

**ANALISIS KEAMANAN JARINGAN MENGGUNAKAN METODE
VULNERABILITY ASSESSMENT DAN *PENETRATION TESTING*
DENGAN MENGIMPLEMENTASIKAN IDS/IPS
MENGGUNAKAN SURICATA**

(SKRIPSI)

**Oleh
ABYU HAFIZH SEPTRianto
NPM 1915031064**



**JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
2025**

**ANALISIS KEAMANAN JARINGAN MENGGUNAKAN METODE
VULNERABILITY ASSESSMENT DAN *PENETRATION TESTING*
DENGAN MENGIMPLEMENTASIKAN IDS/IPS
MENGGUNAKAN SURICATA**

**Oleh
ABYYU HAFIZH SEPTRianto**

Skripsi

**Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA TEKNIK**

Pada

**Jurusan Teknik Elektro
Fakultas Teknik Universitas Lampung**



**FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG**

2025

ABSTRAK

ANALISIS KEAMANAN JARINGAN MENGGUNAKAN METODE VULNERABILITY ASSESSMENT DAN PENETRATION TESTING DENGAN MENGIMPLEMENTASIKAN IDS/IPS MENGGUNAKAN SURICATA

Oleh

ABYU HAFIZH SEPTRianto

Dengan pesatnya perkembangan teknologi internet, lanskap ancaman digital telah meningkat berlipat ganda, membuat arsitektur aplikasi web dan basis data semakin rentan terhadap eksploitasi. Penelitian ini mengusulkan kerangka kerja keamanan jaringan yang diperkuat dengan Penilaian Kerentanan dan Pengujian Penetrasi (VAPT) dan implementasi Suricata sebagai solusi IDS/IPS. Melampaui sekadar identifikasi ancaman, penelitian ini melakukan serangkaian simulasi serangan dunia nyata seperti SQL Injection, Brute Force, dan Reverse Shell untuk menilai efektivitas IDS/IPS Suricata. Dengan menggunakan OpenVAS untuk pemindaian kerentanan awal dan Suricata untuk IDS/IPS, lingkungan simulasi ancaman dengan fidelitas tinggi telah dibuat. Analisis forensik dari kerangka kerja awal menunjukkan kerentanan kritis yang memungkinkan serangan tanpa mitigasi untuk meningkat ke hak akses root. Setelah aturan yang disesuaikan diimplementasikan, kerangka kerja keamanan diubah secara signifikan. Kerangka kerja tersebut terbukti tangguh secara operasional dengan Akurasi 90% dan Tingkat Positif Palsu 0%. Meskipun Tingkat Negatif Palsu sebesar 16,67% menunjukkan perlunya kalibrasi ulang lebih lanjut dari kerangka kerja dengan serangan brute force yang tidak jelas, penelitian ini dengan jelas menunjukkan bahwa implementasi Suricata dengan VAPT secara signifikan memperkuat keamanan server. Dengan implementasi peringatan berbasis Telegram, kerangka kerja ini dapat digunakan sebagai solusi presisi tinggi dan kecepatan tinggi untuk mitigasi ancaman modern.

Kata kunci: *Network Security, Vulnerability Assessment, Analisis Jaringan, Penetration Testing, IDS/IPS, Suricata, OpenVAS, SQL Injection, Brute Force, Reverse Shell*

ABSTRACT

NETWORK SECURITY ANALYSIS USING VULNERABILITY ASSESSMENT AND PENETRATION TESTING METHODS WITH IDS/IPS IMPLEMENTATION BASED ON SURICATA

By

ABYYU HAFIZH SEPTRIANTO

The digital threat environment has increased manifold with the rapid growth of the internet, making the database and web application architectures even more vulnerable to exploitation. The use of Suricata IDS/Ips and Vulnerability Assessment and Penetration Testing (VAPT) provides a stronger network security framework, according to the study conducted in the current research study. The study not only assesses the threat environment but also tests the efficiency of the Suricata IDS/Ips by simulating several types of attacks, including SQL Injection, Brute Force, and Reverse Shell attacks. By employing OpenVAS for initial vulnerability scanning and Suricata for IDS/IPS, a high-fidelity threat simulation environment was created. Forensic analysis of the initial framework indicated a critical vulnerability that allowed unmitigated attacks to escalate to root privileges. Once the customized rules were implemented, the security framework was significantly altered. The framework was found to be operationally resilient with an Accuracy of 90% and a 0% False Positive Rate. While a False Negative Rate of 16.67% indicates the need for further recalibration of the framework with obscure brute force attacks, the research clearly indicates that the implementation of Suricata with VAPT significantly strengthens the security of the server. With the implementation of Telegram-based alerts, the framework can be employed as a high-precision, high-speed solution for the mitigation of modern threats.

Keyword: Network Security, Vulnerability Assessment, Network Analysis, Penetration Testing, IDS/IPS, Suricata, OpenVAS, SQL Injection, Brute Force, Reverse Shell

Judul Skripsi

: **ANALISIS KEAMANAN JARINGAN
MENGUNAKAN METODE *VULNERABILITY
ASSESSMENT* DAN *PENETRATION TESTING*
DENGAN MENGIMPLEMENTASIKAN
IDS/IPS MENGGUNAKAN SURICATA**

Nama Mahasiswa

: **Abyy Hafizh Septianto**

Nomor Pokok Mahasiswa

: **1915031064**

Program Studi

: **Teknik Elektro**

Jurusan

: **Teknik Elektro**

Fakultas

: **Teknik**

MENYETUJUI

1. **Komisi Pembimbing**

Pembimbing Utama

Pembimbing Pendamping


Ir. Yetti Yuniati, S.T., M.T.

NIP 198001132009122002


Aryanto, S.T., M.T.

NIP 199006212019031011

2. **Mengetahui**

**Ketua Jurusan
Teknik Elektro**



Herlinawati, S.T., M.T.

NIP. 197103141999032001

**Ketua Program Studi
Teknik Elektro**



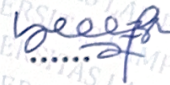
Sumadi, S.T., M.T.

NIP. 197311042000031001

MENGESAHKAN

1. Tim Penguji

Pembimbing 1 : **Ir. Yetti Yuniati, S.T., M.T.**


.....

Pembimbing 2 : **Aryanto, S.T., M.T.**


.....

Penguji : **Misfa Susanto, S.T., M.Sc., Ph.D.**


.....



2. Dekan Fakultas Teknik Universitas Lampung



Dr. Eng. Ir Helmy Fitriawan, ST., M.Sc. }

NIP 197509282001121002

Tanggal Lulus Ujian Skripsi: **28 November 2025**

SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini, menyatakan bahwa skripsi saya dengan judul “Analisis Keamanan Jaringan Menggunakan Metode *Vulnerability Assessment* dan *Penetration Testing* dengan Mengimplementasikan IDS/IPS Menggunakan Suricata” dibuat oleh saya sendiri. Semua hasil yang tertuang pada skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila pernyataan saya tidak benar dan terbukti bahwa skripsi ini merupakan salinan atau dibuat oleh orang lain, maka saya bersedia menerima sanksi sesuai dengan ketentuan hukum atau akademik yang berlaku.

Bandar Lampung, 28 November 2025

Penulis,



Abyyu Hafizh Septrianto

1915031064

RIWAYAT HIDUP



Penulis dilahirkan di Jakarta, pada tanggal 4 September 2001. Penulis merupakan anak pertama dari pasangan Bapak Sugeng Riyanto dan Ibu Endang Trimukti. Penulis menyelesaikan pendidikannya di TK Fajar Pertiwi pada tahun 2007, SD Negeri 05 Cengkareng Timur pada tahun 2013, SMP Negeri 248 Jakarta Barat pada tahun 2016, dan SMK Telkom Sandhy Putra Jakarta pada tahun 2019. Pada tahun 2019 penulis terdaftar sebagai mahasiswa Program Studi Teknik Elektro, Jurusan Teknik Elektro, Fakultas Teknik Universitas Lampung melalui jalur SBMPTN. Selama menjadi mahasiswa, penulis memfokuskan untuk mendalami bidang konsentrasi Telekomunikasi dan Teknologi Informasi. Selama menjadi mahasiswa penulis aktif mengikuti organisasi Himpunan Mahasiswa Teknik Elektro Universitas Lampung selama dua periode kepengurusan, sebagai anggota Departemen Kaderisasi dan Pengembangan Organisasi. Penulis juga mengikuti kegiatan Program Penumbuhan dan Pengembangan Wirausaha Desa (Program Wira Desa) pada tahun 2021. Penulis melaksanakan Kerja Praktik di PT Telkom Indonesia STO Tanjung Karang pada tahun 2022. Penulis juga mengikuti Program Magang Studi Independen Bersertifikat mengambil kelas *Network Security Operation Center (Blue Cyber Team)* dan mendapatkan sertifikat *Certified Network Defender (CND)*. Selain itu, penulis juga mengikuti kegiatan Bimbingan Teknis Tim Tanggap Insiden Siber Gelombang V yang diselenggarakan oleh Pusat Pengembangan Sumber Daya Manusia Badan Siber dan Sandi Negara pada Agustus 2024.

MOTTO

Beri aku sepuluh pemuda, niscaya akan kuguncangkan dunia.

(Ir. Soekarno)

Tidak ada yang lebih indah dari sebuah proses, tidak ada yang lebih mulia dari sebuah tanggung jawab.

(Abyyu Hafizh Septrianto)

Security is not a product, but a process.

(Bruce Schneier)

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Syujud syukur kupersembahkan kepada Allah سُبْحَانَهُ وَتَعَالَى, Tuhan Yang Maha Esa dan Maha Besar. Berkat Limpahan rahmat-Mu saya bisa menjadi pribadi yang bertaqwa, beriman, dan berilmu. Semoga dengan keberhasilan yang telah dicapai ini saya dapat menuju masa depan yang lebih baik dan dapat menggapai cita-cita serta selalu berada di jalan-Mu

KUPERSEMBAHKAN SKRIPSI INI TERUNTUK:

“Ayahanda Sugeng Riyanto dan Ibunda Endang Trimukti atas dukungan dan kasih sayang yang diberikan mulai dari saya ada di dunia ini sampai saya sudah besar seperti sekarang ini. Terima kasih kepada Ayah dan Ibu atas doa yang tak henti-hentinya dipanjatkan serta pengorbanan yang tak terhitung nilainya. Semoga dengan ilmu dan cita-cita yang saya dapatkan kelak akan menjadi amal jariyah bagi Ayah dan Ibu”

“Terimakasih untuk Adik tercinta Haidar Afif Kamil Ramadhan yang selalu mendoakan dan memberikan dukungan dan semangat untuk menyelesaikan perkuliahan ini”

“Terima Kasih untuk Annisa Tri Hapsari yang senantiasa memberikan perhatian, pengertian, dan dukungan tiada henti, serta telah membantu dalam menyelesaikan penulisan skripsi ini”

“Terima Kasih kepada teman-teman Jurusan Teknik Elektro angkatan 2019, *Basecamp*’19, Saudara Rahmad Sidik yang telah membantu dan mendorong saya untuk menyelesaikan penulisan skripsi ini. Semoga kelak apa yang kita impikan akan menjadi kenyataan”

SANWACANA

Segala puji syukur kehadiran Allah SWT, Tuhan semesta alam yang Maha Pengasih lagi Maha Penyayang, atas limpahan rahmat, taufik serta hidayah-Nya sehingga penulis dapat menyelesaikan penulisan skripsi dengan judul **“Analisis Keamanan Jaringan Menggunakan Metode *Vulnerability Assessment* dan *Penetration Testing* dengan Mengimplementasikan IDS/IPS Menggunakan Suricata”**. Sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro Jurusan Teknik Elektro Fakultas Teknik Universitas Lampung. Penulisan skripsi ini tidak lepas dari bantuan, bimbingan, motivasi, dan saran yang diberikan dari semua pihak. Oleh karena itu, pada kesempatan ini penulis ingin mengucapkan terima kasih kepada :

1. Kedua orang tua yang sangat penulis sayangi, Bapak Sugeng Riyanto dan Almarhumah Ibu Endang Trimukti yang senantiasa mengajarkan kebaikan, memberikan perhatian, semangat, materi, kasih sayang, do'a terbaik, dukungan, serta kepercayaan atas setiap keputusan yang diambil oleh penulis hingga detik ini dan juga adik tercinta Haidar Afif Kamil Ramadhan Ramadhan yang selalu mendukung langkah penulis.
2. Ibu Herlinawati, S.T., M.T selaku Ketua Jurusan Teknik Elektro Universitas Lampung.
3. Bapak Sumadi, S.T., M.T selaku Ketua Program Studi Teknik Elektro Universitas Lampung.
4. Ibu Yetti Yuniati, S.T., M.T selaku Dosen Pembimbing Utama yang telah memberikan bimbingan dan dukungan dalam pengerjaan skripsi ini.
5. Bapak Aryanto, S.T., M.T selaku Dosen Pembimbing Pendamping yang telah memberikan bimbingan dan dukungan dalam pengerjaan skripsi ini.

6. Bapak Misfa Susanto, S.T., M.sc., Ph.D selaku Dosen Penguji yang telah memberikan masukan dan saran yang membangun dalam penyelesaian penulisan skripsi ini.
7. Kepada Annisa Tri Hapsari yang senantiasa menampung keluh kesah penulis baik dikala sedih ataupun bahagia selama pengerjaan skripsi ini dilaksanakan, dan juga telah membantu dalam mengerjakan skripsi ini.
8. Kepada sanak saudara Teknik Elektro angkatan 2019, *Basecamp*'19, saudara Rahmad Sidik, yang telah memberikan motivasi kepada penulis untuk menyelesaikan skripsi ini.
9. Seluruh pihak yang telah membantu dan memberikan dukungannya yang tak bisa disebutkan satu persatu.

Penulis berharap semoga skripsi ini dapat menjadi referensi bagi pengembangan keilmuan di bidang Teknik Elektro. Oleh karena itu, semoga penelitian ini bermanfaat dan berguna bagi pembaca dan juga bagi penulis sendiri.

Bandar Lampung, 28 November 2025

Penulis,

Abyyu Hafizh Septrianto

NPM. 1915031064

DAFTAR ISI

	Halaman
DAFTAR ISI	i
DAFTAR GAMBAR	v
DAFTAR TABEL.....	viii
I. PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Tujuan Penelitian.....	4
1.3 Rumusan Masalah	4
1.4 Batasan Masalah.....	4
1.5 Manfaat Penelitian.....	5
1.6 Sistematika Penulisan.....	5
II. TINJAUAN PUSTAKA	7
2.1 Penelitian Terdahulu	7
2.2 Keamanan Jaringan	14
2.3 <i>Website</i>	14
2.4 <i>Hardening</i>	16
2.5 <i>Vulnerability</i>	16
2.5.1 <i>Vulnerability Assessment</i>	16
2.5.2 <i>Vulnerability Scanning</i>	17
2.5.3 <i>OpenVAS</i>	17
2.6 <i>Penetration Testing</i>	34
2.7 <i>Security Information and Event Management (SIEM)</i>	35
2.7.1 <i>Intrusion Detection System (IDS)</i>	35
2.7.2 <i>Intrusion Prevention System (IPS)</i>	36

2.7.3	Suricata.....	37
2.8	<i>Confusion Matrix</i>	59
III.	METODE PENELITIAN	61
3.1	Waktu dan Tempat Penelitian	61
3.2	Alat dan Bahan	61
3.3	Tahapan Penelitian.....	62
3.4	Diagram Alir Penelitian.....	64
3.5	Pengumpulan Data	67
3.6	Perancangan dan Pengembangan Topologi Jaringan.....	68
3.7	<i>Penetration Testing</i>	69
3.8	Tahapan Simulasi.....	76
3.8.1	Simulasi Serangan Tahap 1	76
3.8.2	Simulasi Serangan Tahap 2	80
3.9	Indikator Pencapaian	84
IV.	HASIL DAN PEMBAHASAN.....	85
4.1	Pembahasan	85
4.1.1	Perancangan Server.....	86
4.2	Serangan Tahap Pertama	91
4.2.1	Vulnerability Scanning.....	91
4.2.2	Serangan Menggunakan Teknik <i>SQL Injection</i>	102
4.2.3	Proses Serangan Menggunakan <i>Brute Force</i>	113
4.2.4	Serangan <i>Reverse Shell</i>	115
4.3	Serangan Tahap Kedua	118
4.3.1	Serangan Menggunakan Serangan <i>SQL Injection</i> Tahap Kedua....	126
4.3.2	Serangan Menggunakan Teknik <i>Brute Force</i> Tahap Kedua.....	129
4.3.3	Serangan <i>Reverse Shell</i> Tahap Kedua	130
4.3.4	Percobaan Serangan <i>Multiple Login Failed</i>	131
4.3.5	Percobaan Aktivitas Normal	132
4.3.6	Hasil Integrasi Notifikasi Suricata Menggunakan Telegram	133
4.4	Evaluasi <i>False Positive Rate</i>	136

4.4.1	Data Aktivitas Serangan.....	136
4.4.2	Evaluasi Deteksi Aktivitas Normal	138
4.4.3	Perhitungan Evaluasi Metrik.....	139
V	KESIMPULAN.....	142
5.1	Kesimpulan.....	142
5.2	Saran	143
	DAFTAR PUSTAKA	144
	LAMPIRAN	

