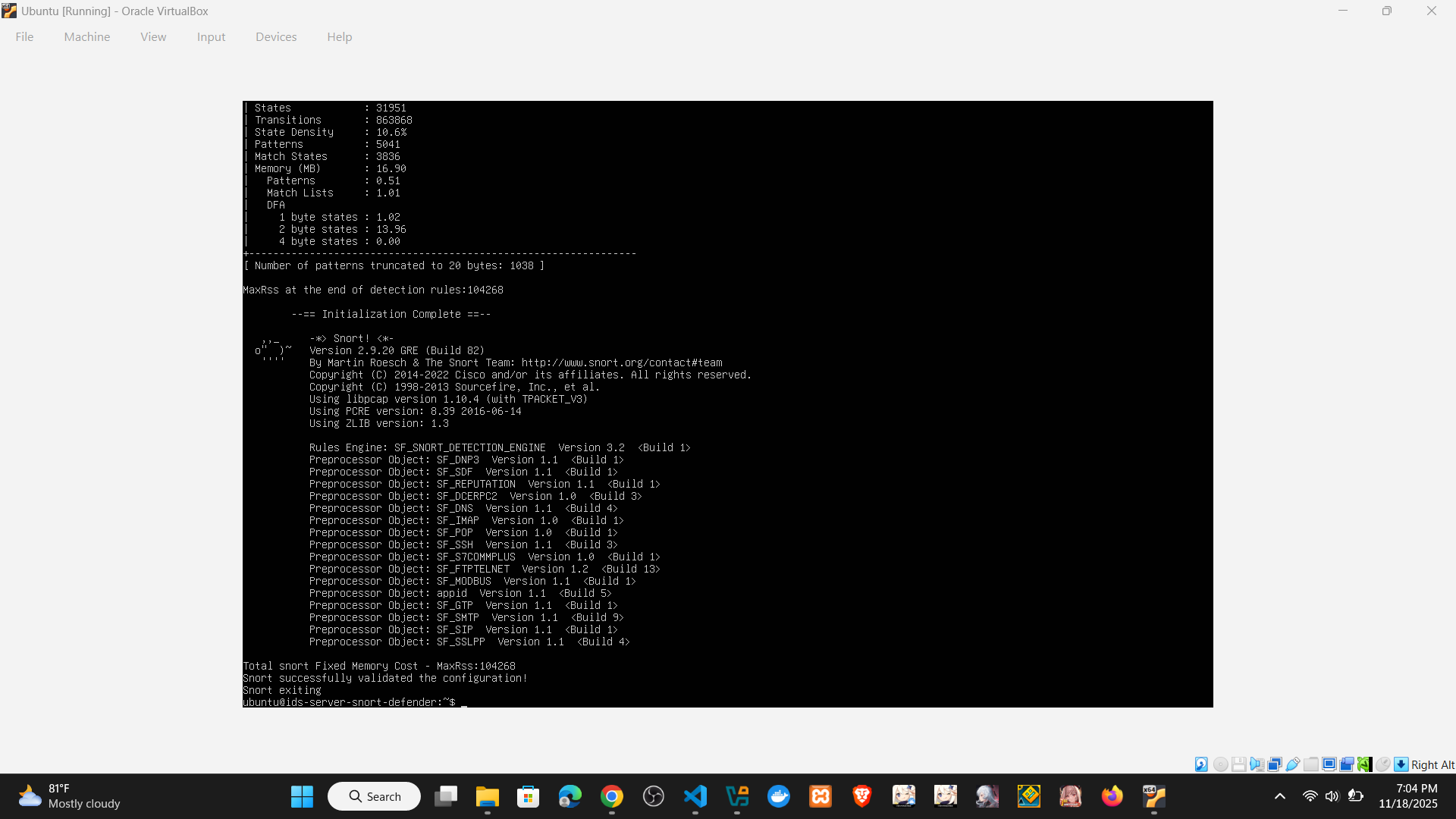
4.3.4 Validasi Konfigurasi

Langkah terakhir adalah menguji apakah semua konfigurasi Snort kita valid sebelum menjalankannya.

Jalankan perintah tes konfigurasi:

sudo snort -T -c /etc/snort/snort.conf

1. Tunggu beberapa saat. Anda harus melihat pesan sukses di akhir output. Snort successfully validated the configuration!



##### Gambar 14. Validasi Konfigurasi

4.4 Membuat Aturan Kustom (Custom Rule)

Sesuai proposal (Skenario 2), kita perlu membuat aturan agar Snort berteriak jika ada yang mencoba *login* SSH berkali-kali (Brute Force) ke Metasploitable2.

Kita tidak menulis aturan ini di snort.conf, melainkan di file khusus bernama local.rules.

1. Buka File Aturan Lokal

Ketik perintah ini di terminal Ubuntu:

sudo nano /etc/snort/rules/local.rules

2. Masukkan Aturan (Rule)

File ini mungkin kosong atau berisi komentar. Arahkan ke baris paling bawah, lalu ketik (atau salin) baris berikut ini. Pastikan diketik dalam SATU baris panjang:

alert tcp any any -> 192.168.56.20 22 (msg:"BAHAYA: Serangan Brute Force SSH Terdeteksi"; flags:S; threshold: type both, track by\_src, count 5, seconds 60; sid:1000001; rev:1;)

Penjelasan Aturan:

1. alert tcp any any -> 192.168.56.20 22  
   Memantau paket TCP dari mana saja yang menuju ke IP Target (.20) port 22 (SSH).
2. msg:"..."   
   Pesan yang akan muncul di log.
3. flags:S   
   Hanya menghitung paket SYN (permintaan koneksi awal).
4. threshold... count 5, seconds 60  
   Jika ada 5 permintaan koneksi dalam 60 detik dari IP yang sama, bunyikan alarm.
5. sid:1000001  
   ID unik untuk aturan ini (Custom rule biasanya dimulai dari 1000000).