DAFTAR GAMBAR

	Halaman
Gambar 2. 1 <i>Roadmap</i> Penelitian Terdahulu.	13
Gambar 2. 2 Contoh <i>Website</i> Statis.....	15
Gambar 2. 3 Contoh <i>Website</i> Dinamis	15
Gambar 2. 4 <i>Dashboard</i> OpenVAS.....	18
Gambar 2. 5 Kelompok metrik CVSS.....	20
Gambar 2. 6 Metrik dan persamaan CVSS	21
Gambar 2. 7 Kelompok metrik CVSS.....	25
Gambar 2. 8 Aliran perhitungan skor CVSS.....	28
Gambar 2. 9 Topologi Dasar Sistem Suricata	38
Gambar 2. 10 <i>OSI Layer</i>	41
Gambar 2. 11 Contoh <i>raw packet</i>	42
Gambar 2. 12 Paket JSON yang sudah diuraikan	45
Gambar 2. 13 Contoh <i>rules alert</i> Suricata untuk paket ICMP	46
Gambar 2. 14 Contoh <i>rules drop</i> Suricata.....	47
Gambar 2. 15 Contoh <i>rules pass</i> Suricata.....	49
Gambar 2. 16 Contoh <i>rules reject</i> Suricata	50
Gambar 2. 17 Aturan <i>iptables FORWARD</i>	52
Gambar 2. 18 Aturan <i>iptables</i> pada <i>port 80</i>	52
Gambar 2. 19 Ilustrasi Lalu Lintas Jaringan HTTP antara Suricata dan Internet	53
Gambar 2. 20 Suricata menggunakan <i>mode NFQUEUE</i>	53
Gambar 2. 21 Contoh paket data normal	55
Gambar 2. 22 Paket data HTTP normal	56
Gambar 2. 23 Paket data tidak normal	58

Gambar 3. 1 Diagram Alir Penelitian.....	67
Gambar 3. 2 Topologi Jaringan Sebelum Diimplementasikan <i>Server</i> IDS/IPS	68
Gambar 3. 3 Topologi Jaringan Setelah Diimplementasikan <i>Server</i> IDS/IPS	69
Gambar 3. 4 Diagram Alir <i>Penetration Testing SQL Injection</i>	71
Gambar 3. 5 Diagram Alir <i>Penetration Testing Brute Force</i>	73
Gambar 3. 6 Diagram Alir <i>Penetration Testing Reverse Shell</i>	75
Gambar 3. 7 Diagram Alir Simulasi Tahap 1	77
Gambar 3. 8 Diagram alir pemindaian Openvas	79
Gambar 3. 9 Diagram Alir Simulasi Tahap 2	81
Gambar 3. 10 Diagram alir sistem kerja Suricata	83
Gambar 4. 1 Konfigurasi Nginx pada <i>server web</i>	86
Gambar 4. 2 Konfigurasi <i>iptables</i> menggunakan NFQUEUE	88
Gambar 4. 3 Konfigurasi <i>iptables</i> NAT <i>forwarding</i>	89
Gambar 4. 4 Hasil <i>Vulnerability Scanning</i> menggunakan OpenVAS	91
Gambar 4. 5 Hasil <i>scanning</i> dengan tingkat <i>severity high</i>	91
Gambar 4. 6 kerentanan arsitektur prosesor.....	94
Gambar 4. 7 kerentanan <i>Speculative Store Bypass (SSB)</i>	95
Gambar 4. 8 kelemahan pada pustaka <i>libde265</i>	97
Gambar 4. 9 Hasil pemindaian kerentanan pada pustaka <i>libfcgi</i> melalui OpenVAS. .	99
Gambar 4. 10 Konfigurasi <i>proxy Burp Suite</i>	102
Gambar 4. 11 Fitur <i>intercept</i> pada <i>Burp Suite</i>	103
Gambar 4. 12 Eksploitasi menggunakan <i>payload 'ORDER BY 1-- -</i>	104
Gambar 4. 13 <i>Database error</i> pada percobaan ke-5 menggunakan <i>payload ' ORDER BY n-- -</i>	105
Gambar 4. 14 Penggunaan <i>payload ' UNION SELECT 1,2,3,4-- -</i>	107
Gambar 4. 15 Penggunaan <i>payload ' UNION SELECT 1,database(),3,4 -- -</i> untuk mengetahui nama <i>database</i> yang digunakan oleh <i>web application</i>	109

Gambar 4. 16 Penggunaan <i>payload</i> ' <i>UNION SELECT 1,2,table_name,4 FROM information_schema.tables WHERE table_schema = database() LIMIT 1 OFFSET 0--</i> - untuk mencari tabel yang ada pada <i>database</i>	110
Gambar 4. 17 Eksploitasi data sensitif menggunakan <i>payload</i> ' <i>UNION SELECT id,nama,username,password FROM admin LIMIT 1 OFFSET 0--</i> -	111
Gambar 4. 18 Hasil serangan <i>brute force</i> sebelum diimplementasikan IDS/IPS.....	113
Gambar 4. 19 <i>Reverse shell</i> berhasil didapatkan melalui <i>Netcat</i> pada <i>port</i> 1234, memberikan akses <i>terminal</i> sebagai <i>user www-data</i>	114
Gambar 4. 20 <i>User www-data</i> memiliki hak istimewa penuh (<i>NOPASSWD: ALL</i>) untuk menjalankan perintah sebagai <i>root</i> tanpa diminta <i>password</i>	115
Gambar 4. 21 <i>File passwd</i> yang berisi semua <i>user</i> pada <i>server web</i> target.....	116
Gambar 4. 22 Konfigurasi <i>Suricata</i> menggunakan mode <i>nfq</i> untuk <i>NFQUEUE</i>	117
Gambar 4. 23 <i>Rules drop</i> yang digunakan untuk serangan <i>SQL Injection</i> dengan <i>payload UNION SELECT</i>	119
Gambar 4. 24 <i>Rules drop</i> yang digunakan untuk serangan <i>SQL Injection</i> dengan <i>payload ORDER BY n</i>	120
Gambar 4. 25 <i>Rules alert</i> untuk serangan <i>multiple login failed</i>	121
Gambar 4. 26 <i>Rules drop</i> untuk serangan <i>brute force</i>	122
Gambar 4. 27 <i>Rules drop</i> untuk <i>upload file</i> yang terindikasi berbahaya	122
Gambar 4. 28 <i>source code</i> integrasi <i>log Suricata</i> dengan <i>bot Telegram</i>	124
Gambar 4. 29 Hasil pengujian menggunakan <i>payload</i> ' <i>UNION SELECT 1,2,database(),4--</i> - setelah diimplementasikan IDS/IPS menggunakan <i>Suricata</i>	125
Gambar 4. 30 Hasil pengujian menggunakan <i>payload</i> ' <i>ORDER Byn--</i> - setelah diimplementasikan IDS/IPS menggunakan <i>Suricata</i>	126
Gambar 4. 31 Hasil dari serangan <i>brute force</i> setelah diimplementasikan IDS/IPS.	128
Gambar 4. 32 Koneksi terputus saat <i>upload file</i> berbahaya.....	129
Gambar 4. 33 Notifikasi <i>Telegram</i> Serangan <i>Brute Force</i>	133
Gambar 4. 34 Notifikasi <i>Telegram</i> Serangan <i>SQL Injection</i>	133
Gambar 4. 35 Notifikasi <i>Telegram</i> Serangan <i>Reverse Shell</i>	134

DAFTAR TABEL

	Halaman
Tabel 2. 1 Penelitian Terdahulu	11
Tabel 2. 2 <i>Metrics Value</i> CVSS Versi 2.0.....	21
Tabel 2. 3 Skala Penilaian Keparahan Kualitatif	29
Tabel 2. 4 <i>Metric Value</i>	30
Tabel 2. 5 <i>Confusion Matrix</i>	59
Tabel 4. 1 Spesifikasi <i>Server</i> Penyerang	85
Tabel 4. 2 Spesifikasi <i>Server Web</i>	86
Tabel 4. 3 Spesifikasi <i>Server</i> Suricata	87
Tabel 4. 4 Jenis Aktivitas Normal yang Diuji	132
Tabel 4. 5 Aktivitas Serangan	129
Tabel 4. 6 Aktivitas Normal	131
Tabel 4. 7 Hasil Perhitungan Evaluasi Metrik	132

I. PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi internet saat ini berkembang sangat pesat. Setiap perusahaan membutuhkan alat untuk menyimpan informasi untuk menyimpan data-data perusahaan maupun *customer* dengan aman. Hal ini sangat membantu sebuah perusahaan menyimpan data-data tersebut, namun tidak menutup kemungkinan bahwa data yang sudah disimpan akan aman. Besar kemungkinan bahwa data pada perusahaan dapat dicuri dengan menyerang *database* dari perusahaan tersebut.

Menurut jurnal penelitian yang berjudul Analisis Keamanan *Website Leads UPNVJ Terhadap Serangan SQL Injection & Sniffing Attack*, *SQL injection* adalah teknik yang dapat mengeksploitasi kueri dari *structured query language* (SQL) untuk bisa menembus menuju *back-end* dari *database*. Jika sudah menembus *database*, penyerang mendapat keleluasaan dalam mendapatkan informasi sensitif yang terdapat pada *database* tersebut, seperti *username*, *password*, nama, alamat, nomor telepon, dan lain-lain [1].

Menurut jurnal penelitian *A Classification of SQL Injection Attacks and Countermeasures*, *web application* yang rentan terhadap *SQL Injection* memungkinkan penyerang mendapatkan akses lengkap ke *database* dasarnya. Karena *database* ini sering berisi informasi konsumen atau pengguna yang sensitif, pelanggaran keamanan yang dihasilkan dapat mencakup pencurian identitas, kehilangan informasi rahasia, dan penipuan. Dalam beberapa kasus, penyerang bahkan dapat menggunakan kerentanan

SQL *Injection* untuk mengendalikan dan merusak sistem yang menghosting *web application* [2].

Karena kerentanan pada *web application* tersebut berbahaya, maka dibutuhkan pencegahan agar permasalahan tersebut dapat diminimalisir. Untuk meminimalisir kerentanan keamanan jaringan, dibutuhkan tindakan untuk meningkatkan keamanan sistem komputer, jaringan, atau aplikasi, yang disebut dengan *hardening*. Menurut jurnal penelitian yang berjudul Implementasi Keamanan Jaringan Pada Cloudfri Dengan Metode *Hardening*, fungsi *hardening* adalah mengurangi resiko keamanan yang menghindari potensi serangan dan melakukan pengerasan permukaan sistem dari serangan. Proses *hardening* bertujuan pada metode untuk menganalisa, mengidentifikasi, menutup, dan mengontrol kelemahan dan kerentanan keamanan jaringan pada organisasi [3].

Untuk mengimplementasikan keamanan jaringan dan untuk meminimalisir sebelum terjadinya serangan dari seorang *network attacker* atau yang biasa disebut dengan *hacker*, sangat penting untuk melakukan analisa pada kerentanan sebuah keamanan jaringan. Analisa kerentanan pada sebuah keamanan jaringan disebut dengan *Vulnerability Assessment*. Menurut jurnal penelitian yang berjudul *Vulnerability Assessment and Penetration Testing*, *vulnerability* atau kerentanan sebenarnya adalah kelemahan dari sebuah sistem. Ini menghasilkan kemungkinan penetrasi ke dalam sistem yang dapat mengakibatkan akses tidak sah. *Vulnerability Assessment* adalah tindakan *scanning* yang digunakan untuk menentukan, menghitung, dan memberi peringkat kerentanan dalam sistem. Dalam proses ini, seperti sistem operasi & perangkat lunak aplikasi dan jaringan dipindai untuk mengidentifikasi terjadinya kerentanan yang diketahui dan tidak diketahui. Kerentanan ini terjadi karena desain perangkat lunak yang tidak sesuai, otentikasi tidak aman atau bahkan banyak kerentanan terjadi akibat kesalahan konfigurasi [4].

Berdasarkan jurnal *CyberSecurity* dan Forensik Digital dengan judul Analisis Serangan Router dengan *Security Information and Event Management* dan Implikasinya pada Indeks Keamanan Informasi yang ditulis oleh Citra Arfanudin dan kawan-kawan, menjelaskan bahwa SIEM (*Security Information and Event Management*) dapat mengenali serangan siber. Contohnya SIEM dapat mengenali serangan DHCP *Starvation*, DHCP *Rouge*, SSH *Bruteforce* dan FTP *Bruteforce*. Pada penelitian ini juga didapatkan bahwa dengan mengimplementasikan SIEM dapat menaikkan indeks Keamanan Informasi (KAMI) [5].

Blacklisted IP Distribution System to handle DdoS attacks on IPS Snort based on Blockchain melakukan simulasi serangan pada sebuah *host* yang telah terpasang IPS dan *Blockchain Node* sebagai pengamanan. Pada penelitian ini juga membuat program *Blacklisted IP Distribution* menggunakan *Phyton* untuk memblokir IP yang diduga mencurigakan. Melalui simulasi yang telah dilakukan, Snort dapat memperoleh informasi serangan sumber serangan baru dari *blockchain* dalam jangka waktu antara 3 samppai 7 detik. Namun pada penelitian ini masih dapat dikembangkan dengan mengubah jenis serangan yang dilakukan [6].

Berdasarkan data di atas maka peneliti memilih judul berupa Analisis Keamanan Jaringan Menggunakan Metode *Vulnerability Assessment* dan *Penetration Testing* dengan Mengimplementasikan IDS/IPS Menggunakan Suricata, yang mana untuk melakukan analisis kerentanan terhadap suatu *website* dengan melakukan *vulnerability scanning* untuk menilai kerentanan dan melakukan simulasi serangan serta memonitoring *log event* pada *website* tersebut dengan mengimplementasikan *Intrusion Detection System* (IDS) untuk tindakan deteksi pada *log event* dan *Intrusion Prevention System* (IPS) sebagai tindakan pencegahan jika ada paket yang mencurigakan.

1.2 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Menganalisis kerentanan pada *website* dengan menggunakan metode *Vulnerability Assessment*.
2. Melakukan *Penetration Testing* sebagai tindakan pengujian terhadap *website*.
3. Mengimplementasikan *Intrusion Detection System* (IDS) untuk tindakan deteksi pada *log event* dan *Intrusion Prevention System* (IPS) sebagai tindakan pencegahan jika ada paket yang mencurigakan.

1.3 Rumusan Masalah

Berdasarkan latar belakang di atas, maka peneliti merumuskan permasalahannya sebagai berikut:

1. Bagaimana cara mengetahui kerentanan sebuah keamanan jaringan pada *website*?
2. Bagaimana proses terjadinya serangan pada *website*?
3. Apakah semua *log* yang terjadi dapat direkam dan dicegah pada saat terjadi serangan?

1.4 Batasan Masalah

Adapun batasan masalah yang diterapkan oleh penulis dalam melaksanakan penelitian ini meliputi beberapa hal, yaitu sebagai berikut:

1. Analisa dengan metode *Vulnerability Assessment* menggunakan tools OpenVAS dan Implementasi IDS/IPS menggunakan tools Suricata.
2. Parameter pengujian hanya membahas apakah paket yang mencurigakan dapat dideteksi oleh IDS dan mengirimkan pesan peringatan, serta diblokir oleh IPS.

1.5 Manfaat Penelitian

Adapun manfaat penelitian ini adalah sebagai berikut:

1. Penelitian ini memberikan informasi dan gambaran bagaimana seorang *hacker* melakukan serangan terhadap *website*.
2. Dengan mengimplementasikan IDS/IPS dapat membantu seorang *network defender* untuk memonitoring dan mengumpulkan informasi semua aktivitas yang masuk pada *server* berupa *log*, serta dapat memblokir paket-paket yang mencurigakan.

1.6 Sistematika Penulisan

Sistematika penulisan laporan akhir bertujuan guna memberikan suatu gambaran secara sederhana terkait pembahasan yang ada di dalam tugas akhir skripsi dan untuk memudahkan dalam memahami isi yang disajikan dalam skripsi ini. Adapun sistematika yang digunakan oleh penulis adalah sebagai berikut:

BAB I. PENDAHULUAN

Bab ini berisikan latar belakang dan masalah, tujuan penelitian, rumusan masalah, batasan masalah, manfaat penelitian, dan sistematika penelitian

BAB II. TINJAUAN PUSTAKA

Bab ini berisikan teori pendukung materi penelitian yang merupakan pengantar dalam pemahaman terkait materi tugas akhir yang dikutip dari berbagai sumber ilmiah, seperti jurnal, proseding, dan buku yang digunakan sebagai panduan dalam penulisan laporan tugas akhir ini

BAB III. METODE PENELITIAN

Bab ini berisikan tentang waktu dan tempat penelitian, alat dan bahan yang digunakan dalam menyelesaikan tugas akhir ini, metode yang digunakan, dan pelaksanaan serta pengamatan dalam pengerjaan tugas akhir.

BAB IV. HASIL DAN PEMBAHASAN

Bab ini berisikan tentang hasil dan pembahasan data simulasi dan memaparkan data yang didapat dari penelitian ini.

BAB V. KESIMPULAN DAN SARAN

Bab ini berisikan tentang kesimpulan yang diperoleh dari pembahasan pada laporan penelitian dan saran yang berhubungan dengan pokok-pokok pembahasan penelitian ini.

II. TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Vulnerability Assessment and Penetration Testing yang ditulis oleh Sachin Umrao, Mandeep Kaur dan Govind Kumar Gupta, melakukan VAPT (*Vulnerability Assessment and Penetration Testing*) untuk mengidentifikasi kerentanan pada sebuah sistem agar meminimalisir serangan *cyber* dari peretas yang memanfaatkan kerentanan untuk mengakses data rahasia. *Penetration Testing* melibatkan serangkaian aktivitas yang dilakukan untuk mengidentifikasi dan mengeksploitasi kerentanan keamanan suatu sistem [4].

Pada penelitian yang berjudul *Surviving Unpatchable Vulnerabilities Through Heterogeneous Network Hardening Options*, mempelajari bagaimana menghadapi kerentanan yang tidak diketahui. Tujuan dari penelitian ini adalah untuk melakukan *hardening* dengan menyatukan beberapa opsi *hardening*, seperti diversifikasi layanan, modifikasi aturan *firewall*, menambah, menghapus, dan merelokasi sumber daya jaringan, dan kontrol akses, semuanya dibawah model yang sama. Kemudian menerapkan metrik keamanan yang dirancang untuk mengevaluasi ketahanan jaringan terhadap kerentanan yang tidak diketahui dan tidak dapat diperbaiki, dan akibatnya mendapatkan solusi optimal untuk memaksimalkan keamanan di bawah batasan biaya yang diberikan [7].

Analysis and Impact of Vulnerability Assessment and Penetration Testing merupakan makalah yang membahas dan menganalisis tentang proses VAPT (*Vulnerability*

Assessment and Penetration Testing) dan alat VAPT untuk menemukan kerentanan dalam sistem. Pada makalah ini memfokuskan kepentingannya di berbagai tingkat organisasi untuk persyaratan adopsi memperbarui langkah-langkah keamanan untuk memberikan perlindungan dari berbagai serangan dunia maya [8].

Pada sebuah penelitian yang berjudul *Implementasi Metode Vulnerability dan Hardening Pada Sistem Keamanan Jaringan*, menjelaskan untuk melakukan optimasi keamanan jaringan dengan pemodelan penilaian kerentanan (*Vulnerability Assessment*) dan proses hardening pada sistem dan desain jaringan komputer untuk mengukur tingkat kerentanan dan mengkategorikan aset jaringan yang kriptikal, yang selanjutnya dilakukan perbaikan pada sistem dan desain jaringan sesuai dengan standar keamanan jaringan. Pada tindakan *hardening* dapat meminimalkan tingkat resiko kerentanan dengan melakukan penguatan terhadap sistem, konfigurasi, dan desain topologi pada infrastruktur jaringan computer [9].

Blacklisted IP Distribution System to handle DdoS attacks on IPS Snort based on Blockchain melakukan simulasi serangan pada sebuah *host* yang telah terpasang IPS dan *Blockchain Node* sebagai pengamanan. Pada penelitian ini juga membuat program *Blacklisted IP Distribution* menggunakan *Phyton* untuk memblokir IP yang diduga mencurigakan. Melalui simulasi yang telah dilakukan, Snort dapat memperoleh informasi serangan sumber serangan baru dari *blockchain* dalam jangka waktu antara 3 samppai 7 detik. Namun pada penelitian ini masih dapat dikembangkan dengan mengubah jenis serangan yang dilakukan [6].

Analisa Keamanan Jaringan Komputer Menggunakan Sistem Deteksi Intrusi Shorewal, merupakan jurnal penelitian yang membahas tentang keamanan jaringan dengan membangun deteksi intrusi pada jaringan komputer di SMK N 3 Kepahiang dengan memanfaatkan metode *Network Intrusion Detection System (NIDS)* menggunakan aplikasi Shorewall, serta mengetahui cara kerja dari Shorewall dalam mencegah dan mengatasi sistem keamanan jaringan komputer. Hasil dari penelitian ini aplikasi

Shorewall dapat digunakan untuk sebagai sistem deteksi intrusi untuk *Mac clone* dan *DdoS attack* dalam jaringan, sehingga dapat melakukan *reject* pada aktivitas tersebut [10].

Pada jurnal yang berjudul Analisis Keamanan Jaringan Komputer dengan Penerapan Teknologi Firewall dan Intrusion Detection System (IDS), melakukan implementasi *firewall* dan IDS pada jaringan komputer pada sebuah perusahaan guna meningkatkan keamanan jaringan. Hasil pada penelitian ini *firewall* dapat mencegah serangan yang masuk pada *server* dan IDS dapat mendeteksi serangan yang masuk pada *server*, namun masih ada serangan yang tidak dapat di cegah oleh *firewall* seperti *malware* dan *phising attack* [11].

Pada jurnal *Inspecting Intrusion Prevention System Signatures for False Blocking using Set Theory* melakukan percobaan menggunakan Snort *Policy* yang banyak digunakan dalam operasi keamanan untuk mengevaluasi keefektifannya. Hasil dari percobaan pada jurnal ini mengungkapkan bahwa metode yang diusulkan mencapai tingkat deteksi 100% dan dieksekusi lebih cepat daripada metode konvensional untuk hampir semua representasi masukan pengguna [12].

Pada jurnal Analisis dan Implementasi *Network Security System* Menggunakan Teknik *Host-Based Intrusion Detection System* (HIDS) Berbasis *Cloud Computing*, membahas tentang penggunaan OSSEC sebagai tools *Host-Based Intrusion Detection System* (HIDS). Pada jurnal ini menguji sistem OSSEC untuk mendeteksi adanya serangan yang dilakukan oleh penguji menggunakan teknik *Penetration Testing*. Hasil uji yang terhadap kinerja sistem menunjukkan bahwa pada parameter keakurasian pendeteksian bahwa sistem OSSEC dapat mendeteksi secara akurat dari serangan yang masuk, serta mampu mendeteksi serangan yang masuk pada sistem dengan cepat dan hanya memakai penggunaan CPU dan RAM sehingga tidak memberatkan *server* [13].

Menurut jurnal penelitian berjudul "Analisa Penerapan *Honeypot Cowrie* dan IPS untuk Meningkatkan Keamanan *Web Server*" yang diterbitkan pada Seminar Nasional CORISINDO tanggal 12 September 2025, efektivitas sistem keamanan *web server* dapat ditingkatkan secara signifikan melalui pendekatan integratif yang menggabungkan *Honeypot Cowrie* dan *Intrusion Prevention System (IPS)* Suricata. *Cowrie* berfungsi sebagai sistem jebakan berbasis SSH/Telnet untuk memancing, memantau, dan mencatat aktivitas penyerang, termasuk percobaan *login* dan perintah berbahaya, sedangkan Suricata bertindak sebagai pertahanan aktif yang mendeteksi dan secara otomatis mencegah atau memblokir lalu lintas mencurigakan secara *real-time*. Hasil pengujian menunjukkan bahwa kombinasi kedua sistem ini terbukti mampu mendeteksi dan merespons berbagai jenis serangan seperti *brute force*, *port scanning*, dan DDoS [14].

Menurut dokumen penelitian berjudul "*Overview on Intrusion Detection Systems for Computers Networking Security*" yang diterbitkan dalam jurnal *Computers*, pertumbuhan pesat komunikasi digital dan pertukaran data yang luas telah menjadikan jaringan komputer sebagai bagian integral dari operasi organisasi. Namun, peningkatan konektivitas ini juga memperluas permukaan serangan (*attack surface*), yang memperkenalkan risiko keamanan yang signifikan. Dokumen ini menyajikan tinjauan komprehensif tentang teknologi *Intrusion Detection System (IDS)* untuk keamanan jaringan, menguji metode tradisional dan kemajuan terbaru. Tinjauan tersebut mencakup arsitektur dan jenis-jenis IDS, teknik deteksi utama, dataset dan lingkungan pengujian, serta implementasi dalam lingkungan jaringan modern seperti *cloud computing*, jaringan virtualisasi, *Internet of Things (IoT)*, dan sistem kontrol industri [15].

Penelitian terdahulu yang berkaitan dengan penelitian skripsi ini dapat dilihat secara keseluruhan pada Tabel 2.1

Tabel 2. 1 Penelitian Terdahulu

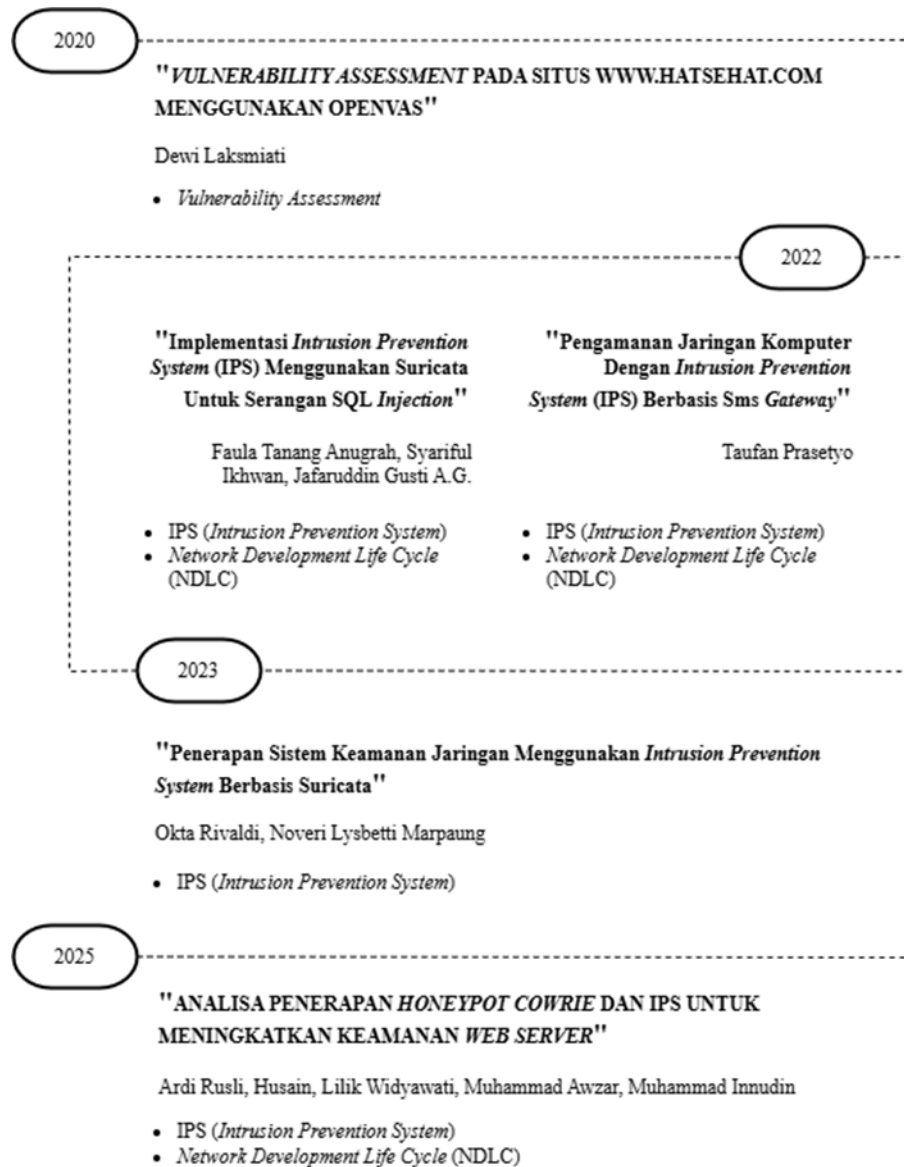
No	Judul Jurnal	Penulis/Tahun	Metode	Hasil
1	<i>Vulnerability Assessment</i> pada Situs www.hatsehat.com Menggunakan OpenVAS	Dewi Laksmiati, 2020	<i>Vulnerability Assessment</i>	Pengujian menggunakan OpenVAS pada situs tersebut menemukan total 15 celah keamanan level <i>High</i> dan 48 level <i>Medium</i> .
2	Pengamanan Jaringan Komputer Dengan <i>Intrusion Prevention System</i> (IPS) Berbasis SMS Gateway	Taufan Prasetyo, 2022	<i>Intrusion Prevention System</i> (IPS), <i>Network Development Life Cycle</i> (NDLC)	Sistem dapat mendeteksi dan mencegah serangan serta mengirim notifikasi SMS otomatis meski masih terdapat <i>delay</i> .
3	Implementasi <i>Intrusion Prevention System</i> (IPS) Menggunakan Suricata Untuk Serangan SQL Injection	Faula Tanang Anugrah, Syariful Ikhwan, Jafaruddin Gusti A.G., 2022	<i>Intrusion Prevention System</i> (IPS), <i>Network Development Life Cycle</i> (NDLC)	Suricata berhasil mendeteksi dan memblokir semua jenis SQL Injection dengan rata-rata <i>response time</i> 4,26 ms.
4	Penerapan Sistem Keamanan Jaringan Menggunakan <i>Intrusion Prevention System</i>	Okta Rivaldi, Noveri Lysbetti Marpaung, 2023	<i>Intrusion Prevention System</i> (IPS)	Suricata berhasil mendeteksi dan memblokir <i>Syn Flood</i> , <i>Port Scanning</i> , dan <i>Ping of Death</i> secara <i>real-time</i> .

No	Judul Jurnal	Penulis/Tahun	Metode	Hasil
	Berbasis Suricata			
5	Analisa Penerapan Honeypot Cowrie dan IPS Untuk Meningkatkan Keamanan Web Server	Ardi Rusli, Husain, Lilik Widyawati, Muhammad Awzar, Muhammad Innudin, 2025	Intrusion Prevention System (IPS), Network Development Life Cycle (NDLC)	Kombinasi Honeypot Cowrie dan IPS Suricata terbukti efektif mendeteksi serangan (seperti brute force, port scanning, dan DDoS) secara waktu-nyata, mencatat log detail, serta melakukan pencegahan otomatis pada web server.

Meskipun banyak penelitian sebelumnya telah membahas keamanan jaringan, sebagian besar masih memiliki ruang lingkup yang terbatas, seperti hanya berfokus pada tahap *Vulnerability Assessment* tanpa melanjutkan ke proses eksploitasi atau implementasi sistem pertahanan yang nyata. Beberapa studi juga hanya menguji IDS/IPS pada satu jenis serangan tertentu atau mengintegrasikannya dengan konteks khusus yang tidak mencerminkan kondisi ancaman yang beragam. Keterbatasan tersebut menunjukkan bahwa penelitian terdahulu umumnya belum menerapkan evaluasi keamanan yang bersifat menyeluruh dan belum mencakup seluruh tahapan penting dalam siklus pertahanan jaringan.

Penelitian ini menggunakan pembaruan dengan menerapkan metodologi VAPT (*Vulnerability Assessment* dan *Penetration Testing*) secara komprehensif, dimulai dari proses identifikasi kerentanan menggunakan OpenVAS hingga simulasi berbagai serangan siber seperti *SQL Injection*, *Brute Force*, dan *Reverse Shell*. Setelah tahap eksploitasi, penelitian ini melanjutkan dengan implementasi Suricata dalam mode ganda (IDS dan IPS) untuk memberikan gambaran yang lebih realistis mengenai

efektivitas sistem pertahanan. Evaluasi kuantitatif menunjukkan tingkat akurasi deteksi sebesar 90,91%, sehingga memperlihatkan kemampuan yang kuat dalam menghadapi ancaman nyata. Dengan cakupan yang meliputi audit, eksploitasi, pertahanan, dan evaluasi, penelitian ini memberikan pendekatan yang lebih menyeluruh dibandingkan studi-studi sebelumnya.



Gambar 2. 1 Roadmap Penelitian Terdahulu.

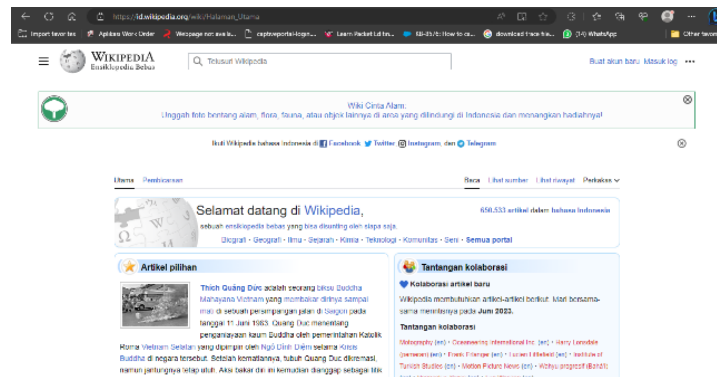
2.2 Keamanan Jaringan

Keamanan jaringan adalah serangkaian tindakan yang bertujuan untuk mengamankan sistem komputer dalam jaringan. Keamanan jaringan membantu melindungi sistem komputer dalam jaringan dari intrusi yang tidak diinginkan dan mengurangi kemungkinan menjadi korban pencurian data. Dengan menggunakan alat keamanan jaringan, kita dapat menemukan celah keamanan sebelum informasi tersebut disalahgunakan oleh peretas [16].

Keamanan jaringan komputer melibatkan empat hubungan yang berbeda, yaitu potensi hubungan dengan empat aspek utama ketika menggambarkan bentuk-bentuk ancaman terhadap keamanan jaringan komputer. Ada empat bentuk utama ancaman terhadap keamanan jaringan komputer: penyalahgunaan informasi Internet of Things, penolakan layanan serangan latar belakang, kerusakan pada integritas lingkungan jaringan komputer, dan kebocoran informasi komputer [17].

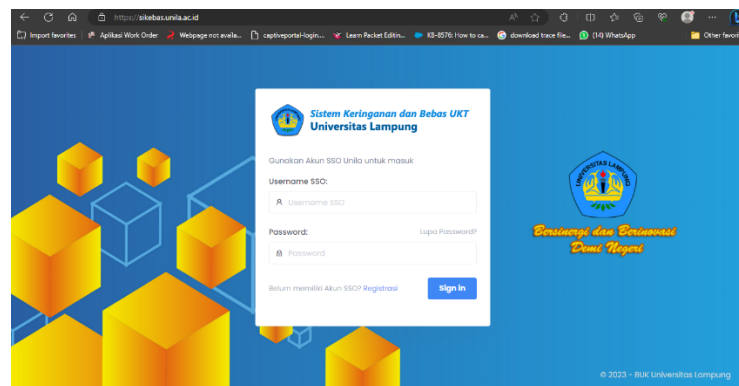
2.3 Website

Website adalah sebutan bagi sekelompok halaman *web* (*web page*), yang umumnya merupakan bagian dari suatu nama *domain* (*domain name*) atau *subdomain* di *World Wide Web* (WWW) di Internet. Sebuah *web page* adalah dokumen yang ditulis dalam format HTML (*Hyper Text Markup Language*), yang hampir selalu bisa diakses melalui HTTP, yaitu protokol yang menyampaikan informasi dari server *website* untuk ditampilkan kepada para pemakai melalui *web browser* baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait dimana masing-masing dihubungkan dengan jaringan-jaringan halaman. Dalam sisi pengembangannya, *website* statis hanya bisa diupdate oleh pemiliknya saja.