3. Simpan dan Keluar

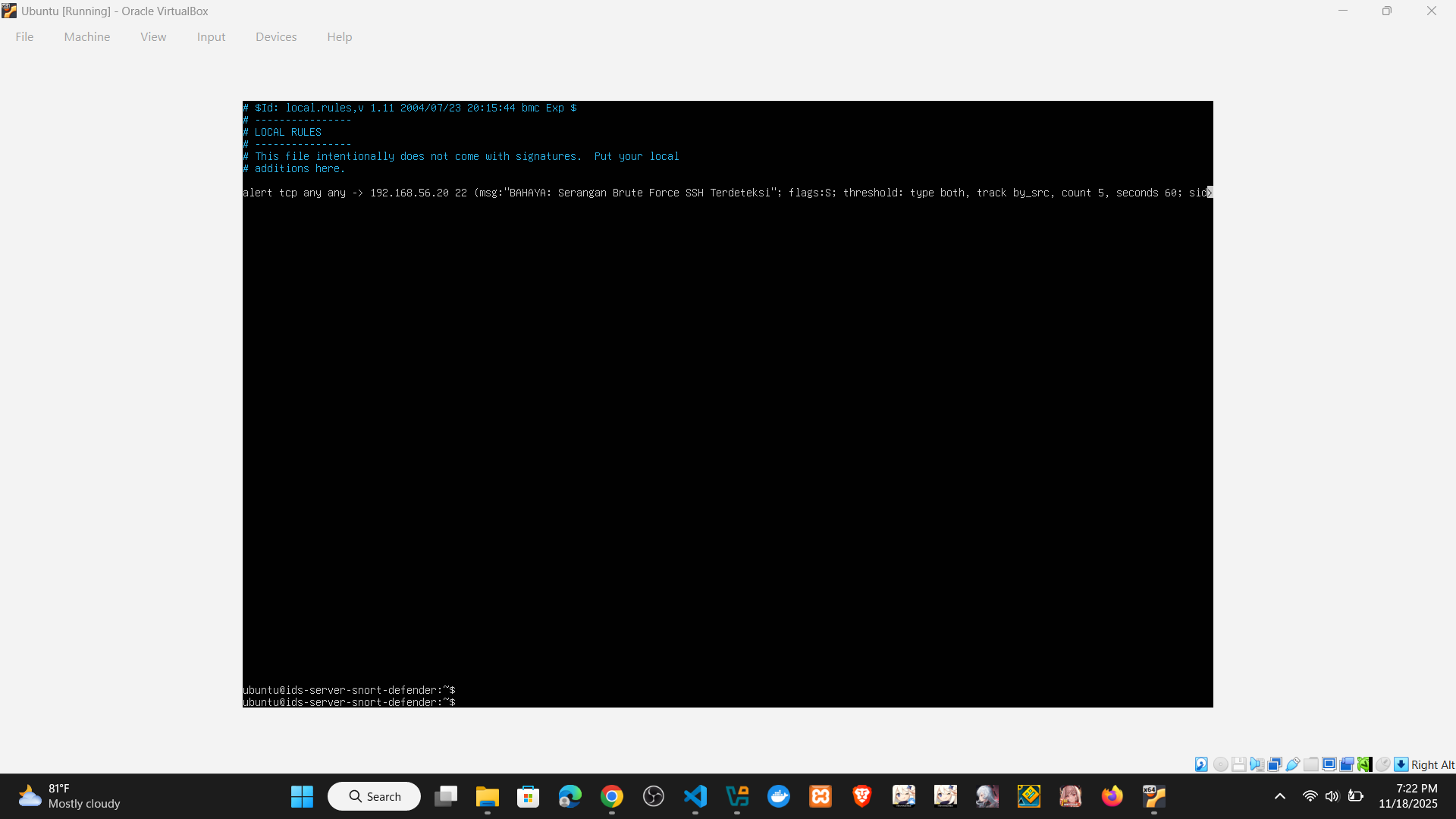
1. Tekan Ctrl + O -> Enter.
2. Tekan Ctrl + X.

4. Validasi Terakhir

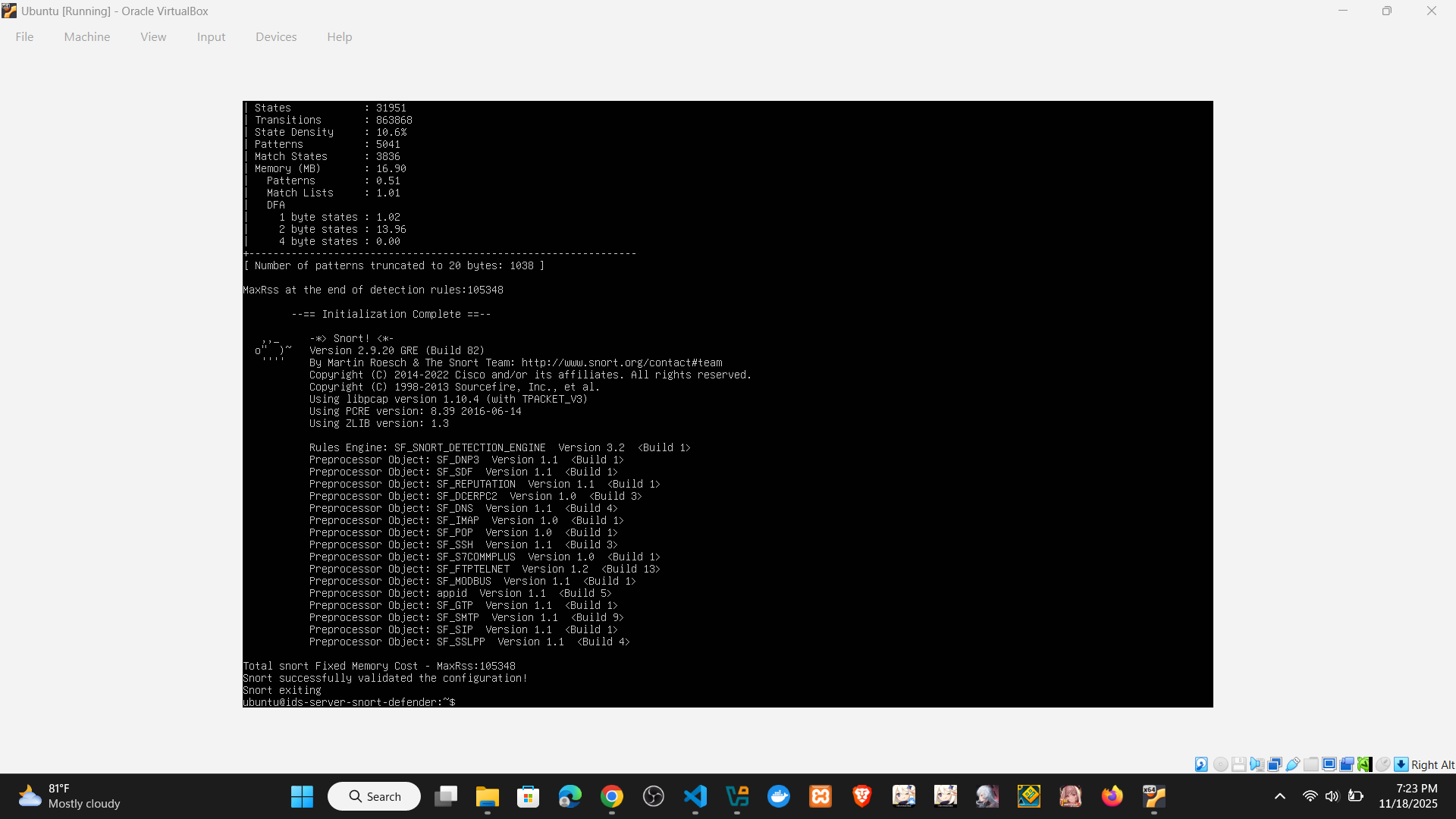
Cek sekali lagi apakah aturan baru tadi menyebabkan error atau tidak dengan

sudo snort -T -c /etc/snort/snort.conf

Jika muncul pesan "Snort successfully validated the configuration!" lagi, maka sudah selesai



##### Gambar 15. Memasukkan Aturan



##### Gambar 16. Snort successfully validated the configuration

# **BAB V**

# **PENGUJIAN & ANALISIS**

## 5.1 Skenario Pengujian 1: Port Scanning

### 5.1.1 Tujuan Pengujian

Pengujian ini bertujuan untuk memverifikasi kemampuan IDS Snort dalam mendeteksi aktivitas pengintaian jaringan (*reconnaissance*), khususnya teknik *Port Scanning*. Skenario ini mensimulasikan upaya penyerang untuk memetakan layanan dan *port* terbuka pada mesin target sebelum melancarkan serangan lebih lanjut.

### 5.1.2 Metodologi Pengujian

Pengujian dilakukan dengan menggunakan tiga mesin virtual yang telah dikonfigurasi dalam jaringan *Host-only* (192.168.56.0/24):

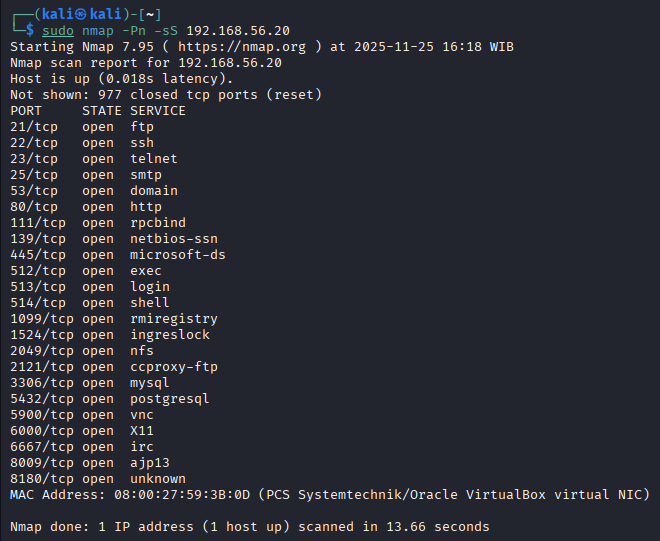
1. Penyerang (Attacker): Kali Linux (192.168.56.10) menggunakan alat Nmap.
2. Target (Victim): Metasploitable2 (192.168.56.20).
3. IDS (Defender): Ubuntu Server (192.168.56.30) dengan Snort yang berjalan dalam mode konsol (*Alert to Console*).

Perintah Serangan: Penyerang menjalankan perintah Nmap dengan opsi *Stealth Scan* (-sS) dan mengabaikan ping (-Pn) untuk memastikan pemindaian tetap berjalan meskipun target tidak merespons ICMP echo.

sudo nmap -Pn -sS 192.168.56.20

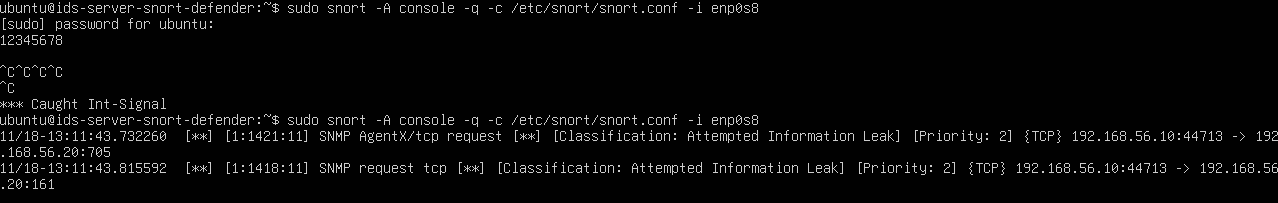
### 5.1.3 Hasil Pengujian

1. Sisi Penyerang (Kali Linux) Nmap berhasil memindai target dan mengidentifikasi sejumlah *port* terbuka (seperti port 21/ftp, 22/ssh, 23/telnet, 80/http).



##### Gambar 17. Output Nmap pada Kali Linux menunjukkan daftar port terbuka pada target.

1. Sisi IDS (Ubuntu Server) Segera setelah Nmap dijalankan, Snort yang sedang memantau antarmuka enp0s8 berhasil mendeteksi lalu lintas mencurigakan tersebut dan menampilkan peringatan (*alert*) secara *real-time* di layar konsol.

**

##### Gambar 18. Log deteksi Snort menunjukkan adanya upaya *scanning* dan permintaan koneksi yang mencurigakan.

### 5.1.4 Analisis Hasil

Berdasarkan hasil pengujian di atas, dapat dianalisis bahwa:

1. Keberhasilan Deteksi: Snort berhasil mendeteksi paket-paket *probe* yang dikirimkan oleh Nmap. Hal ini dibuktikan dengan munculnya log peringatan seperti:
   1. [\*\*] [1:1421:11] SNMP AgentX/tcp request [\*\*]
   2. [\*\*] [1:1418:11] SNMP request tcp [\*\*]
   3. Klasifikasi: Attempted Information Leak
2. Mekanisme Kerja: Peringatan ini muncul karena aturan (*rules*) bawaan Snort dan preprocessor sfportscan yang telah diaktifkan pada tahap konfigurasi (snort.conf) bekerja dengan baik. Snort mengenali pola paket TCP SYN yang dikirim secara massal dan cepat ke berbagai port target sebagai indikasi kuat adanya upaya pengumpulan informasi (*information gathering*).
3. Kesimpulan: Implementasi IDS Snort untuk mendeteksi serangan *Port Scanning* dinyatakan BERHASIL. Sistem mampu memberikan peringatan dini terhadap aktivitas pengintaian jaringan.