Gambar 2. 2 Contoh *Website* Statis

Bersifat statis apabila isi informasi *website* tetap, jarang berubah, dan isi informasinya searah hanya dari pemilik *website*, seperti yang ditunjukkan pada Gambar 2.2. Contoh *website* statis adalah berisi profil perusahaan. Bersifat dinamis apabila isi informasi *website* selalu berubah-ubah, dan isi informasinya interaktif dua arah berasal dari pemilik serta pengguna *website*. Contoh *website* dinamis adalah seperti *Friendster*, *Multiply*, *Portal Website*, dll. Dalam sisi pengembangannya, *website* dinamis bisa diupdate oleh pengguna maupun pemilik. Contoh *website* dinamis seperti yang ditunjukkan pada Gambar 2.3.



Gambar 2. 3 Contoh *Website* Dinamis

2.4 *Hardening*

Prosedur yang meminimalkan ancaman yang datang dengan mengatur konfigurasi dan menonaktifkan aplikasi dan layanan yang tidak diperlukan. Instalasi *firewall*, instalasi *antivirus*, menghapus *cookie*, membuat *password*, menghapus program yang tidak diperlukan [9]. Tujuan dari *Hardening* adalah untuk menghilangkan resiko ancaman yang bisa terjadi pada komputer, hal ini biasanya dilakukan dengan menghapus semua *program/file* maupun *port-port* (tcp/udp) yang tidak diperlukan, memasang *firewall* maupun *Web Application Firewall*.

2.5 *Vulnerability*

Vulnerability (kerentanan) adalah kelemahan atau cacat dalam suatu sistem yang dapat dieksploitasi oleh penyerang untuk mendapatkan akses yang tidak sah, mengganggu operasi normal sistem, atau mencuri informasi sensitif. Kerentanan dapat ditemukan di berbagai tingkat sistem, termasuk perangkat keras, perangkat lunak, dan manusia [18].

2.5.1 *Vulnerability Assessment*

Vulnerability Assessment adalah proses mendefinisikan, mengidentifikasi, mengklasifikasikan, dan memprioritaskan kerentanan dalam sistem komputer, aplikasi, dan infrastruktur jaringan dan memberikan organisasi melakukan penilaian dengan pengetahuan, kesadaran, dan latar belakang risiko yang diperlukan untuk memahami ancaman terhadap lingkungannya dan bereaksi dengan tepat [18].

Dengan teknik *Vulnerability Assessment* pengguna dapat menemukan kerentanan yang dapat mengakibatkan berbagai serangan berbahaya seperti serangan *Denial of Service* (DoS), *Router Advertisements* (RA) *Flooding*, *Address Resolution Protocol* (ARP) *Poisoning*, *SQL Injection* pada sistem dan menerapkan *vulnerability assessment* dan

penetration testing pada sistem. Ketika daftar kerentanan diperoleh oleh administrator, mereka dapat menyingkirkan kerentanan tersebut. Untuk menghilangkan kerentanan, administrator harus menerapkan tambalan yang tersedia, pembaruan, menginstal perangkat lunak yang diperlukan [8].

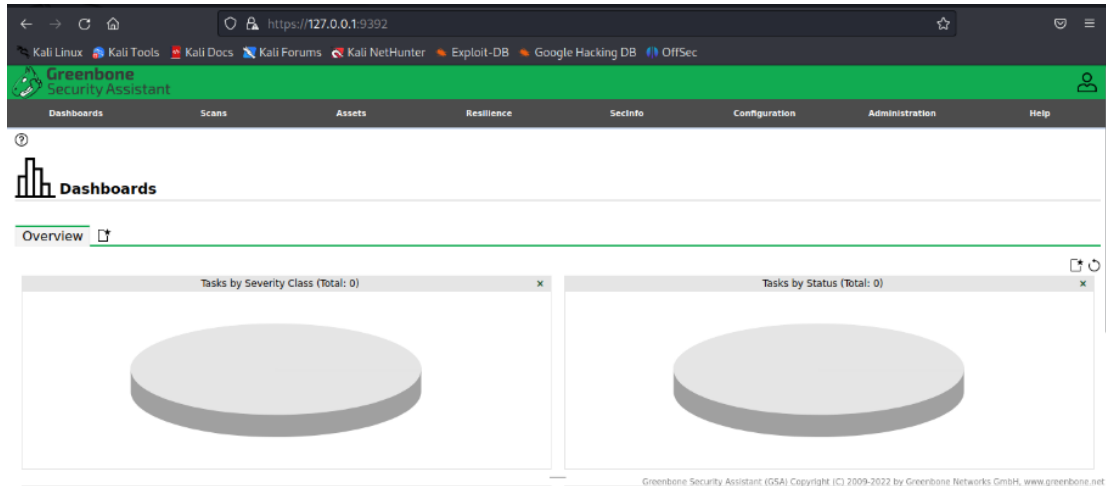
2.5.2 *Vulnerability Scanning*

Aplikasi yang menjalankan security scan dan penilaian kerentanan sebenarnya menjalankan scanning dan kalkulasi, bukan pada bagaimana kerentanan dideteksi melainkan pada apa yang rentan [9]. *Vulnerability assessment* bergantung pada penemuan berbagai jenis kerentanan sistem atau jaringan yang ditemukan dengan menerapkan *vulnerability scanning*, yang berarti proses penilaian mencakup penggunaan berbagai alat, pemindai, dan metodologi untuk mengidentifikasi kerentanan, ancaman, dan risiko [18].

2.5.3 OpenVAS

Open Vulnerability Assessment System menyediakan berbagai layanan, memberikan solusi manajemen kerentanan yang luar biasa. OpenVAS adalah alat pemindaian sumber terbuka. OpenVAS bekerja pada arsitektur *Client-server*. *UI web* OpenVAS adalah komponen klien tempat kita dapat mengonfigurasi pemindaian, meluncurkan pemindaian, dan meninjau dokumen hasil. OpenVAS adalah pemindai jarak jauh karena tidak perlu dipasang di setiap sistem. OpenVAS dapat melakukan *scanning* sampai lebih dari 53.000 scanning NVT [16]. Open Vulnerability Assessment System (OpenVAS) merupakan komponen utama dari Greenbone Vulnerability Management (GVM) yang berfungsi sebagai sistem pemindai kerentanan jaringan. OpenVAS bekerja dengan pendekatan sistematis untuk melakukan pemindaian terhadap perangkat atau layanan dalam suatu jaringan guna mendeteksi celah keamanan

(vulnerability) berdasarkan basis data kerentanan yang terus diperbarui. Tampilan *dashboard* OpenVAS dapat dilihat pada Gambar 2.4.



Gambar 2. 4 *Dashboard* OpenVAS

Sebelum pemindaian dimulai, OpenVAS akan menggunakan basis data kerentanan yang disebut *Network Vulnerability Tests* (NVTs). NVT adalah sekumpulan skrip deteksi kerentanan yang dikembangkan dalam bahasa NASL (*Nessus Attack Scripting Language*). Skrip ini diperoleh melalui *feed* resmi dari *Greenbone* dan memuat informasi tentang ribuan kerentanan yang diidentifikasi berdasarkan sistem klasifikasi CVE (*Common Vulnerabilities and Exposures*). NVT-NVT ini memiliki signature atau pola tertentu yang sesuai dengan karakteristik kerentanan yang dikenal. Saat proses pemindaian dijalankan, skrip-skrip ini digunakan untuk menguji target.

Target pemindaian dilakukan oleh pengguna melalui antarmuka web (GSA) atau perintah baris. Target dapat berupa alamat IP tunggal, rentang IP, atau domain. Pengguna juga dapat menentukan jenis pemindaian yang diinginkan, seperti pemindaian lengkap, pemindaian *port* spesifik, dan pemindaian *web application*, yang selanjutnya akan di teruskan ke *Greenbone Vulnerability Manager* (gvmd) sebagai pengelola proses. OpenVAS melakukan proses pemindaian secara bertahap dengan urutan sebagai berikut:

a. *Host Discovery*

Sistem akan mendeteksi apakah host aktif atau tidak dengan menggunakan teknik ICMP *echo request* (ping), TCP SYN ping, DNS *reverse lookup*. Jika tidak ada *respons*, *host* dianggap tidak aktif dan dilewati dalam proses lanjutan.

b. *Port Scanning*

Scanner melakukan pemindaian terhadap *port* TCP dan UDP untuk mengetahui layanan apa saja yang sedang berjalan. Ini dapat dilakukan secara intensif (*full scan*) atau terbatas (*top ports only*).

c. *Service Detection*

Setelah *port* diketahui, sistem akan mencoba mengenali layanan (*service*) apa yang berjalan di balik *port* tersebut, seperti HTTP, HTTPS, FTP, SSH, SMB, *database server* (MySQL, PostgreSQL, MSSQL). Deteksi ini dilakukan melalui metode *Banner grabbing* (mengambil identitas layanan) dan *Fingerprinting* (membandingkan respon layanan terhadap pola yang dikenal).

d. *Vulnerability Testing* (Eksekusi NVT)

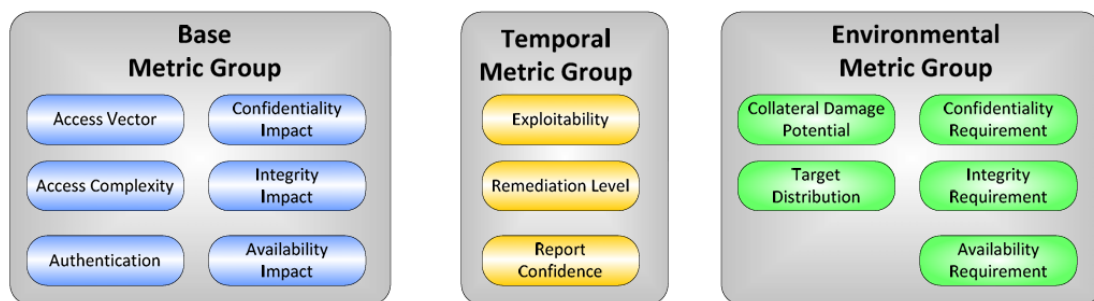
Setelah layanan diidentifikasi, OpenVAS akan mencocokkan jenis layanan dengan skrip NVT yang relevan. Skrip NVT dijalankan untuk menguji CVE-CVE yang diketahui menyerang versi layanan yang akan dipindai kerentanannya dengan mengirimkan permintaan khusus (*payload*) ke server, mengamati respon apakah ada gejala kerentanan, dan menandai kerentanan jika pola cocok dengan signature yang ada.

Setelah NVT dijalankan, hasil kerentanan akan dikategorikan menjadi *low*, *medium*, *high*, dan *critical*. OpenVAS menggunakan analisis berbasis signature dan perilaku. OpenVAS tidak melakukan eksploitasi secara penuh, namun cukup untuk memastikan kerentanan ada tanpa menyebabkan kerusakan.

Penilaian tingkat keparahan kerentanan oleh OpenVAS tidak hanya terbagi menjadi Tinggi (*High*) dan Rendah (*Low*), melainkan diklasifikasikan ke dalam empat kategori utama, yaitu *low*, *medium* dan *high*. Klasifikasi ini didasarkan pada Skor Dasar CVSS (*CVSS Base Score*), yaitu nilai numerik pada skala 0.0 hingga 10.0.

2.5.3.1 Common Vulnerability Scoring System (CVSS) Versi 2.0

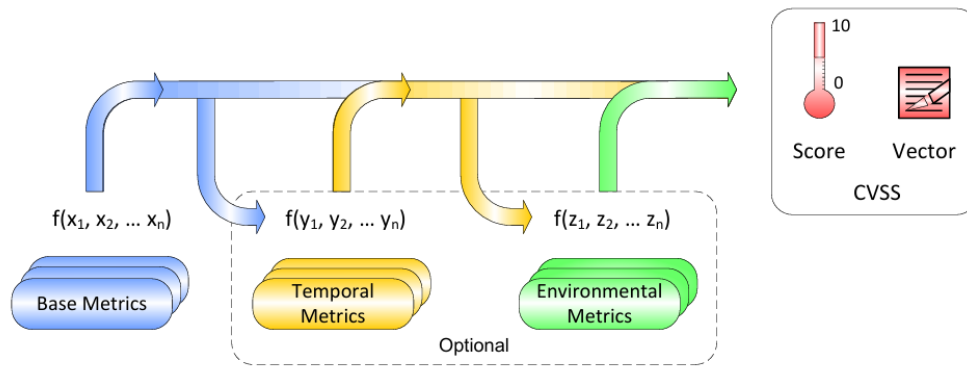
Common Vulnerability Scoring System (CVSS) adalah kerangka terbuka untuk mengkomunikasikan karakteristik dan dampak dari kerentanan (*vulnerability*) pada sistem teknologi informasi. CVSS terdiri dari tiga kelompok metrik utama, yaitu *Base Metrics*, *Temporal Metrics*, dan *Environment Metrics* [19].



Gambar 2. 5 Kelompok Metrik CVSS

Tujuan dari kelompok dasar CVSS adalah untuk mendefinisikan dan mengomunikasikan karakteristik fundamental suatu kerentanan. Pendekatan objektif untuk mengkarakterisasi kerentanan ini memberikan representasi kerentanan yang jelas dan intuitif kepada pengguna. Pengguna kemudian dapat menggunakan grup temporal dan lingkungan untuk memberikan informasi kontekstual yang lebih akurat mencerminkan risiko terhadap lingkungan unik mereka. Hal ini memungkinkan mereka untuk membuat keputusan yang lebih tepat ketika mencoba memitigasi risiko yang ditimbulkan oleh kerentanan.

CVSS bekerja dengan menetapkan nilai pada setiap metrik dasar. Nilai-nilai ini kemudian dihitung menggunakan rumus matematis untuk menghasilkan skor dari 0 hingga 10, beserta vektor (*string teks*) yang menjelaskan metrik yang digunakan [19].



Gambar 2. 6 Metrik dan Persamaan CVSS

Jika diperlukan, skor dasar (*Base Score*) dapat disesuaikan menggunakan metrik Temporal dan Environmental untuk memberikan konteks tambahan sesuai kondisi dunia nyata dan lingkungan organisasi. Setiap metrik dalam kelompok metrik Dasar CVSS Versi 2.0, baik dalam komponen Eksploitabilitas maupun Dampak, memiliki serangkaian nilai (*ratings*) diskrit yang telah ditetapkan. Nilai-nilai tersebut merupakan input kualitatif yang kemudian dikuantifikasi untuk digunakan dalam persamaan perhitungan skor. Penentuan nilai kualitatif ini harus didasarkan pada definisi dan kriteria yang ketat sebagaimana dijelaskan dalam standar CVSS [19]. Rincian nilai yang mungkin untuk setiap metrik adalah sebagai berikut:

Tabel 2. 2 *Metrics Value* CVSS Versi 2.0

<i>Metrics Group</i>	<i>Metrics Name</i>	<i>Metrics Value</i>	<i>Numerical Value</i>
<i>Exploitability Metrics</i>	<i>Access Vector (AV)</i>	<i>Local (L)</i>	0.395
		<i>Adjacent Network (A)</i>	0.646
		<i>Network (N)</i>	1
	<i>Access</i>	<i>High (H)</i>	0.35

	<i>Complexity</i> (AC)	<i>Medium (M)</i>	0.61
		<i>Low (L)</i>	0.71
	<i>Authentication</i> (Au)	<i>Multiple (M)</i>	0.45
		<i>Single (S)</i>	0.56
		<i>None (N)</i>	0.704
<i>Impacts Metrics</i>	<i>Confidentiality</i> <i>Impact (C)</i>	<i>None (N)</i>	0
		<i>Partial (P)</i>	0.275
		<i>Complete (C)</i>	0.66
	<i>Integrity</i> <i>Impact (I)</i>	<i>None (N)</i>	0
		<i>Partial (P)</i>	0.275

<i>Metrics Group</i>	<i>Metrics Name</i>	<i>Metrics Value</i>	<i>Numerical Value</i>
	<i>Availability</i> <i>Impact (A)</i>	<i>Complete (C)</i>	0.66
		<i>None (N)</i>	0
		<i>Partial (P)</i>	0.275
		<i>Complete (C)</i>	0.66

2.5.3.1.1 Base Metrics

Kelompok metrik dasar menangkap karakteristik kerentanan yang konstan seiring waktu dan di seluruh lingkungan pengguna. Metrik Vektor Akses, Kompleksitas Akses, dan Autentikasi menangkap bagaimana kerentanan diakses dan apakah kondisi tambahan diperlukan untuk mengeksploitasinya atau tidak. Ketiga metrik dampak mengukur bagaimana suatu kerentanan, jika dieksploitasi, akan secara langsung memengaruhi aset teknologi informasi, dengan dampak didefinisikan secara independen sebagai tingkat hilangnya *confidentiality*, *integrity*, dan *availability* [19]. *Base metrics* memiliki enam metrik utama, yaitu:

1. *Access Vector* (AV) menentukan seberapa jauh penyerang dapat berada untuk mengeksploitasi kerentanan.
2. *Access Complexity* (AC) mengukur tingkat kesulitan eksploitasi.
3. *Authentication* (Au) menilai jumlah autentikasi yang dibutuhkan sebelum eksploitasi berhasil.
4. *Confidentiality Impact* (C) dampak pada kerahasiaan data.
5. *Integrity Impact* (I) dampak pada keutuhan data.
6. *Availability Impact* (A) dampak pada ketersediaan layanan.

Dalam CVSS versi 2, skor dasar (*Base Score*) dihitung dari dua komponen utama, yaitu *Impact* dan *Exploitability*, dengan penyesuaian fungsi $f(Impact)$.

$$Base\ Score = round_1[(0.6 \times Impact) + (0.4 \times Exploitability) - 1.5] \times f(Impact)]$$

dengan

$$f(Impact) = \begin{cases} 0, & \text{jika } impact = 0 \\ 1.176, & \text{jika } impact > 0 \end{cases} \dots\dots\dots (2.1)$$

Keterangan:

Impact = menunjukkan besarnya dampak kerentanan terhadap sistem target.