## 5.2 Skenario Pengujian 2: Brute Force Login

### 5.2.1 Tujuan Pengujian

Pengujian ini bertujuan untuk memverifikasi kemampuan IDS Snort dalam mendeteksi serangan *Brute Force* (tebak kata sandi secara paksa) pada layanan SSH (Secure Shell). Skenario ini juga bertujuan untuk memvalidasi efektivitas Aturan Kustom (*Custom Rule*) yang telah dibuat sebelumnya, yaitu mendeteksi lonjakan koneksi SSH yang melebihi ambang batas (*threshold*) tertentu dalam waktu singkat.

### 5.2.2 Metodologi Pengujian dan Troubleshooting

1. Penyerang: Kali Linux (192.168.56.10) menggunakan Nmap Scripting Engine (NSE) dengan skrip ssh-brute.
2. Target: Metasploitable2 (192.168.56.20).
3. IDS: Ubuntu Server (192.168.56.30) dengan aturan kustom:

alert tcp any any -> 192.168.56.20 22 (msg:"BAHAYA..."; threshold: count 5, seconds 60)

Pengujian ini dilakukan dalam dua tahap dikarenakan adanya kendala teknis pada percobaan pertama.

#### 5.2.2.1 Percobaan 1: Menggunakan Hydra (Gagal)

Pada tahap awal, skenario serangan direncanakan menggunakan alat THC-Hydra, sebuah alat *login cracker* yang populer.

Perintah yang Dijalankan:

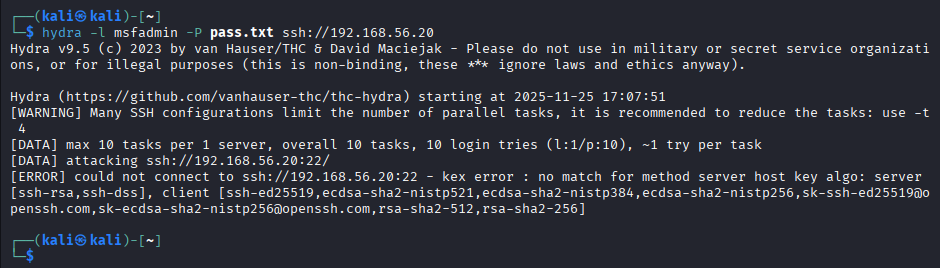
hydra -l msfadmin -P pass.txt ssh://192.168.56.20

Penjelasan Perintah:

1. hydra  
   Memanggil program Hydra.
2. -l msfadmin  
   Menentukan username target tunggal, yaitu msfadmin.
3. -P pass.txt  
   Menentukan daftar password yang akan dicoba dari file pass.txt.
4. ssh://192.168.56.20  
   Protokol (SSH) dan alamat IP target.

Hasil dan Kendala: Saat perintah dijalankan, Hydra gagal melakukan serangan dan langsung berhenti dengan pesan kesalahan:[ERROR] could not connect to ssh://192.168.56.20:22 - kex error

Kesalahan ini terjadi karena adanya inkompatibilitas algoritma enkripsi (Key Exchange Algorithm) antara Kali Linux versi terbaru dengan layanan SSH OpenSSH lama yang berjalan di Metasploitable2. Kali Linux menolak koneksi karena menganggap algoritma enkripsi target sudah usang (legacy) dan tidak aman.



##### Gambar 19. Percobaan serangan menggunakan Hydra gagal akibat kesalahan pertukaran kunci (kex error).

#### 5.2.2.2 Percobaan 2: Menggunakan Nmap Scripting Engine (Berhasil)

Sebagai langkah mitigasi untuk tetap menjalankan skenario pengujian tanpa mengubah konfigurasi keamanan Kali Linux secara drastis, serangan dialihkan menggunakan Nmap Scripting Engine (NSE) dengan skrip ssh-brute.

Perintah yang Dijalankan:

sudo nmap -Pn -p 22 --script ssh-brute --script-args user=msfadmin,passdb=pass.txt 192.168.56.20

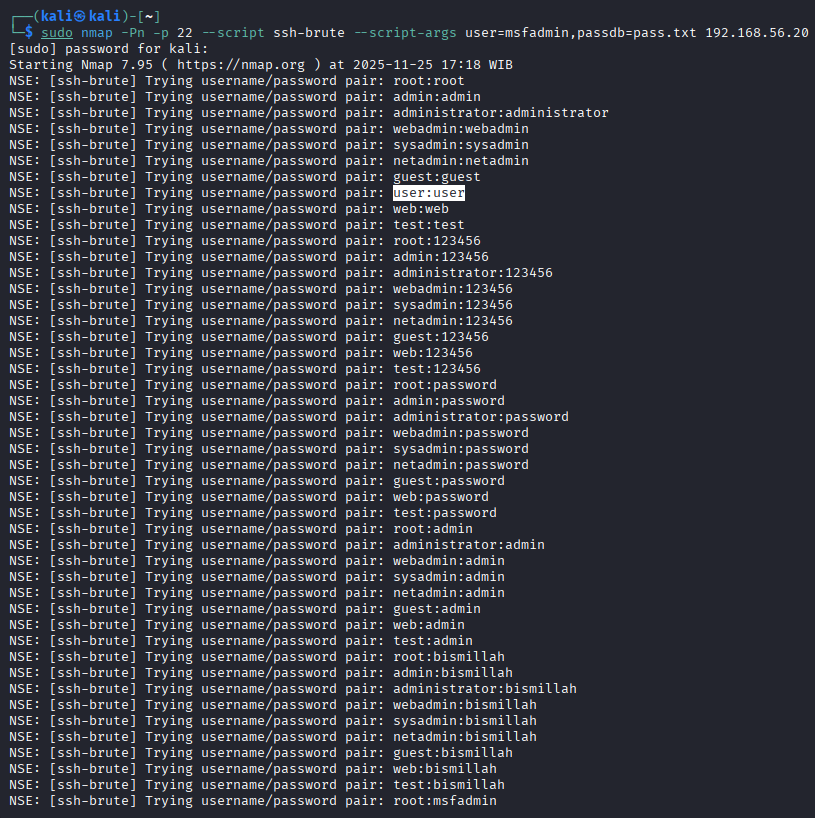
Penjelasan Perintah:

1. sudo nmap   
   Menjalankan Nmap dengan hak akses root.
2. -Pn  
   Menonaktifkan host discovery (ping), memaksa Nmap menganggap target hidup.
3. -p 22  
   Menargetkan spesifik port 22 (SSH).
4. --script ssh-brute  
   Menggunakan skrip bawaan Nmap untuk melakukan brute force SSH.
5. --script-args  
   Memberikan argumen ke skrip:
   1. user=msfadmin  
      Username yang dicoba.
   2. passdb=pass.txt  
      File database password yang digunakan.
6. 192.168.56.20: Alamat IP target.