Exploitability = menggambarkan seberapa mudah suatu kerentanan dapat dieksploitasi.

$f(Impact)$ = Untuk mengatur skala hasil akhir agar lebih proporsional terhadap dampak sebenarnya.

1.176 = Faktor pengali empiris yang menyeimbangkan bobot *Impact* dan *Exploitability*.

0.6 & 0.4 = Pembobotan nilai *Impact* dan *Exploitability*

1.5 = untuk menyesuaikan hasil agar distribusi skor *Base* lebih merata di rentang 0–10. Nilai tersebut diperoleh dari kalibrasi

terhadap ratusan contoh kerentanan oleh pengembang CVSS (*FIRST.org*).

Komponen skor Dasar (*Base Score*) dalam CVSS Versi 2.0 terdiri dari skor Eksploitabilitas (*Exploitability*) dan skor Dampak (*Impact*). Untuk menghitung skor Dampak, digunakan kombinasi dari tiga sub-metrik dampak. persamaan yang menggabungkan ketiga sub-metrik ini untuk menghasilkan skor Dampak adalah:

$$Impact = 10.41 \times (1 - (1 - C)(1 - I)(1 - A)) \dots\dots\dots (2.2)$$

Keterangan:

- 10.41 = Untuk menormalkan hasil agar skala Impact berada antara 0–10. Nilai ini ditentukan secara empiris oleh *FIRST.org* agar kombinasi maksimum ($C=I=A=1$) menghasilkan $Impact \approx 10$.
- C = *Confidentiality Impact*
- I = *Integrity Impact*
- A = *Availability Impact*
- 1 = Total Potensi Dampak (100%).

Penghitungan Skor Dampak (*Impact Sub-Score*) telah berhasil mengukur tingkat kerugian potensial yang mungkin terjadi pada aspek kerahasiaan, integritas, dan ketersediaan sistem. Namun, tingkat keparahan suatu kerentanan (*Base Score*) tidak hanya ditentukan oleh dampak kerugian, tetapi juga oleh Skor Eksploitabilitas (*Exploitability Sub-Score*) yang merepresentasikan kemudahan serangan, berikut ini merupakan persamaan skor eksploitabilitas:

$$Exploitability = 20 \times AV \times AC \times Au \dots\dots\dots (2.3)$$

Keterangan:

- 20 = untuk menyesuaikan skala *Exploitability* agar sebanding dengan *Impact*, sehingga nilai maksimum mendekati 10 bila metrik termudah dipilih. semakin mudah suatu serangan dilakukan, maka nilai *Exploitability* akan semakin besar.

AV = Menunjukkan seberapa jauh jarak serangan dapat dilakukan.

AC = Tingkat kesulitan serangan.

Au = Jumlah autentikasi yang dibutuhkan

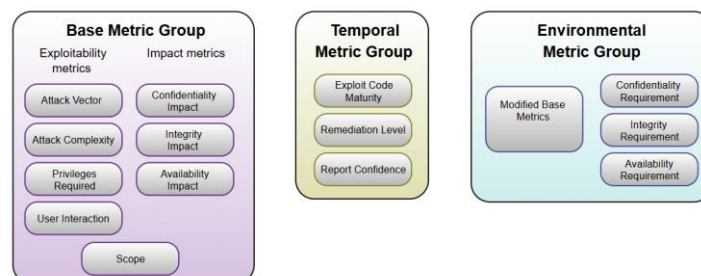
2.5.3.2 Common Vulnerability Scoring System (CVSS) Versi 3.1

Common Vulnerability Scoring System (CVSS) adalah standar terbuka (*open framework*) yang berfungsi untuk mengomunikasikan karakteristik dan tingkat keparahan kerentanan pada perangkat lunak, perangkat keras, dan *firmware* secara konsisten dan terukur. CVSS dikelola oleh [FIRST.Org](https://first.org), Inc. (*FIRST*), sebuah organisasi nirlaba yang menyediakan metodologi penilaian kerentanan yang terstandardisasi, netral *vendor*, dan netral *platform* [20].

Tujuan utama dari CVSS adalah menyediakan skor numerik yang menunjukkan tingkat keparahan kerentanan relatif terhadap kerentanan lainnya. Skor CVSS berkisar dari 0 hingga 10 dan juga direpresentasikan sebagai *vector string*, yaitu representasi tekstual terkompresi dari nilai-nilai yang digunakan untuk menurunkan skor.

2.5.3.3 Kelompok Metrik CVSS Versi 3.1

CVSS versi 3.1 terdiri dari tiga kelompok metrik utama, yaitu Metrik Dasar (*Base*), Metrik Temporal (*Temporal*), dan Metrik Lingkungan (*Environmental*) [20].



Gambar 2. 7 Kelompok Metrik CVSS

Metrik CVSS dirangkum sebagai berikut:

1. Metrik Dasar (*Base Group*) yang merepresentasikan kualitas intrinsik kerentanan yang konstan dari waktu ke waktu dan di seluruh lingkungan pengguna.
2. Metrik Temporal (*Temporal Group*) yaitu mencerminkan karakteristik kerentanan yang dapat berubah seiring waktu, seperti ketersediaan kode eksploitasi atau *patch*.
3. Metrik Lingkungan (*Environmental Group*) yaitu metrik yang merepresentasikan karakteristik kerentanan yang unik dan relevan untuk lingkungan pengguna tertentu, seperti adanya kontrol keamanan atau tingkat kepentingan aset.

Metrik Dasar menghitung skor awal (*Base Score*) yang mencerminkan tingkat keparahan kerentanan terburuk yang masuk akal. Metrik ini dibagi menjadi dua sub-grup yaitu Metrik Eksploitabilitas dan Metrik Dampak.

2.5.3.3.1 Metrik Eksploitabilitas

Metrik ini mengukur kemudahan dan cara teknis eksploitasi kerentanan terhadap komponen yang rentan (*vulnerable component*) [20]. Metrik ini meliputi:

1. *Attack Vector* (AV)

Nilai metrik ini akan semakin besar jika penyerang berada jauh (secara logis, dan fisik) untuk mengeksploitasi komponen yang rentan. Asumsinya adalah jumlah penyerang potensial untuk suatu kerentanan yang dapat dieksploitasi dari seluruh jaringan lebih besar daripada jumlah penyerang potensial yang dapat mengeksploitasi kerentanan yang memerlukan akses fisik ke perangkat, sehingga memerlukan Skor Dasar yang lebih tinggi.

2. *Attack Complexity* (AC)

Menjelaskan kondisi di luar kendali penyerang yang harus ada untuk mengeksploitasi kerentanan. Kondisi tersebut mungkin memerlukan

pengumpulan informasi lebih lanjut tentang target, atau pengecualian komputasi.

3. *Privileges Required* (PR)

Menggambarkan tingkat hak akses yang harus dimiliki penyerang sebelum berhasil mengeksploitasi kerentanan. Hak Istimewa yang Diperlukan biasanya Tidak Ada untuk kerentanan kredensial yang dikodekan secara keras atau kerentanan yang memerlukan rekayasa sosial

4. *User Interaction* (UI)

Metrik ini menentukan apakah kerentanan dapat dieksploitasi sepenuhnya atas keinginan penyerang, atau apakah pengguna lain (atau proses yang diinisiasi oleh pengguna) harus berpartisipasi dengan cara tertentu.

5. *Scope* (S)

Mengukur apakah kerentanan dalam satu komponen rentan memengaruhi sumber daya dalam komponen di luar cakupan keamanannya. Secara formal, otoritas keamanan adalah mekanisme yang mendefinisikan dan menegakkan kontrol akses dalam hal bagaimana subjek/aktor tertentu dapat mengakses objek/sumber daya tertentu yang dibatasi dengan cara yang terkendali. Jika kerentanan dalam komponen yang rentan dapat memengaruhi komponen yang berada dalam cakupan keamanan yang berbeda dari komponen yang rentan, maka komponen tersebut berada dalam cakupan keamanan yang berbeda dari komponen yang rentan.

2.5.3.3.2 Metrik Dampak

Metrik ini menangkap efek dari kerentanan yang berhasil dieksploitasi pada komponen yang mengalami hasil terburuk (impacted component) [20]. Metrik ini meliputi:

1. *Confidentiality* (C)

Metrik ini mengukur dampak terhadap kerahasiaan sumber daya informasi yang dikelola oleh suatu komponen perangkat lunak akibat kerentanan yang berhasil

dieksploitasi. Kerahasiaan mengacu pada pembatasan akses dan pengungkapan informasi hanya kepada pengguna yang berwenang, serta pencegahan akses atau pengungkapan kepada pengguna yang tidak berwenang.

2. *Integrity (I)*

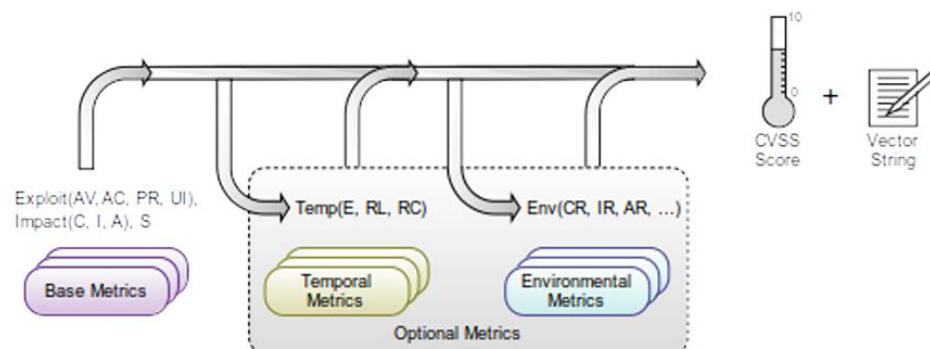
Metrik ini mengukur dampak terhadap integritas kerentanan yang berhasil dieksploitasi. Integritas mengacu pada kredibilitas dan kebenaran informasi.

3. *Availability (A)*

Metrik ini mengukur dampak terhadap ketersediaan komponen terdampak akibat kerentanan yang berhasil dieksploitasi. Sementara metrik dampak Kerahasiaan dan Integritas berlaku untuk hilangnya kerahasiaan atau integritas data yang digunakan oleh komponen terdampak, metrik ini mengacu pada hilangnya ketersediaan komponen terdampak itu sendiri, seperti layanan jaringan

2.5.3.4 Penilaian

Secara spesifik, persamaan dasar diturunkan dari dua subpersamaan, yaitu persamaan subskor eksploitasi, dan persamaan subskor dampak. Persamaan subskor eksploitasi diturunkan dari metrik eksploitasi dasar, sedangkan persamaan subskor Dampak diturunkan dari metrik Dampak Dasar.



Gambar 2. 8 Aliran Perhitungan Skor CVSS

Gambar 2.8 menunjukkan alur komponen dalam sistem penilaian kerentanan CVSS. Skor Dasar kemudian dapat disempurnakan dengan memberi skor pada metrik Temporal dan Lingkungan agar lebih akurat mencerminkan tingkat keparahan relatif yang ditimbulkan oleh kerentanan terhadap lingkungan pengguna pada titik waktu tertentu. Memberi skor pada metrik Temporal dan Lingkungan tidak diwajibkan, tetapi disarankan untuk skor yang lebih presisi [20].

Menilai metrik CVSS juga menghasilkan string vektor, representasi tekstual dari nilai metrik yang digunakan untuk menilai kerentanan. *String* vektor ini adalah string teks berformat khusus yang berisi setiap nilai yang ditetapkan untuk setiap metrik, dan harus selalu ditampilkan bersama skor kerentanan.

2.5.3.5 Skala Penilaian Keparahan Kualitatif

Skor numerik yang dihasilkan dari perhitungan Metrik Dasar, Temporal, dan Lingkungan dikategorikan ke dalam skala peringkat kualitatif untuk memudahkan prioritas dan manajemen risiko seperti pada Tabel 2.3 yang merupakan skala penilaian keparahan kualitatif [20]. Hasil akhir perhitungan CVSS Score merupakan indikator penting dalam proses pengambilan keputusan mitigasi kerentanan.

Tabel 2. 3 Skala Penilaian Keparahan Kualitatif

<i>Rating</i>	<i>CVSS Score</i>
<i>None</i>	0,0
<i>Low</i>	0,1 – 3,9
<i>Medium</i>	4,0 – 6,9
<i>High</i>	7,0 – 8,9
<i>Critical</i>	9,0 – 10,0

2.5.3.6 *Vector String*

Vector String adalah representasi tekstual dari semua nilai metrik CVSS yang digunakan untuk memperoleh skor. *Vector string* dimulai dengan label versi ("CVSS:3.1/") diikuti oleh serangkaian pasangan metrik/nilai yang dipisahkan oleh garis miring ("/"). *Vector String* adalah cara baku untuk mengomunikasikan penilaian kerentanan, menjamin konsistensi di antara platform dan pengguna [20].

2.5.3.7 *Metric Values*

Setiap metrik pada *Common Vulnerability Scoring System* (CVSS) versi 3.1 memiliki nilai numerik tertentu yang digunakan dalam proses perhitungan skor kerentanan. Nilai-nilai ini disebut *metric values*. Tujuan dari pemberian nilai numerik pada setiap metrik adalah untuk mengubah parameter kualitatif (seperti *Low*, *High*, atau *Network*) menjadi bentuk kuantitatif yang dapat dihitung secara matematis dalam rumus CVSS [20].

Secara umum, metrik CVSS dibagi menjadi beberapa kelompok utama, yaitu *Base Metrics*, *Temporal Metrics*, dan *Environmental Metrics*. Setiap metrik dalam kelompok ini memiliki beberapa opsi nilai yang menggambarkan kondisi atau karakteristik dari kerentanan yang dianalisis. Tabel 2.4 menunjukkan nilai numerik yang digunakan pada setiap metrik dalam perhitungan CVSS v3.1.

Tabel 2. 4 *Metric Value*

<i>Metric</i>	<i>Metric Value</i>	<i>Numerical Value</i>
<i>Attack Vector / Modified Attack Vector</i>	<i>Network</i>	0.85
	<i>Adjacent</i>	0.62
	<i>Local</i>	0.55
	<i>Physical</i>	0.2
<i>Attack Complexity /</i>	<i>Low</i>	0.77

<i>Modified Attack Complexity</i>	<i>High</i>	0.44
<i>Privileges Required / Modified Privileges Required</i>	<i>None</i>	0.85
	<i>Low</i>	0.62 (or 0.68 if Scope / Modified Scope is Changed)
	<i>High</i>	0.27 (or 0.5 if Scope / Modified Scope is Changed)
<i>User Interaction / Modified User Interaction</i>	<i>None</i>	0.85
	<i>Required</i>	0.62
<i>Confidentiality / Integrity / Availability / Modified Confidentiality / Modified Integrity / Modified Availability</i>	<i>High</i>	0.56
	<i>Low</i>	0.22
	<i>None</i>	0
	<i>Not Defined</i>	1
	<i>High</i>	1
<i>Exploit Code Maturity</i>	<i>Functional</i>	0.97
	<i>Proof of Concept</i>	0.94
	<i>Unproven</i>	0.91

<i>Metric</i>	<i>Metric Value</i>	<i>Numerical Value</i>
<i>Report Confidence Confidentiality Requirement / Integrity Requirement / Availability Requirement</i>	<i>Not Defined</i>	1
	<i>Confirmed</i>	1
	<i>Reasonable</i>	0.96
	<i>Unknown</i>	0.92
	<i>Not Defined</i>	1
	<i>High</i>	1.5
	<i>Medium</i>	1
	<i>Low</i>	0.5

2.5.3.8 Persamaan CVSS v3.1

Persamaan CVSS v3.1 dibagi menjadi tiga kerangka utama yang bersifat hierarkis, yaitu *Base Score*, *Temporal Score*, dan *Environmental Score* [20]. Setiap kerangka memiliki komponen uniknya sendiri.

2.5.3.8.1 Persamaan Metrik Dasar (*Base Score*)

Persamaan ini merupakan inti dari CVSS, yang menghasilkan skor intrinsik kerentanan. Komponen utamanya adalah *Impact Sub-Score* (ISS) dan *Exploitability Sub-Score* (ESS). *Impact Sub-Score* menghitung tingkat kerusakan yang disebabkan oleh kerentanan yang berhasil dieksploitasi terhadap tiga aspek keamanan, yaitu Kerahasiaan, Integritas, dan Ketersediaan[20] dengan menggunakan rumus:

$$ISS = 1 - (1 - C) \times (1 - I) \times (1 - A) \dots\dots\dots (2.4)$$

Keterangan:

ISS = Skor Dampak Bersih. Probabilitas adanya dampak pada setidaknya satu metrik C, I, atau A.

C = *Confidentiality Impact*

I = *Integrity Impact*

A = *Availability Impact*

1 = Total Potensi Dampak (100%).

Selanjutnya, nilai *Impact* ditentukan berdasarkan kondisi *Scope* (S) dari kerentanan. *Scope* menunjukkan apakah eksploitasi suatu komponen dapat mempengaruhi sumber daya di luar komponen tersebut [20].

1. *Unchanged Scope* (S:U)

Apabila *Scope* = *Unchanged* (S:U), maka kerentanan hanya berdampak pada komponen yang sama, dan *Impact* dihitung dengan rumus:

$$Impact = 6.42 \times ISS \dots\dots\dots (2.5)$$

2. *Changed Scope* (S:C)

Sebaliknya, apabila *Scope* = *Changed* (S:C), maka kerentanan dapat mempengaruhi komponen lain di luar batas otoritas awal, sehingga perhitungannya menjadi lebih kompleks, yaitu:

$$Impact = 7.52 \times (ISS - 0.029) - 3.25 \times (ISS - 0.02)^{15} \dots\dots\dots (2.6)$$

Keterangan:

<i>Impact</i>	= Skor dampak keseluruhan
ISS	= Nilai <i>Impact Sub-Score</i>
6.42	= Diperoleh dari model empiris CVSS untuk mengubah ISS menjadi skala 0–10 agar sebanding dengan nilai <i>Exploitability</i> .
7.52	= Konstanta pengali utama untuk menyesuaikan skala dampak
0.029 & 0.02	= Konstanta penggeser untuk menstabilkan kurva pada nilai rendah
3.25	= Konstanta koreksi untuk menjaga proporsi dampak maksimum

ESS (*Exploitability Sub-Score*) menghitung kemudahan teknis eksploitasi kerentanan berdasarkan empat metrik akses, yaitu *Attack Vector* (AV), *Attack Complexity* (AC), *Privileges Required* (PR), dan *User Interaction* (UI) [20] menggunakan persamaan berikut:

$$ESS = 8.22 \times AV \times AC \times PR \times UI \dots\dots\dots (2.7)$$

Keterangan:

ESS = Skor Eksploitabilitas.

AV = *Attack Vector*

AC = *Attack Complexity*

PR = *Privileges Required*

UI = *User Interaction*

8.22 = Faktor Penskalaan. Konstanta yang digunakan untuk mengatur agar ESS (dan BS) berada dalam rentang skor 0-10.

BS adalah *Base Score* yang dihitung dengan menggabungkan ISS dan ESS, serta menerapkan faktor *Scope* (S) jika ada perubahan lingkup (*Changed Scope*). Jika nilai *Impact* bernilai nol (tidak ada dampak), maka *Base Score* otomatis bernilai nol [20]. Namun apabila *Impact* lebih besar dari nol, perhitungannya dibedakan berdasarkan nilai *Scope* (S).

1. *Unchanged Scope* (S:U):

$$BS = \text{Roundup} (\text{Minimum} [(ISS + ESS), 10]) \dots\dots\dots (2.8)$$

2. *Changed Scope* (S:C):

$$BS = \text{Roundup} (\text{Minimum} [1.08 \times (ISS + ESS), 10]) \dots\dots\dots (2.9)$$

Keterangan:

BS	=	Base Score
Roundup	=	Fungsi Pembulatan ke Atas. Membulatkan skor ke satu desimal tertinggi terdekat (misalnya, 7.11 menjadi 7.2) untuk memastikan bahwa setiap perubahan metrik sekecil apapun selalu tercermin dalam skor akhir.
Minimum [(...),10	=	Batas Atas Skor. Memastikan skor numerik <i>Base</i> tidak melebihi nilai maksimum 10.0.
ISS	=	Skor Dampak bersih
ESS	=	Skor eksploitabilitas
1.08	=	Faktor Peningkatan <i>Scope</i> . Digunakan hanya jika S:C. Ini menaikkan skor karena kerentanan memiliki potensi dampak yang lebih luas, di luar komponen utama yang rentan.

2.6 *Penetration Testing*

Penetration Testing adalah cara proaktif untuk mengidentifikasi kerentanan dalam aset digital dengan secara aktif mencari kerentanan dan mengeksploitasinya dari sudut pandang penyerang. Untuk mencapai tujuan keamanan dunia maya, yaitu *Integrity*, *Availability* dan *Confidentiality*, *penetration testing* telah menjadi elemen wajib. Saat ini, ada beragam opsi untuk *penetration testing*. Ada berbagai sistem dengan alat yang melakukan *penetration testing* termasuk Kali Linux. Melalui tes ini, penguji dapat mengidentifikasi kerentanan secara terorganisir dan terkendali. Dengan demikian,

mereka membuat laporan tentang masalah yang memerlukan perbaikan sistem dan menambal kerentanan keamanan kepada manajemen. Ini dianggap sebagai penilaian risiko dan dapat digunakan untuk memverifikasi keamanan jaringan. Dengan demikian, tujuan *penetration testing* jaringan adalah untuk melindungi data dan memastikan keamanan secara keseluruhan, terutama saat mengelola data penting. Contohnya termasuk *SQL Injection*, konfigurasi *firewall* yang tidak memadai, dan *virus* atau *malware* tradisional [21].

2.7 Security Information and Event Management (SIEM)

SIEM adalah sistem yang dapat mengelola *log* serangan siber yang dihasilkan oleh beberapa sumber data. Biasanya, SIEM mendapatkan beberapa sumber data serangan siber dari IDS. SIEM juga dapat digunakan sebagai *live monitoring* aliran trafik jaringan secara *realtime* dari beberapa IDS. IDS bertindak seperti sensor untuk menangkap arus lalu lintas jaringan dan mendeteksi anomali pada jaringan. Pada dasarnya, IDS hanya mendeteksi serangan siber di jaringan menggunakan metode *rule-based* dan memberikan peringatan kepada tim TI atau keamanan siber di perusahaan [22].

2.7.1 Intrusion Detection System (IDS)

Intrusion Detection System adalah entitas untuk mengaudit sistem dan operasi jaringan terhadap tindakan pelanggaran kebijakan. IDS dapat dibagi menjadi beberapa kategori menggunakan beberapa pendekatan. Dua jenis pertama adalah: berbasis jaringan dan berbasis *host*, tergantung di mana perilaku serangan yang diamati. IDS berbasis jaringan memantau dan menganalisis lalu lintas jaringan dan berfokus pada keamanan jaringan. IDS berbasis *host* mengidentifikasi aktivitas berbahaya dengan memantau

proses dan kejadian sistem di lingkungan perangkat lunak yang terkait dengan komputer tertentu [10]. Ada dua jenis utama IDS, yaitu:

1. *Network-based* IDS (NIDS) memiliki fungsi untuk memantau lalu lintas jaringan yang mengalir melalui segmen jaringan atau perangkat seperti switch atau router. Ini memeriksa header dan isi lalu lintas untuk mencari tanda-tanda aktivitas yang mencurigakan atau tanda-tanda serangan.
2. *Host-based* IDS (HIDS), berbeda dengan NIDS, HIDS berfokus pada sistem individu. Ini memantau aktivitas pada perangkat tuan rumah, termasuk perubahan file, perubahan konfigurasi, percobaan login yang mencurigakan, dan aktivitas lainnya yang tidak biasa.

IDS bekerja dengan melakukan pengumpulan data dari sumber seperti *log* sistem, lalu lintas jaringan atau aktivitas sistem. Data yang telah dikumpulkan akan dianalisis oleh IDS dengan membandingkannya dengan aturan, tanda-tanda, atau pola yang telah ditentukan sebelumnya. Ini dapat mencakup aturan *signature* yang mendefinisikan serangan yang diketahui, serta analisis perilaku untuk mendeteksi aktivitas yang tidak biasa. Jika IDS menemukan aktivitas yang cocok dengan aturan atau pola yang mencurigakan, itu menghasilkan peringatan. Peringatan yang dihasilkan dapat berupa notifikasi kepada administrator, penyimpanan dalam *log*, atau tindakan respons lainnya, seperti memblokir alamat IP yang mencurigakan [10].

Namun, penting untuk diingat bahwa IDS mungkin juga menghasilkan kesalahan palsu (*false positives*) atau gagal mendeteksi serangan yang tidak diketahui (*false negatives*). Oleh karena itu, penggunaan IDS biasanya diintegrasikan dengan alat keamanan lainnya, seperti *Intrusion Prevention Systems* (IPS) [10].

2.7.2 Intrusion Prevention System (IPS)

Intrusion Prevention System (IPS) merupakan aplikasi keamanan jaringan yang dipasang di situs *web*, dan memberikan solusi untuk melindungi aset informasi dari

serangan dunia maya. *Signature* adalah salah satu pengaturan kunci untuk IPS. ini diwakili oleh ekspresi reguler dan digunakan untuk memeriksa input pengguna untuk mendeteksi serangan. Input pengguna yang terdeteksi oleh *signature* diproses berdasarkan mode operasi IPS. Secara umum, IPS lebih sering dipakai dibandingkan dengan IDS karena IPS digunakan untuk memblokir serangan dan bukan hanya melakukan *monitoring* saja [12].

IPS mengkombinasikan teknik *firewall* dan metode *Intrusion Detection System* (IDS) dengan sangat baik. IPS menggunakan *signatures* untuk mendeteksi aktivitas *traffic* di jaringan dan terminal, dimana pendeteksian paket yang masuk dan keluar (*inbound-outbound*) dapat dicegah sedini mungkin sebelum merusak atau mendapatkan akses ke dalam jaringan lokal. Jadi *early detection* dan *prevention* menjadi penekanan pada IPS ini[23]. Ada dua jenis IPS, yaitu:

1. *Network-based IPS* (NIPS), yang ditempatkan di antara lalu lintas jaringan dan infrastruktur jaringan internal. Ini memonitor lalu lintas secara kontinu dan dapat mengambil tindakan untuk memblokir atau menolak lalu lintas yang diidentifikasi sebagai ancaman.
2. *Host-based IPS* (HIPS) beroperasi pada tingkat perangkat atau sistem individu. Ini memantau aktivitas pada perangkat tuan rumah dan dapat mengambil tindakan untuk mencegah serangan yang ditujukan pada perangkat tersebut.

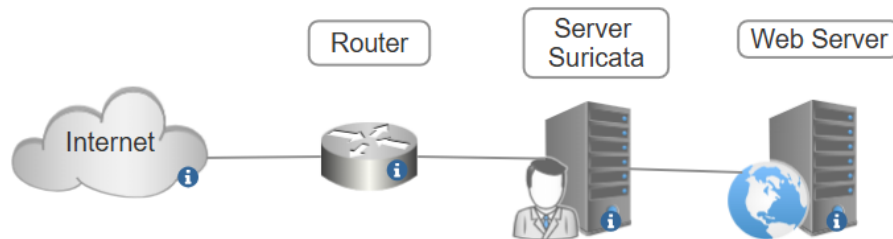
2.7.3 Suricata

Suricata adalah teknologi pendeteksi serangan dan pencegahan serangan yang bersifat *open source* dan memiliki banyak pengguna. Suricata dapat dikonfigurasi sebagai IDS atau IPS, yang dibuat dan dikembangkan oleh komunitas yang luas di bawah bimbingan *Open Information Security Foundation* (OISF). Proyek ini dimulai pada tahun 2009, dan dirilis secara resmi pertama kali pada tahun 2010. Tujuan awal dari proyek Suricata IDS adalah untuk mengembangkan mesin deteksi intrusi *based on*

signatures. Suricata menggunakan teknologi multi-threading, dukungan HTTP tingkat lanjut, dan pengenalan protokol port-independen.

Suricata melakukan pelacakan *flow packet* (aliran paket) pada sebagian besar protokol TCP/IP. Dalam kasus TCP, ini adalah pemetaan langsung *flow packet* ke sesi TCP. Untuk UDP, hal ini diselesaikan dengan melihat informasi IP (IP sumber, IP port dan tujuan, serta port) dan menerapkan logika batas waktu [24]. Jadi, Suricata melacak apa yang terjadi pada *flow packet* saat terjadi komunikasi antara klien dengan server. Semua IP *event* berisi kunci '*flow_id*' yang sama untuk semua *event* dalam satu *flow packet*. Hal ini memungkinkan *user* untuk mengelompokkan semua *event*.

Biasanya Suricata diletakkan di depan lingkungan server yang berjalan. Ini bertujuan untuk melakukan deteksi dan prevensi terhadap lalu lintas internet yang ingin masuk ke lingkungan server utama seperti *webserver*.



Gambar 2. 9 Topologi Dasar Sistem Suricata

Dengan topologi seperti Gambar 2.9 Suricata mampu untuk melakukan deteksi dan *filtering* paket yang akan menuju ke server *webserver*. Hal ini dapat memungkinkan juga untuk memblokir anomali yang mencoba untuk masuk ke dalam server. Setiap kali Suricata mendeteksi *user agent* (agen pengguna) HTTP yang belum pernah dilihat di jaringan oleh Suricata, itu akan memicu peringatan. Peristiwa ini dapat digunakan

untuk membuat daftar item yang sebelumnya tidak terlihat untuk semua *fields* yang dapat dicocokkan dengan *sticky buffer*.

2.7.3.1 Sistem Kerja Suricata

Sistem kerja Suricata adalah dengan mengekstrak dan menghasilkan log transaksi protokol secara independent dari lalu lintas jaringan. Hasilnya, *network defender* memiliki tanggung jawab untuk menemukan jenis peristiwa yang paling sesuai untuk dicari hasilnya berdasarkan data. Terdapat dua metode pada sistem kerja Suricata, yaitu mode IDS dan mode IPS. Pada mode IDS, Suricata hanya menduplikasi lalu lintas jaringan tanpa mempengaruhi koneksi untuk menangkap dan menguraikan paket jaringan yang nantinya akan dianalisis apakah paket tersebut normal atau tidak. Mode IPS Suricata beroperasi secara *inline*, yang berarti sistem ini berada di jalur lalu lintas jaringan, memungkinkan untuk memproses dan memblokir paket berbahaya sebelum mencapai tujuan yang dilindungi [25].

Suricata yang dijalankan dalam mode IPS (*Intrusion Prevention System*) dapat bekerja dengan memanfaatkan modul *NFQUEUE* pada kernel Linux. Mekanisme kerja tersebut dapat dijelaskan melalui alur sebagai berikut:

1. Paket yang dikirim melalui jaringan pertama kali diterima oleh *Network Interface Card* (NIC), kemudian diproses oleh *network stack* kernel Linux. Kernel menggunakan mekanisme *Netfilter hooks* untuk memutuskan jalur pemrosesan paket, seperti melalui rantai *PREROUTING*, *INPUT*, *FORWARD*, dan *OUTPUT*.
2. Pada tahap selanjutnya, aturan *iptables* yang telah dikonfigurasi akan mengarahkan paket menuju antrean *NFQUEUE*. Paket yang dialihkan akan ditahan sementara di kernel hingga mendapatkan *verdict* dari *userspace*.
3. Antrean *NFQUEUE* memungkinkan aplikasi *userspace* dalam penelitian ini yaitu Suricata untuk menerima paket melalui *library libnetfilter_queue*.

Suricata kemudian membaca setiap paket dari antrean dan melakukan proses analisis.

4. Suricata melakukan proses *decoding* terhadap protokol jaringan (*Ethernet*, *IP*, *TCP/UDP*), normalisasi, serta *fragment reassembly*. Tujuan tahapan ini adalah agar paket berada dalam format standar sebelum dilakukan analisis lebih lanjut. Tahap ini disebut *Preprocessing Paket*.
5. Pada tahap *flow tracking*, Suricata membangun identitas *flow* untuk setiap koneksi jaringan (misalnya berdasarkan 5 *tuple*: *IP* sumber, *IP* tujuan, *port* sumber, *port* tujuan, dan protokol). Selain itu, dilakukan *TCP stream reassembly* agar *payload* aplikasi yang terbagi ke dalam beberapa segmen *TCP* dapat dianalisis secara utuh.
6. Setelah aliran paket direkonstruksi, Suricata mengekstrak informasi protokol tingkat aplikasi, seperti *URI*, *header*, dan *body* pada *HTTP*; *Server Name Indication* (*SNI*) pada *TLS*, serta *query* pada *DNS*. Informasi ini digunakan untuk mencocokkan aturan (*rules*) yang berbasis konten aplikasi.
7. Suricata kemudian membandingkan paket dengan aturan (*ruleset*) yang telah dikonfigurasi. Aturan dapat berupa *signature-based detection*, *flowbits*, maupun penggunaan *threshold* untuk mendeteksi serangan berulang. Jika sebuah paket sesuai dengan kondisi pada aturan, maka Suricata mengeksekusi aksi atau perintah yang didefinisikan.
8. Setelah keputusan diambil, Suricata mengirimkan *verdict* ke kernel melalui fungsi `nfq_set_verdict()`. Kernel kemudian melanjutkan pemrosesan paket sesuai keputusan tersebut, misalnya meneruskan paket (*ACCEPT*) atau membuangnya (*DROP/REJECT*).
9. Pada tahap *Post-Processing* Suricata menyimpan hasil deteksi ke dalam *file log* *eve.json*, memperbarui statistik, serta dapat menyimpan *pcap* untuk analisis forensik.

2.7.3.1.1 Mode IDS

Mode IDS pada Suricata menggunakan alat dalam sistem operasi linux untuk menangkap paket jaringan dan memeriksa panjang, isi, dan informasi *header*-nya. Libpcap (*Library Packet Capture*) yang merupakan *library portable* untuk membaca dan menangkap paket yang melewati lalu lintas jaringan yang nantinya akan difilter dan diklasifikasikan. Setelah paket ditangkap, Suricata akan menguraikan paket sesuai dengan lapisan OSI (*layer OSI*).



Gambar 2. 10 *OSI Layer*

Libpcap akan menangkap paket mentah (*raw*) yang melewati lingkungan jaringan server yang kemudian akan diuraikan oleh Suricata sesuai dengan lapisan OSI (*OSI layer*). *Raw packet* (paket mentah) merupakan paket yang masih berupa sekumpulan *byte*. Setelah paket mentah berhasil di tangkap, Suricata akan menguraikan paket atau biasa yang disebut dengan *decoding packet*. Proses ini menguraikan paket mentah yang masih berupa bilangan *hexadecimal* menjadi informasi paket tersebut sesuai dengan *OSI layer*. Contoh paket mentah yang ditangkap oleh Libpcap sebagai berikut:

```

0000 52 54 00 12 35 02 08 00 27 1b 4e c4 08 00 45 00
0010 00 3c 1c 46 40 00 40 06 b1 e6 c0 a8 01 68 c0 a8
0020 01 01 d1 f5 00 50 a5 6d 5a 4d 00 00 00 00 70 18
0030 fa f0 1d 4c 00 00 48 45 4c 4c 4f 20 57 4f 52 4c
0040 44 21 0a

```

Gambar 2. 11 Contoh *Raw Packet*

Sesuai dengan OSI *layer*, baris pertama pada paket mentah di atas merupakan *header* paket *ethernet* yang berada pada *layer* kedua yaitu *layer Data Link*.