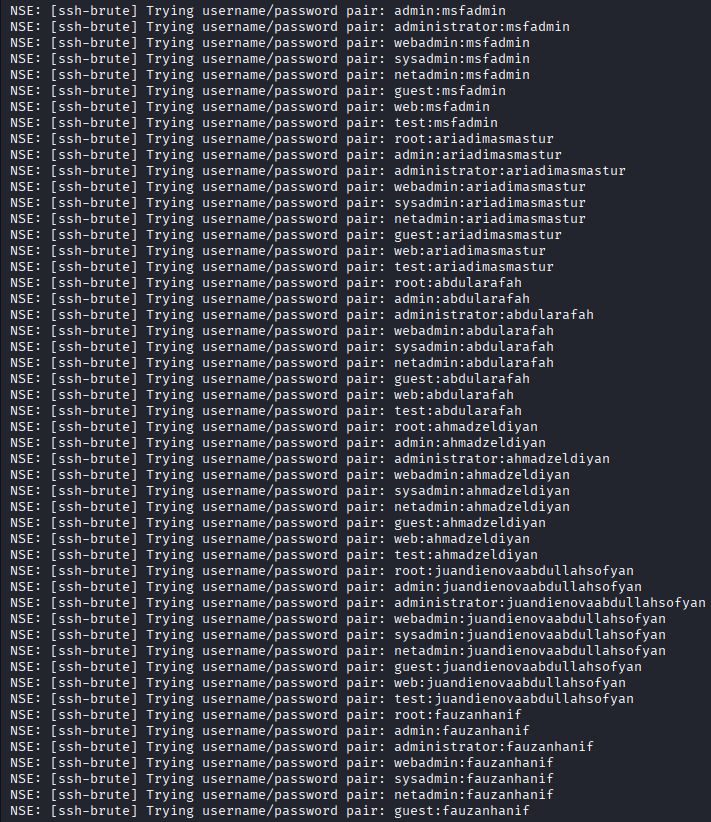
### 5.2.3 Hasil Pengujian

#### 5.2.3.1 Sisi Penyerang (Kali Linux)

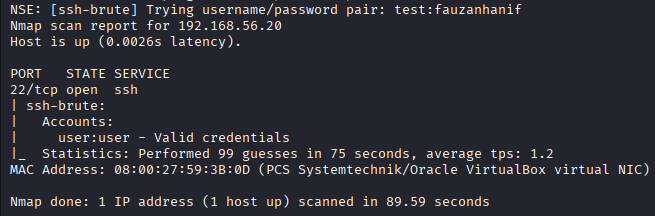
Nmap berhasil melakukan koneksi dan menjalankan serangan dictionary attack menggunakan daftar password yang disediakan. Dari hasil pemindaian, Nmap bahkan berhasil mengidentifikasi kredensial valid pada target untuk akun user dengan password user.



##### Gambar 20. Output Nmap menunjukkan proses percobaan login (brute force) yang masif dan ditemukannya kredensial valid.



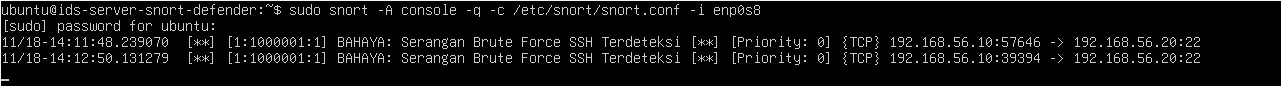
##### Gambar 21. Output Nmap menunjukkan proses percobaan login (brute force) yang masif



##### Gambar 22. Output Nmap menunjukkan proses percobaan login (brute force) yang masif

#### 5.2.3.2 Sisi IDS (Ubuntu Server)

Segera setelah Nmap mengirimkan gelombang percobaan login, Snort berhasil mendeteksi anomali trafik tersebut. Sesuai dengan aturan threshold yang ditetapkan (lebih dari 5 percobaan dalam 60 detik), Snort memicu peringatan (alert).



##### Gambar 23. Log Snort menampilkan peringatan "BAHAYA: Serangan Brute Force SSH Terdeteksi" sebagai respons terhadap serangan.

### 5.2.4 Analisis Hasil

Adaptabilitas Pengujian: Kegagalan pada alat Hydra membuktikan bahwa kompatibilitas protokol enkripsi antar sistem operasi yang berbeda generasi (Kali Linux 2024 vs Metasploitable lama) dapat menjadi hambatan dalam penetration testing. Penggunaan Nmap ssh-brute terbukti menjadi solusi alternatif yang efektif.

Efektivitas Custom Rule: Aturan kustom yang dibuat dengan parameter threshold: count 5, seconds 60 terbukti berfungsi dengan baik. Snort tidak membanjiri layar dengan peringatan untuk setiap paket percobaan, melainkan memberikan satu peringatan tegas saat ambang batas serangan terlampaui.

Kesimpulan: Sistem IDS berhasil mendeteksi pola serangan Brute Force SSH, terlepas dari alat apa yang digunakan oleh penyerang, karena deteksi didasarkan pada perilaku lalu lintas jaringan (traffic behavior), bukan pada signature aplikasi penyerang tertentu.

## 5.3 Implementasi Custom Rule Lanjutan: Deteksi Serangan DoS

Setelah berhasil mengamankan akses layanan (SSH) pada minggu sebelumnya, fokus pengembangan pada minggu ke-5 diarahkan pada deteksi serangan terhadap ketersediaan jaringan (*Availability*), yaitu *Denial of Service* (DoS) berbasis protokol ICMP (*Internet Control Message Protocol*).

Tujuan dari tahapan ini adalah untuk memvalidasi fleksibilitas Snort dalam menangani serangan volumetrik menggunakan mekanisme *thresholding*.