- 52 54 00 12 35 02 merupakan *MAC address* pengirim
- 08 00 merupakan tipe protokol *ethernet* yang digunakan yaitu IPv4
- 27 1b 4e c4 merupakan *MAC Address* penerima

Pada baris kedua berada di *layer Network* pada OSI *layer*, yang bertugas untuk mengatur alamat IP, fragmentasi, dan *routing* paket antar jaringan.

- 00 3c 1c 46 merupakan *ID* unik untuk fragmentasi paket
- 40 00 merupakan *flags* dan *fragment offset* yang digunakan untuk menentukan paket tersebut terfragmentasi atau tidak. Pada kasus ini 40 menunjukkan bahwa *flag* “*Don’t Fragment*” tidak diaktifkan, dan 00 menunjukkan tidak ada *offset* fragmentasi karena paket ini tidak terfragmentasi.
- 40 menunjukkan TTL (*Time to Live*) Menunjukkan berapa *hop* (lompatan) paket dapat dilakukan sebelum dibuang. Pada kasus ini paket dapat melompati 64 perangkat jaringan sebelum dihapus.
- 06 menunjukkan jenis protokol yang dipakai, pada kasus ini protokol menggunakan protokol TCP.
- b1 e6 menunjukkan *Checksum header IP* yang digunakan untuk verifikasi integritas *header IP*.
- c0 a8 01 68 menunjukkan alamat pengirim menggunakan IPv4, pada kasus ini *ID* c0 a8 01 68 diterjemahkan menjadi 192.168.1.104 yang merupakan IP pengirim.

- g. c0 a8 01 01 menunjukkan alamat penerima menggunakan IPv4, pada kasus ini *ID* c0 a8 01 01 diterjemahkan menjadi 192.168.1.1 yang merupakan IP penerima.

Baris ketiga merupakan paket yang berupa *header* TCP yang berada pada *layer* OSI lapisan ke empat yaitu *Transport layer*, yang bertanggung jawab untuk pengiriman data secara reliabel.

- a. d1 f5 merupakan *port* pengirim yang jika di *decimal*-kan menjadi 53749, yang berarti pengirim mengirim paket melalui *port* 53749.
- b. 00 50 merupakan *port* penerima yang dituju oleh pengirim, pada kasus ini *port* penerima yaitu *port* 80 yang merupakan *port* untuk HTTP.
- c. a5 6d 5a 4d merupakan nomor urut atau *sequence number* dalam protokol TCP untuk mengindikasikan urutan pengiriman data.
- d. 00 00 00 00 *acknowledgment number* merupakan nomor urut paket yang diterima. Pada kasus ini kosong karena paket ini merupakan paket SYN atau paket awal untuk menyambungkan koneksi.
- e. 70 18 merupakan *flags* dan *window size* yang menunjukkan bahwa ini adalah paket dengan *flag* ACK + PSH yang artinya segmen ini digunakan untuk memberikan tanda terima (*acknowledgment*) atas data yang telah diterima dan data yang terkandung dalam segmen ini segera diteruskan ke aplikasi penerima.
- f. fa f0 merupakan *checksum* untuk memverifikasi integritas dari paket tersebut.

Baris keempat merupakan *layer* aplikasi pada *layer* OSI yang bertugas untuk membaca *payload* atau isi dari paket yang diterima. 48 45 4c 4c 4f 20 57 4f 52 4c 44 21 0a merupakan *payload* dalam bentuk teks *hexadecimal* yang jika diterjemahkan ke dalam bilangan *decimal* sesuai dengan teks ASCII (*American Standard Code for Information Interchange*) yang merupakan standar pengkodean karakter yang digunakan untuk merepresentasikan teks dalam komputer, menjadi HELLO WORLD!, di mana:

- a. 48 45 4c 4c 4f bilangan *hexadecimal* dari HELLO.
- b. 20 terjemahan sebagai spasi.

- c. 57 4f 52 4c 44 terjemahan WORLD.
- d. 21 terjemahan dari tanda seru (!).
- e. 0a merupakan terjemahan untuk baris baru.

Setelah menguraikan paket yang diterima, suricata membuat *output log* dalam bentuk *json* yang berisikan hasil uraian dari paket mentah yang ditangkap.

Suricata memiliki beberapa metode deteksi, salah satunya adalah *Signature-based detection*. Teknik ini digunakan untuk mendeteksi intrusi berbahaya dari luar lingkungan jaringan komputer dengan cara melakukan pencocokan *signature* (pola). Suricata dirancang untuk mendeteksi paket data ataupun *flow packet* yang masuk ke lingkungan jaringan komputer dan akan diperiksa terhadap kumpulan pola/aturan yang diatur oleh Suricata, jika paket atau *flow packet* data cocok dengan salah satu *signature*, itu dianggap sebagai ancaman atau aktivitas mencurigakan. Suricata sangat mendukung untuk pendeteksian HTTP. Objek HTTP berisi semua informasi tentang permintaan dan respons. *Fields* seperti *hostname* atau *http_user_agent* berasal dari klien dan *fields* seperti *status*, *length*, atau *http_content_type* berasal dari server. Log juga menyertakan *tx_id* yang merupakan singkatan dari pengenalan transaksi. Ini memberikan jumlah transaksi HTTP (permintaan + *respons*) yang terlihat pada *flow* pada saat permintaan.

```

{
  "timestamp": "2024-11-20T12:00:00",
  "ethernet": {
    "destination_mac": "52:54:00:12:35:02",
    "source_mac": "08:00:27:1b:4e:c4",
    "ethertype": "0800"
  },
  "ip": {
    "version": 4,
    "header_length": 20,
    "total_length": 60,
    "identification": "1c46",
    "flags": "4000",
    "ttl": 64,
    "protocol": "TCP",
    "source_ip": "192.168.1.104",
    "destination_ip": "192.168.1.1"
  },
  "tcp": {
    "source_port": 53749,
    "destination_port": 80,
    "sequence_number": 2776356941,
    "acknowledgment_number": 0,
    "flags": "SYN, ACK",
    "window_size": 64240,
    "checksum": "1d4c"
  },
  "payload": {
    "data": "HELLO WORLD!",
    "length": 12
  }
}

```

Gambar 2. 12 Paket JSON yang Sudah Diuraikan

Suricata menganalisis berbasis *flow* untuk membangun *stateful connection tracking*, di mana semua koneksi dicatat dan disimpan berdasarkan alamat pengirim, *port* pengirim dan protokol yang digunakan. Dengan *flow* ini suricata dapat mengenali pola koneksi yang mencurigakan, seperti *port scanning* atau DDoS.

Sistem *monitoring* Suricata menggunakan Metode *signature-based detection*, di mana Suricata menggunakan *rules* (aturan) yang berisi *signature* (pola) dari berbagai jenis serangan atau aktivitas yang mencurigakan. Aturan yang digunakan dalam fitur *rules* yang dikonfigurasi pada Suricata yaitu:

- a. *Alert*
- b. *Drop*
- c. *Pass*
- d. *Reject*

2.7.3.1.1.1 *Rules Alert*

Alert merupakan perintah *rules* pada Suricata, digunakan untuk mencatat kejadian yang mencurigakan atau serangan yang terdeteksi tanpa menghentikan lalu lintas. Ini adalah tipe aturan yang paling umum dalam IDS. Berikut ini merupakan contoh dari konfigurasi *rules alert*.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP
Ping Detected";icode:0;itype:8;sid:1000005;rev:1;)
```

Gambar 2. 13 Contoh *Rules Alert* Suricata untuk Paket ICMP

Keterangan:

- | | | |
|---------------------------|---|--|
| <i>alert</i> | : | Perintah aturan pada Suricata untuk mencatat aktivitas pada lalu lintas yang mencurigakan. |
| <i>icmp</i> | : | Protokol yang dideteksi oleh Suricata. |
| <i>\$HOME_NET any</i> | : | Mendefinisikan sebagai lingkungan jaringan internal. |
| <i>\$EXTERNAL_NET any</i> | : | Mendefinisikan semua jaringan yang bukan bagian dari <i>\$HOME_NET</i> atau lingkup jaringan internal. |
| <i>msg</i> | : | Merupakan pesan yang akan ditampilkan di log Suricata. |
| <i>icode:0</i> | : | Merupakan variabel untuk memberikan informasi lebih lanjut atau detail spesifik tentang pesan ICMP dalam konteks jenis pesan ICMP yang telah ditentukan oleh <i>itype</i> . <i>Icode:0</i> merupakan kode ICMP untuk <i>echo request</i> (ping). |
| <i>itype:8</i> | : | Mengidentifikasi jenis pesan ICMP. Jenis pesan ini menunjukkan fungsi atau tujuan dari pesan ICMP |

tersebut. *Itype:8* menunjukkan ini merupakan pesan *echo request* (ping).

sid:1000005 : ini merupakan ID unik untuk *rules*. *sid* (*Signature ID*) digunakan untuk mengidentifikasi *rules* secara unik dalam kumpulan *rules* Suricata.

rev:1 : Nomor revisi untuk sebuah *rules*. *rev* digunakan untuk melacak versi dari *rules* tersebut, sehingga dapat mengetahui jika ada pembaruan atau perubahan pada *rules*.

Maksud dari *rules* tersebut, suricata akan memberikan pesan *alert* jika ada paket yang menggunakan protokol ICMP dari jaringan *external* yang menuju ke lingkungan jaringan *internal server* dengan mencocokkan tipe dan kode ICMP yang masuk ke jaringan *internal server*.

2.7.3.1.1.2 Rules Drop

Drop merupakan perintah *rules* Suricata yang digunakan untuk memblokir lalu lintas jaringan yang mencurigakan berdasarkan aturan yang telah ditentukan. Pada sistem ini, aturan dengan kata kunci *drop* tidak hanya mencatat aktivitas yang mencurigakan tetapi juga secara aktif mencegah dengan memblokir lalu lintas tersebut mencapai server tujuan. Berikut ini merupakan contoh struktur dari *rules drop* pada Suricata.

```
drop tcp any any -> 192.168.1.10 80
(content:"/malicious.php";http_uri;msg:"Malicious HTTP
request blocked";sid:1001;rev:1;)
```

Gambar 2. 14 Contoh *Rules Drop* Suricata

Rules drop memiliki sistem kerja seperti perintah *alert* dengan menangkap paket mentah pada lalu lintas jaringan dan melakukan pencocokan *signature* (pola) yang digunakan

oleh pengirim. Jika pola yang digunakan cocok dengan *rules drop* yang dikonfigurasi, Suricata akan menjatuhkan paket tersebut agar tidak masuk ke *server* yang di tuju. Pada *rules* diatas memungkinkan Suricata untuk menjatuhkan paket yang memiliki permintaan pada *port* 80, yang menggunakan *header* HTTP dan *content* yang berisi permintaan akses ke */malicious.php* ke *server*.

2.7.3.1.1.3 Rules Pass

Rules pass dalam Suricata adalah salah satu aksi dalam aturan deteksi yang digunakan untuk secara eksplisit mengecualikan paket dari proses inspeksi lebih lanjut. Dengan menggunakan aturan ini, Suricata dapat melewati lalu lintas tertentu yang dianggap aman atau tidak relevan tanpa memicu inspeksi atau analisis lebih dalam. Proses kerja aturan *pass* ini melibatkan tahapan sistematis mulai dari penerimaan paket hingga pengambilan keputusan akhir untuk meneruskan lalu lintas tanpa inspeksi.

Ketika paket diterima melalui antarmuka jaringan yang telah dikonfigurasi, Suricata memulai proses analisis awal untuk memahami struktur dan tujuan dari paket tersebut. Informasi yang diambil mencakup elemen-elemen penting seperti header protokol (IP, TCP, atau UDP), alamat sumber dan tujuan, port yang digunakan, serta payload jika tersedia. Tahap ini memastikan bahwa setiap paket memiliki metadata lengkap yang nantinya akan digunakan untuk pencocokan terhadap aturan.

Setelah *parsing* awal selesai, Suricata membandingkan paket dengan semua aturan yang telah didefinisikan dalam konfigurasi. Aturan dalam Suricata bekerja dengan urutan tertentu, dan aturan dengan aksi *pass* biasanya memiliki prioritas tinggi untuk memastikan bahwa lalu lintas yang sesuai dapat dilewatkan segera tanpa memengaruhi kinerja sistem. Pada tahap ini, Suricata melakukan proses pencocokan berdasarkan beberapa aspek, seperti protokol yang digunakan (TCP, UDP, ICMP), alamat IP sumber dan tujuan, serta nomor port. Jika paket memenuhi semua kriteria yang disebutkan

dalam aturan *pass*, maka Suricata segera mengambil keputusan untuk menghentikan inspeksi lebih lanjut. Sebagai contoh, aturan berikut:

```
pass tcp 192.168.1.10 any -> 192.168.1.20 80 (msg:"Allow  
http Traffic";sid:1002;)
```

Gambar 2. 15 Contoh *Rules Pass* Suricata

Pada *rules pass* ini, Suricata akan menginstruksikan Suricata untuk melewati semua lalu lintas TCP dari IP sumber 192.168.1.10 ke IP tujuan 192.168.1.20 pada port 80 tanpa inspeksi lebih lanjut.

Ketika sebuah paket cocok dengan aturan *pass*, proses inspeksi untuk paket tersebut dihentikan. Paket tidak lagi melalui aturan lain yang mungkin memiliki aksi *alert* atau *drop*. Hal ini dilakukan untuk menghemat sumber daya komputasi dengan menghindari analisis terhadap lalu lintas yang dianggap aman. Setelah keputusan dibuat untuk melewati paket, Suricata menyerahkan paket kembali ke antarmuka jaringan untuk diteruskan ke tujuan akhir. Tidak ada modifikasi yang dilakukan pada paket, dan tidak ada *log* yang dibuat secara *default*.

2.7.3.1.1.4 *Rules Reject*

Perintah *reject* dalam Suricata adalah aksi yang digunakan untuk secara aktif memutus lalu lintas yang dianggap tidak sah atau mencurigakan. Tidak seperti perintah *drop* yang hanya memblokir paket secara diam-diam tanpa memberikan respons kepada pengirim, perintah *reject* mengirimkan respons balik ke pengirim untuk memberi tahu bahwa koneksi atau paket telah ditolak. Proses ini melibatkan beberapa tahapan dari penerimaan paket hingga penanganan aktif untuk menolak lalu lintas tersebut.

Suricata menerima paket melalui antarmuka jaringan dan paket akan dianalisis pada tahap awal untuk mengidentifikasi protokol yang digunakan (seperti TCP, UDP, atau ICMP), alamat sumber dan tujuan, serta payload jika ada. Informasi ini digunakan untuk menentukan apakah paket memenuhi kondisi aturan yang telah ditentukan.

```
reject tcp any any -> 192.168.1.100 80 (msg:"Blokir akses ilegal";content:"/admin";http_uri;nocase;sid:1003;rev:1;)
```

Gambar 2. 16 Contoh *Rules Reject* Suricata

Aturan ini bekerja dengan memonitor semua lalu lintas TCP dari sumber mana pun (*any any*) yang menuju ke alamat IP tujuan 192.168.1.100 pada port 80 (HTTP). Ketika mendeteksi permintaan HTTP yang berisi *string* `"/admin"` di bagian URI (*Uniform Resource Identifier*), Suricata akan mencocokkan pola tersebut menggunakan parameter *content* dan *http_uri*. Opsi *nocase* memungkinkan pencocokan tidak bergantung pada huruf besar atau kecil. Jika aturan ini terpenuhi, Suricata akan secara aktif mengirimkan paket RST (*Reset*) untuk memutus koneksi kepada pengirim, sehingga mencegah akses lebih lanjut ke URI tersebut. Selain itu, log akan mencatat pesan "Blokir akses ilegal ke *web server*" untuk membantu administrator melacak kejadian ini.

2.7.3.1.2 Mode IPS

Dalam mode *Intrusion Prevention System* (IPS), Suricata tidak hanya mendeteksi ancaman, tetapi juga mengambil langkah-langkah aktif untuk mencegah dampak dari ancaman tersebut. Suricata beroperasi secara *inline*, yang berarti sistem ini berada di jalur lalu lintas jaringan, memungkinkan untuk memproses dan memblokir paket berbahaya sebelum mencapai tujuan yang dilindungi.

Untuk mendukung kinerjanya dalam mode IPS, Suricata memanfaatkan *AF-Packet*, yang merupakan modul kernel Linux untuk menangkap paket jaringan secara langsung dari *interface* jaringan dengan latensi rendah dan performa tinggi. Keunggulan *AF-Packet* terletak pada kemampuannya untuk melakukan *zero-copy*, yaitu pemrosesan paket tanpa perlu menyalin data dari kernel ke ruang pengguna, yang mengurangi *overhead* dan meningkatkan efisiensi. Selain itu, *AF-Packet* mendukung pemrosesan *multithreading*, yang memungkinkan Suricata untuk memanfaatkan banyak CPU dalam menangani lalu lintas jaringan dengan *throughput* tinggi.

Selain mode *af-packet*, Suricata mendukung berbagai metode akuisisi paket (*packet acquisition methods*) untuk menganalisis lalu lintas jaringan. Salah satu metode tersebut adalah mode NFQUEUE, yang memungkinkan Suricata digunakan sebagai *Intrusion Prevention System* (IPS) secara *inline* dengan bantuan sistem *firewall* Linux berbasis Netfilter. NFQUEUE merupakan salah satu target dalam subsistem Netfilter pada kernel Linux yang memungkinkan paket jaringan diteruskan ke ruang pengguna (*userspace*) untuk diproses oleh aplikasi eksternal. Dalam konteks Suricata, NFQUEUE berfungsi sebagai perantara antara *iptables* dan *engine* analisis Suricata, sehingga Suricata dapat mengambil keputusan untuk meneruskan (*accept*) atau memblokir (*drop*) paket secara *real-time*.

NFQUEUE Dalam mode Pencegahan Intrusi, Suricata harus dapat membuang paket jika paket tersebut berisi sesuatu yang dirujuk sebagai lalu lintas berbahaya. Untuk melakukannya, Suricata perlu menggunakan metode penangkapan yang berbeda. Metode penangkapan IPS yang paling banyak digunakan adalah antrean *Netfiter*. Metode ini memungkinkan Suricata untuk melakukan tindakan seperti *DROP* atau *ACCEPT* pada paket. Rangkaian aturan *firewall* perlu diperbarui. Aturan baru harus ditambahkan untuk mengirim lalu lintas ke Suricata yang terhubung ke antrean untuk diambil keputusan. Langkah-langkah berikut menjelaskan cara kerja NFQUEUE dengan Suricata dalam mode IPS.

Alur kerja mode NFQUEUE memproses paket jaringan yang masuk melalui *interface* menggunakan kernel Linux dan akan dialihkan ke NFQUEUE menggunakan *firewall* pada *iptables*. Aturan *iptables* digunakan untuk menangkap paket yang sesuai dengan kriteria tertentu, seperti protokol, *port*, atau alamat IP. Paket-paket ini diarahkan ke antrean NFQUEUE menggunakan target NFQUEUE dalam aturan *iptables*. Opsi seperti *--queue-num* digunakan untuk menentukan nomor antrean tempat paket akan diproses lebih lanjut oleh aplikasi seperti Suricata.