### 5.3.1 Konfigurasi Aturan Deteksi (Custom Rule)

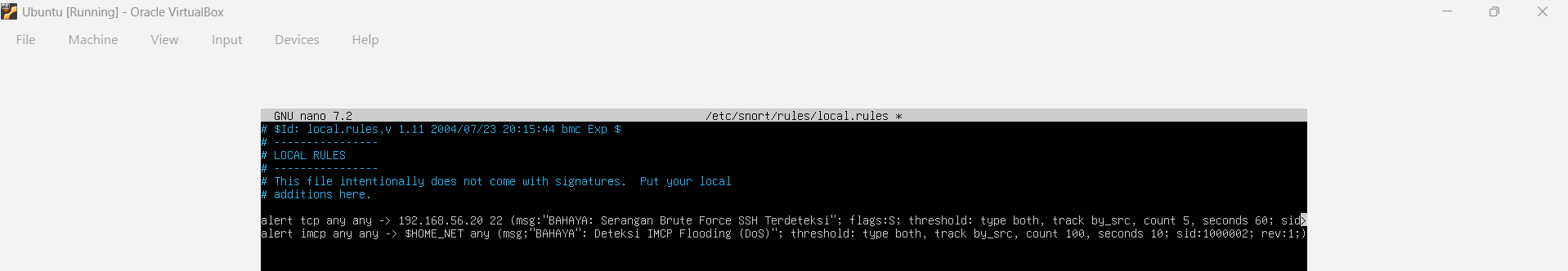
Untuk mendeteksi serangan banjir paket (*packet flooding*), ditambahkan aturan baru pada file konfigurasi lokal /etc/snort/rules/local.rules. Aturan ini dirancang untuk mendeteksi anomali perilaku trafik berdasarkan jumlah paket per detik, bukan hanya berdasarkan tanda tangan (*signature*) paket semata.

Script aturan yang Diterapkan :

alert icmp any any -> $HOME\_NET any (msg:"BAHAYA: Deteksi ICMP Flooding (DoS)"; threshold: type both, track by\_src, count 100, seconds 10; sid:1000002; rev:1;)

Analisis Komponen Aturan:

1. Header (alert icmp...)  
   Memerintahkan Snort untuk memantau seluruh paket ICMP yang masuk ke jaringan lokal ($HOME\_NET).
2. Threshold (count 100, seconds 10)  
   Ini adalah filter anti-noise. Alarm hanya akan dipicu jika satu alamat IP sumber (track by\_src) mengirimkan lebih dari 100 paket dalam jendela waktu 10 detik.
3. SID (1000002)  
   Identifikasi unik untuk membedakan aturan ini dengan aturan Brute Force SSH (1000001) yang dibuat sebelumnya.

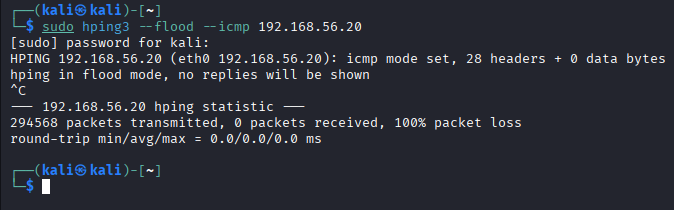


##### Gambar 24. Konfigurasi penambahan aturan deteksi ICMP Flood pada file local.rules

### 5.3.2 Pengujian Variasi 1: Serangan Menggunakan hping3

Pengujian pertama dilakukan menggunakan *packet generator* hping3 dari mesin penyerang (Kali Linux). Alat ini dikonfigurasi dalam mode *flood* untuk membanjiri target secepat mungkin.

1. Perintah : sudo hping3 --flood --icmp 192.168.56.20
2. Tujuan : Menguji respons IDS terhadap alat serangan spesifik yang sering   
    digunakan untuk *stress testing*.



##### Gambar 25. Eksekusi serangan flooding menggunakan hping3 pada Kali Linux

Hasil Deteksi Snort :   
 Berdasarkan pemantauan log, Snort mendeteksi serangan ini dengan klasifikasi "Attempted Information Leak" dan pesan "ICMP PING NMAP".

Analisis :   
 Meskipun serangan ini bersifat *flooding*, Snort memprioritaskan deteksi berdasarkan *signature* (tanda tangan) paket. Header paket yang dihasilkan oleh hping3 memiliki karakteristik yang identik dengan pemindaian *Nmap Ping Sweep*. Oleh karena itu, aturan bawaan Snort (GID:1, SID:469) tertrigger lebih dulu dibandingkan *custom rule* yang kita buat. Ini menunjukkan bahwa Snort mampu mengenali *tools* spesifik yang digunakan penyerang.

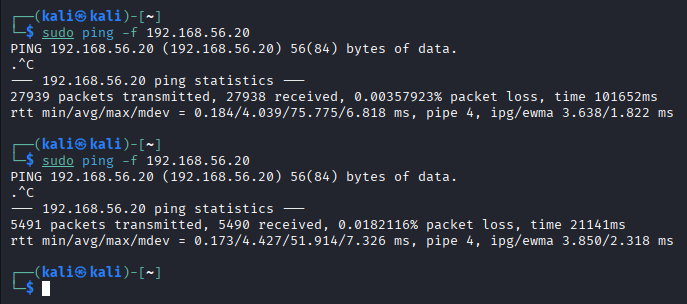


##### Gambar 26. Log Snort mendeteksi pola serangan hping3 sebagai aktivitas 'ICMP PING NMAP'

### 5.3.3 Pengujian Variasi 2: Serangan Menggunakan Ping Flood

Untuk menguji efektivitas *Custom Rule* yang telah dibuat (SID: 1000002), dilakukan pengujian kedua menggunakan utilitas standar ping dengan parameter *flood*.

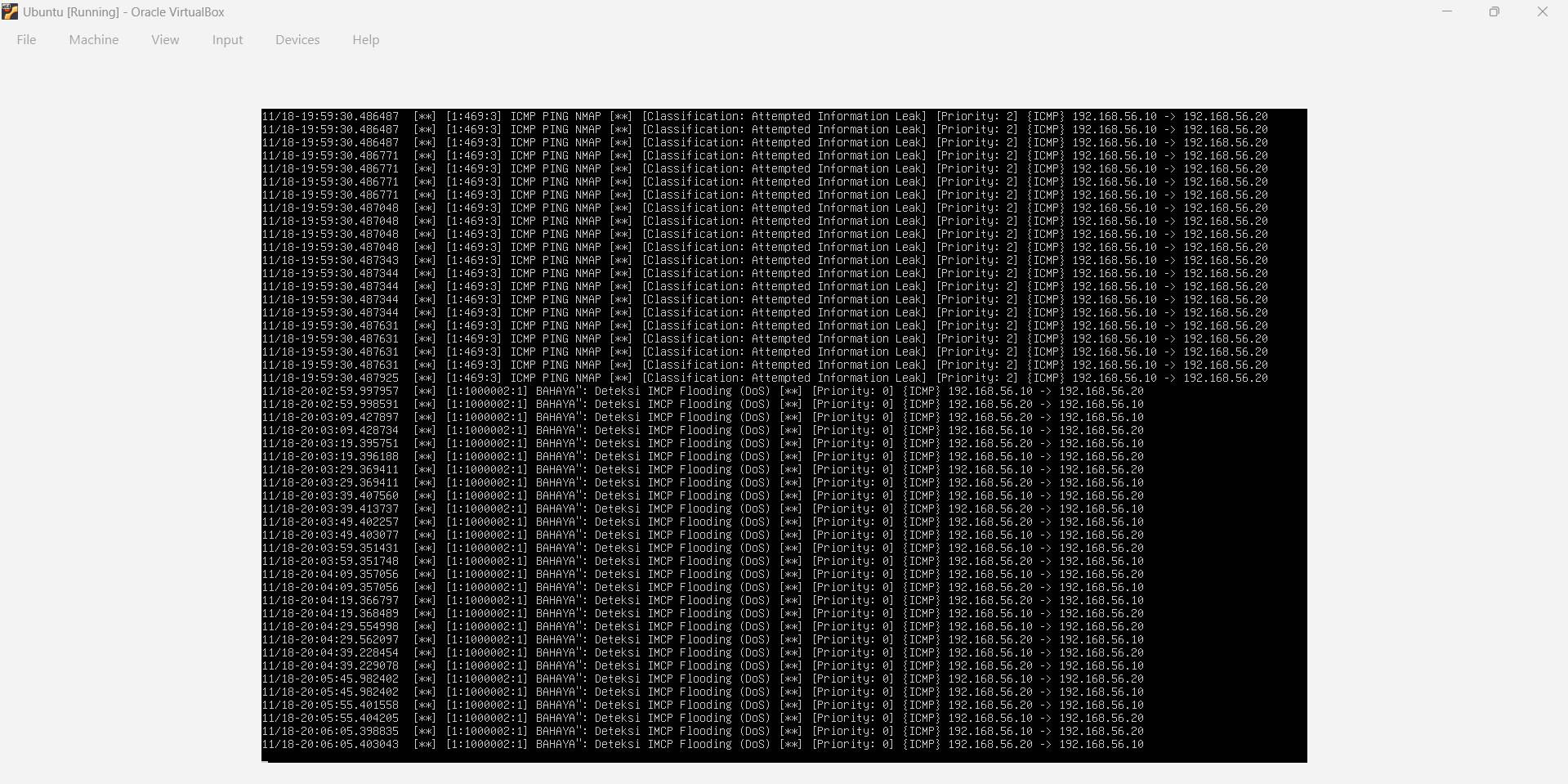
1. Perintah: sudo ping -f 192.168.56.20
2. Karakteristik: Mengirimkan paket *ICMP Echo Request* standar yang valid secara protokol, namun dengan intensitas/frekuensi yang sangat tinggi (ribuan paket per detik).



##### Gambar 27. Eksekusi serangan Ping Flood (-f) yang mengirimkan ribuan paket per detik

Hasil Deteksi Snort :   
 Pada pengujian ini, Snort berhasil memicu aturan kustom dengan pesan "BAHAYA: Deteksi ICMP Flooding (DoS)".

Analisis :   
 Paket yang dikirim oleh perintah ping adalah paket standar yang tidak memiliki *signature* berbahaya (tidak dianggap sebagai Nmap). Oleh karena itu, Snort memeriksa volume trafik. Karena intensitas paket melampaui ambang batas yang ditetapkan dalam aturan (100 paket/10 detik), mekanisme *threshold* bekerja dan memicu peringatan. Hal ini membuktikan bahwa *custom rule* berfungsi efektif untuk mendeteksi serangan volumetrik murni.



##### Gambar 28. Log Snort menampilkan peringatan 'BAHAYA' sesuai konfigurasi threshold

### 5.3.4 Analisis dan Evaluasi Log

Berdasarkan rangkaian pengujian pada Bab 5.3, dapat ditarik analisis mendalam mengenai perilaku IDS Snort:

1. Hierarki Deteksi :   
    Snort bekerja dengan membandingkan lalu lintas jaringan terhadap *database* aturan. Jika sebuah paket cocok dengan aturan spesifik (seperti *signature* Nmap pada hping3), aturan tersebut akan diprioritaskan. Jika tidak ada *signature* spesifik (seperti pada ping -f), Snort akan mengandalkan aturan berbasis perilaku (*behavior-based*) seperti *threshold* volume yang telah kita konfigurasi.
2. Efektivitas Threshold :   
    Penggunaan parameter threshold: count 100, seconds 10 terbukti krusial. Tanpa parameter ini, setiap paket ping (bahkan yang sah untuk cek koneksi) akan memicu peringatan, menyebabkan *False Positive*. Dengan threshold, IDS hanya memberikan peringatan ketika terjadi anomali lonjakan trafik yang mengindikasikan serangan DoS.

# **KESIMPULAN**

Berdasarkan seluruh tahapan perancangan, implementasi, dan pengujian yang telah dilakukan selama pengerjaan proyek akhir ini, dapat ditarik beberapa kesimpulan utama:

1. Keberhasilan Implementasi IDS :   
    Sistem *Intrusion Detection System* (IDS) menggunakan Snort berhasil diimplementasikan pada lingkungan virtual. Snort mampu bekerja secara *real-time* dalam memantau lalu lintas jaringan dan mendeteksi aktivitas mencurigakan yang ditujukan ke server target (Metasploitable2).
2. Efektivitas Deteksi Multi-Vektor :   
    Sistem terbukti mampu mendeteksi berbagai jenis serangan dengan metode yang berbeda:
   1. Port Scanning :   
       Berhasil dideteksi menggunakan preprocessor sfportscan bawaan Snort, yang mengenali pola pemindaian port TCP/UDP.
   2. SSH Brute Force :   
       Berhasil dideteksi menggunakan *Custom Rule* berbasis ambang batas (*threshold*). Aturan ini efektif memberikan peringatan hanya ketika terjadi kegagalan login berulang (5 kali dalam 60 detik), sehingga meminimalisir *false positive* dari kesalahan ketik pengguna yang sah.
   3. DoS ICMP Flood :   
       Berhasil dideteksi melalui dua pendekatan. Serangan menggunakan alat spesifik (hping3) dikenali melalui *signature* paket (header Nmap), sedangkan serangan murni (ping flood) dikenali melalui anomali volume trafik (*traffic anomaly*) menggunakan aturan *threshold* (100 paket/10 detik).
3. Pentingnya Konfigurasi Threshold :   
    Dalam pembuatan aturan kustom (*custom rules*), penggunaan parameter threshold sangat krusial. Pengujian menunjukkan bahwa tanpa pembatasan *rate-limit*, IDS akan membanjiri log dengan peringatan untuk setiap paket tunggal, yang justru menyulitkan analisis. *Thresholding* membantu administrator fokus pada insiden yang benar-benar memiliki intensitas serangan tinggi.

# **SARAN PENGEMBANGAN**

Sistem IDS yang dibangun dalam proyek ini masih bersifat dasar dan berfokus pada deteksi dalam lingkungan simulasi. Untuk pengembangan lebih lanjut agar sistem keamanan menjadi lebih tangguh dan siap diterapkan pada skala produksi, disarankan beberapa peningkatan berikut :

1. Integrasi IPS (*Intrusion Prevention System*)  
    Saat ini Snort hanya memberikan peringatan (*alert*) tanpa menghentikan serangan. Pengembangan selanjutnya dapat mengonfigurasi Snort dalam mode *Inline* atau mengintegrasikannya dengan *Firewall* (seperti iptables atau alat Fail2Ban). Hal ini memungkinkan sistem untuk memblokir alamat IP penyerang secara otomatis begitu serangan terdeteksi.
2. Implementasi Dashboard Monitoring (GUI)  
    Pemantauan log melalui terminal (CLI) kurang efisien untuk jangka panjang. Disarankan untuk membangun antarmuka grafis (GUI) menggunakan alat seperti Splunk, ELK Stack (Elasticsearch, Logstash, Kibana), atau Snorby. Visualisasi ini akan memudahkan administrator dalam melihat tren serangan, grafik statistik, dan manajemen log secara terpusat.
3. Sistem Notifikasi Real-Time  
    Supaya administrator dapat merespons insiden dengan cepat, sistem IDS dapat dihubungkan dengan layanan notifikasi eksternal. Script tambahan dapat dibuat agar Snort mengirimkan peringatan "BAHAYA" langsung ke Email, Telegram Bot, atau Slack segera setelah aturan kritis terpenuhi.
4. Otomatisasi Pembaruan Rules  
    Ancaman siber terus berkembang. Disarankan untuk menginstal alat manajemen aturan seperti PulledPork yang dapat mengunduh dan memperbarui *database* aturan Snort (*Community Rules*) secara otomatis setiap hari, memastikan IDS selalu mengenali jenis serangan terbaru.
5. Migrasi ke Lingkungan Fisik (Dedicated Hardware)  
    Proyek ini disimulasikan dalam lingkungan virtual (*virtual environment*) yang berbagi sumber daya dengan sistem operasi utama (Host OS). Meskipun pengujian berjalan lancar dengan alokasi sumber daya yang tinggi (CPU Multi-core dan RAM >16GB), untuk penerapan di dunia nyata dengan volume lalu lintas yang masif, sangat disarankan untuk menjalankan IDS pada perangkat keras terdedikasi (*dedicated server/appliance*) atau *cloud instance* terpisah untuk menghindari *bottleneck* performa.

# **DAFTAR PUSTAKA**

[1] A. K. Putra dan S. Wijaya, "Analisis Tren Ancaman Siber dan Kebutuhan Sistem

Monitoring Jaringan di Indonesia," *Jurnal Nasional Keamanan Jaringan*, vol. 7, no.

2, hlm. 112-120, 2024.

[2] W. Stallings, *Network Security Essentials: Applications and Standards*, 7th ed. London:

Pearson Education, 2021.

[3] Cisco Systems, Inc., "Snort: The Open Source IDS/IPS," 2025. [Online].

[4] M. T. Santoso dan S. V. Hartono, "Deteksi Serangan Brute-Force dan Port Scanning

Menggunakan Aturan Kustom IDS," *Jurnal Proteksi: Jurnal Ilmiah Teknik*

*Informatika*, vol. 11, no. 1, hlm. 34-42, 2023.

[5] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed*

*Systems*, 3rd ed. Indianapolis: Wiley, 2021.

[6] Nmap Project, "Nmap Network Scanning: The Official Project Guide (Online Edition),"   
 *Insecure.org*, 2025. [Online].

[7] B. E. H. Putra dan A. Wijaya, "Implementasi Deteksi Serangan Brute Force SSH

Menggunakan Snort pada Jaringan Skala Kecil," *CYBERNET: Jurnal Pendidikan*

*Teknologi Informasi*, vol. 5, no. 1, hlm. 22-29, 2024.

[8] R. A. Ramadhan dan Y. A. P. S. Utama, "Analisis Pola Serangan Distributed Denial of   
 Service (DDoS) Menggunakan Metode Log Analysis pada Snort IDS," *Jurnal   
 Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 6, no. 4, hlm.   
 1543-1550, 2022.

[9] S. Z. Sunaristi, "Implementasi Sistem Deteksi Intrusi Menggunakan Snort Terhadap   
 Serangan ICMP Flooding pada Infrastruktur Cloud," *Jurnal Informatika dan   
 Keamanan Jaringan*, vol. 4, no. 1, hlm. 25-31, 2023.

[10] D. P. S. Wibowo, "Optimasi Kinerja Snort IDS Menggunakan Rule Tuning dan   
 Thresholding pada Jaringan Cloud," *Jurnal Nasional Teknik Elektro dan Teknologi   
 Informasi (JNTETI)*, vol. 11, no. 3, hlm. 210-218, 2023.

[11] A. S. Pratama dan R. Hidayat, "Uji Penetrasi Jaringan Menggunakan Teknik Packet   
 Flooding dengan Hping3 dan Analisis Forensik," *Jurnal RESTI (Rekayasa Sistem dan   
 Teknologi Informasi)*, vol. 8, no. 2, hlm. 201-209, 2024.

[12] Oracle Corporation, "Oracle VM VirtualBox User Manual: Version 7.1," *VirtualBox.org*,   
 2025. [Online].

[13] M. I. F. Rozi dan F. A. Yulianto, "Perbandingan Akurasi Deteksi Serangan DoS Antara   
 Snort dan Suricata pada Lingkungan Virtual," *e-Proceeding of Engineering*, vol. 11,   
 no. 2, hlm. 1123-1130, 2024.