```
iptables -I FORWARD -j NFQUEUE --queue-num 0
```

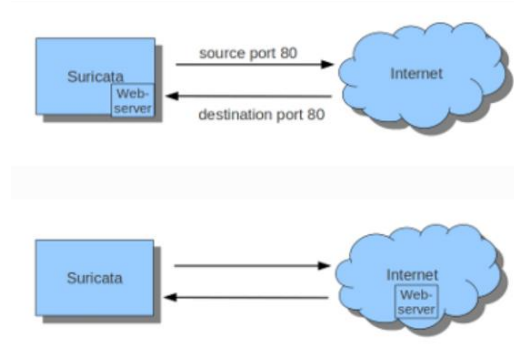
Gambar 2. 17 Aturan *Iptables FORWARD*

Aturan *iptables* pada gambar bekerja untuk mengalihkan paket dan mengirim traffic ke Suricata. Untuk membuat Suricata mengatur *traffic* masuk dan keluar pada salah satu *port* (contoh *port* 80), yang memungkinkan Suricata melakukan inspeksi terhadap lalu lintas HTTP, mendeteksi serangan, serta mencatat aktivitas yang mencurigakan. Aturan *iptables*-nya sebagai berikut:

```
iptables -I INPUT -p tcp --sport 80 -j NFQUEUE --queue-num 0
iptables -I OUTPUT -p tcp --dport 80 -j NFQUEUE --queue-num 0
```

Gambar 2. 18 Aturan *Iptables* pada *Port* 80

Dari aturan *iptables* pada gambar, suricata mengatur permintaan masuk dan keluar dari server seperti ilustrasi berikut:



Gambar 2. 19 Ilustrasi Lalu Lintas Jaringan HTTP antara Suricata dan Internet

Paket-paket yang masuk ke antrian NFQUEUE akan ditangani oleh aplikasi pengguna, dalam hal ini Suricata, yang dijalankan dengan opsi `-q` seperti pada gambar berikut:

```
suricata.service - LSB: Next Generation IDS/IPS
Loaded: loaded (/etc/init.d/suricata; generated)
Active: active (running) since Thu 2025-05-01 12:30:02 WIB; 59min ago
Docs: man:systend-sysv-generator(8)
Tasks: 10 (limit: 4562)
Memory: 63.8M
CPU: 48.821s
CGroup: /system.slice/suricata.service
└─858 /usr/bin/suricata -c /etc/suricata/suricata.yaml --pidfile /var/run/suricata.pid -q 0 -

y 01 12:30:02 suricata systemd[1]: Starting LSB: Next Generation IDS/IPS...
y 01 12:30:02 suricata susicata[686]: Starting suricata in IPS (nfqueue) mode... done.
y 01 12:30:02 suricata systemd[1]: Started LSB: Next Generation IDS/IPS.
```

Gambar 2. 20 Suricata Menggunakan Mode NFQUEUE

Suricata akan memproses paket sesuai dengan *rules* deteksi yang telah dikonfigurasi. Berdasarkan hasil analisis, Suricata akan memutuskan apakah paket akan diteruskan kembali ke kernel (dan akhirnya ke tujuan akhir) atau diblokir.

Dalam implementasinya, mode IPS pada Suricata sering kali menggunakan dua *interface* jaringan untuk menangkap dan memproses lalu lintas yang masuk (*ingress*) dan keluar (*egress*). Penggunaan dua *interface* ini memberikan fleksibilitas bagi Suricata untuk menangani lalu lintas dengan lebih efisien dan memungkinkan sistem untuk mengambil tindakan pencegahan yang lebih efektif terhadap ancaman yang terdeteksi.

interface pertama berfungsi untuk menangkap paket yang datang ke jaringan (*ingress traffic*). Setelah paket diterima, Suricata akan menganalisisnya berdasarkan aturan yang telah ditentukan, seperti aturan berbasis tanda tangan (*signature-based*) atau aturan berbasis perilaku. Jika paket terdeteksi mengandung ancaman, Suricata akan melakukan tindakan pencegahan, misalnya dengan memblokir paket atau mengirimkan *TCP Reset* untuk menghentikan koneksi yang mencurigakan. Selanjutnya, paket yang telah diproses akan diteruskan ke *interface* kedua, yang digunakan untuk menangkap lalu lintas yang keluar dari jaringan (*egress traffic*). Pada *interface* kedua ini, Suricata dapat memonitor dan mengelola lalu lintas yang keluar, serta memutuskan untuk memblokir koneksi atau mengirimkan *TCP Reset* jika ditemukan ancaman pada paket yang keluar. Dengan menggunakan dua *interface* ini, Suricata mampu untuk melakukan pemrosesan lalu lintas secara *real-time* pada kedua arah. Hal ini memungkinkan Suricata untuk merespons ancaman dengan cepat dan melakukan tindakan pencegahan lebih awal sebelum ancaman dapat menyebar lebih jauh di jaringan. Selain itu, dengan pemisahan pemrosesan lalu lintas masuk dan keluar, Suricata dapat lebih efektif dalam mengontrol dan mengamankan *flow_id* dalam jaringan.

2.7.3.1.3 Paket Data

Dalam jaringan komputer, data yang dikirim dan diterima melalui jaringan dikemas dalam bentuk paket. Paket data jaringan merupakan unit terkecil dari informasi yang ditransmisikan melalui jaringan. Setiap paket memiliki struktur tertentu yang mencakup informasi penting, seperti data yang dikirim, tujuan pengiriman, dan metadata yang membantu pengiriman data secara efisien. Paket data memungkinkan komunikasi antar perangkat dalam jaringan, termasuk pertukaran informasi antara klien dan server, perangkat IoT, dan aplikasi berbasis jaringan. Paket-paket ini bekerja berdasarkan protokol jaringan, seperti *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP), atau protokol lainnya, yang menentukan bagaimana data

dipecah menjadi paket, diteruskan melalui jaringan, dan disusun kembali di tujuan. Struktur paket secara umum terdiri dari tiga bagian utama, yaitu bagian *Header* yang berisi informasi kontrol, seperti alamat sumber, alamat tujuan, panjang paket, serta protokol yang digunakan, bagian *Payload* yang berisi data aktual yang dikirim, seperti file, teks, atau perintah, dan *Trailer* yang merupakan bagian tambahan yang biasanya berisi checksum untuk memastikan integritas paket selama transmisi.

<pre> "Ethernet": { "Destination MAC": "FF:FF:FF:FF:FF:FF", "Source MAC": "00:1A:A0:B6:C2:D4", "Type": "IPv4" }, "IP": { "Version": 4, "Header Length": "20 bytes", "Total Length": "74 bytes", "Source IP": "192.168.1.1", "Destination IP": "192.168.1.2", "Protocol": "TCP" }, "TCP": { "Source Port": 1234, "Destination Port": 80, "Flags": ["SYN", "ACK"] } </pre>	<pre> "HTTP": { "Method": "POST", "URL": "/submit-data", "Protocol": "HTTP/1.1", "Headers": { "Host": "www.example.com", "User-Agent": "Mozilla/5.0", "Content-Type": "application/json", "Content-Length": 29 }, "Payload": { "data": { "username": "user123", "password": "pass456" } } } </pre>
--	--

Gambar 2. 21 Contoh Paket Data Normal

Gambar 2.21 merupakan *packet* data normal yang sudah diuraikan dari bentuk awal *raw packet* atau paket mentah. *Packet* data tersebut memiliki bagian *header Ethernet* yang menentukan tujuan dan sumber *MAC address* serta tipe protokol. *Header IP* yang mengidentifikasi protokol *TCP* yang digunakan dan alamat *IP* sumber serta tujuan. *Header TCP* yang menyediakan informasi *transport* seperti *port* sumber dan tujuan, serta kontrol *flag*. *Header HTTP* yang menguraikan informasi *header* dari permintaan *HTTP POST*, termasuk tipe konten dan panjang *payload*. Lalu ada bagian *payload* yang merupakan data mentah yang dikirim dalam *request POST*. *Payload* pada *layer* aplikasi (*HTTP*) biasanya berisi data mentah seperti string, *JSON*, atau file dalam format biner. Dalam jaringan, paket data dapat dikategorikan berdasarkan kondisinya menjadi paket normal dan paket abnormal.

2.7.3.1.3.1 Paket Data Normal

Paket normal adalah paket yang sepenuhnya mematuhi spesifikasi protokol yang digunakan dalam jaringan. Paket ini digunakan untuk komunikasi rutin dan tidak menunjukkan indikasi aktivitas berbahaya.

```
{
  "header": {
    "source_ip": "203.0.113.5",
    "destination_ip": "192.168.1.100",
    "protocol": "TCP",
    "source_port": 34567,
    "destination_port": 80,
    "flags": ["SYN", "ACK"],
    "sequence_number": 1001,
    "acknowledgment_number": 5001
  },
  "payload": {
    "http_request": {
      "method": "GET",
      "url": "/home",
      "http_version": "HTTP/1.1",
      "headers": {
        "Host": "myserver.local",
        "User-Agent": "Mozilla/5.0"
      }
    }
  }
}
```

Gambar 2. 22 Paket data HTTP Normal

Pada paket tersebut menunjukkan permintaan HTTP *GET* berasal dari luar jaringan lokal yang mengarah ke server dengan alamat IP lokal 192.168.1.100. *source_ip* 203.0.113.5 merupakan alamat IP perangkat sumber pengguna yang mengirimkan permintaan, *destination_ip* 192.168.1.100 yang merupakan alamat IP tujuan, yaitu server di dalam jaringan lokal yang menerima permintaan. Paket ini menggunakan protokol TCP untuk komunikasi antara klien dan server.

Pada *header* paket, *source_port* adalah 34567, yang menunjukkan *port* acak yang digunakan oleh perangkat luar untuk mengirimkan permintaan, sedangkan *destination_port* adalah 80, yang merupakan *port* standar untuk HTTP pada server tujuan. *Flag* SYN dan ACK menunjukkan bahwa paket ini bagian dari tahap awal dari

koneksi TCP yang dilakukan antara perangkat luar dan server di klien, yang dimulai dengan proses handshake TCP untuk membangun koneksi yang stabil. *sequence_number* dan *acknowledgment_number* digunakan untuk memastikan urutan pengiriman paket dan memastikan bahwa komunikasi berhasil dilakukan antara kedua pihak.

Pada bagian *payload*, paket ini mengandung permintaan HTTP dengan metode *GET*, yang berarti bahwa perangkat luar meminta untuk mengambil sumber daya yang ada pada server di dalam jaringan lokal. *url* yang diminta adalah */home*, yang bisa jadi halaman utama atau halaman tertentu dari aplikasi yang di-*hosting* di server tersebut. Protokol HTTP yang digunakan adalah versi HTTP/1.1, yang merupakan versi umum dan mendukung fitur-fitur seperti pengelolaan koneksi yang lebih baik. Pada paket ini mencakup informasi seperti *Host*, yang menunjukkan bahwa server yang diminta adalah *myserver.local*, serta *User-Agent*, yang menunjukkan jenis *browser* atau perangkat yang digunakan oleh pengguna luar.

2.7.3.1.3.2 Packet Data Abnormal

Paket abnormal adalah paket yang tidak sesuai dengan spesifikasi protokol atau berisi aktivitas mencurigakan. Paket ini sering digunakan untuk menyerang atau mengeksploitasi kelemahan dalam sistem jaringan. Paket-paket ini bisa menjadi indikator awal dari aktivitas mencurigakan. Deteksi terhadap paket abnormal sangat penting dalam sistem keamanan jaringan karena dapat membantu mencegah serangan sebelum berdampak lebih jauh. Suricata sebagai IDS/IPS mampu mengenali pola-pola paket abnormal melalui rule dan signature yang telah ditentukan sebelumnya.

Paket abnormal dapat muncul akibat manipulasi pada *header*, *payload* yang tidak lazim, atau urutan transmisi yang tidak sesuai standar protokol. Contohnya termasuk paket dengan ukuran tidak wajar, *flag TCP* yang tidak valid, atau *payload* yang

mengandung kode berbahaya. Dalam banyak kasus, paket-paket ini dimanfaatkan dalam teknik *reconnaissance*, *exploit*, atau *denial of service* (DoS), sehingga penting untuk mendeteksinya secara dini guna menjaga integritas dan ketersediaan layanan jaringan.

```
{
  "header": {
    "source_ip": "10.10.10.10",
    "destination_ip": "192.168.1.100",
    "protocol": "TCP",
    "source_port": 65432,
    "destination_port": 80,
    "flags": ["ACK"]
  },
  "payload": {
    "request_method": "POST",
    "url": "/upload.php",
    "http_version": "HTTP/1.1",
    "headers": {
      "Host": "target.com",
      "Content-Type": "application/x-executable"
    },
    "body": {
      "file": "exploit.exe",
      "content": "binary malware data"
    }
  }
}
```

Gambar 2. 23 Paket Data Tidak Normal

Pada paket ini, sumber permintaan datang dari alamat IP 10.10.10.10, yang merupakan perangkat klien yang mencoba mengirimkan data ke server yang memiliki alamat IP 192.168.1.100. Paket ini menggunakan protokol TCP untuk berkomunikasi, di mana klien mengirimkan data ke server melalui *source_port* 65432 ke *destination_port* 80, yang merupakan *port* standar untuk layanan HTTP. Paket ini memiliki *flag* ACK yang menunjukkan bahwa ini adalah bagian dari koneksi TCP yang sudah terjalin sebelumnya, dan paket ini digunakan untuk memastikan pengiriman data lebih lanjut dalam sesi yang sudah aktif.

Pada bagian *payload*, permintaan HTTP yang dikirimkan menggunakan metode *POST*, yang menunjukkan bahwa klien berusaha mengirimkan data ke server. *url* yang diminta adalah */upload.php*, yang bisa jadi adalah halaman atau skrip PHP di server yang menerima file untuk di-*upload*. Pada paket ini terdapat informasi tentang *Host*, yang

menunjukkan nama *domain* server tujuan (*target.com*), serta *Content-Type* yang menyatakan bahwa data yang dikirimkan berupa file dalam format *application/x-executable*. Ini adalah indikasi bahwa file yang dikirimkan adalah sebuah file eksekusi yang umumnya digunakan untuk aplikasi atau program perangkat lunak.

Bagian *body* dari paket ini berisi file dengan nama *exploit.exe*, yang mungkin merupakan file *malware* yang dimaksudkan untuk mengeksploitasi kerentanannya di server. Isi dari *content* adalah "*binary malware data*", yang menunjukkan bahwa file tersebut berisi data dalam format biner, yaitu data *malware* yang dapat merusak sistem jika dieksekusi.

2.8 Confusion Matrix

Dalam evaluasi sistem deteksi intrusi (IDS), dua metrik yang sering digunakan adalah Akurasi (*Accuracy*) dan *False Positive Rate (FPR)*[26]. Keduanya berasal dari analisis *confusion matrix*, yang membandingkan hasil prediksi sistem terhadap data aktual untuk menentukan kinerja deteksi.

Tabel 2. 5 *Confusion Matrix*

<i>Actual</i>	<i>Predicted</i>	<i>Predicted</i>
	<i>Attack</i>	<i>Normal</i>
<i>Attack</i>	TP	FN
<i>Normal</i>	FP	TN

Keterangan:

TP = *True Positive* adalah jumlah serangan yang terdeteksi dengan benar.

TN = *True Negative* adalah jumlah aktivitas normal yang dikenali dengan benar.

FP = *False Positive* adalah jumlah aktivitas normal yang keliru terdeteksi sebagai serangan.

FN = *False Negative* adalah jumlah serangan yang tidak terdeteksi

Akurasi mengukur sejauh mana sistem deteksi mampu mengklasifikasikan dengan benar baik aktivitas serangan maupun aktivitas normal. Perhitungannya menggunakan persamaan:

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \dots\dots\dots (2.10)$$

Sementara itu, *False Positive Rate* (FPR) mengukur proporsi aktivitas normal yang keliru diklasifikasikan sebagai serangan terhadap seluruh aktivitas normal yang diuji. Persamaannya adalah sebagai berikut:

$$FPR = \frac{FP}{FP+TN} \dots\dots\dots (2.11)$$

Selain *False Positive Rate* (FPR) yaitu *False Negative Rate* (FNR) untuk menguji kemampuan sesungguhnya dari sistem IDS/IPS dalam menghadapi serangan. Oleh karena itu, diperlukan perhitungan Tingkat Negatif Palsu (*False Negative Rate* - FNR). Metrik FNR adalah indikator paling penting dari efektivitas pertahanan karena secara spesifik mengukur proporsi kasus di mana serangan nyata berhasil menyusup tanpa terdeteksi atau terblokir oleh sistem. Persamaan yang digunakan adalah sebagai berikut:

$$FNR = \frac{FN}{TP+FN} \dots\dots\dots (2.12)$$

III. METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian ini dilakukan pada bulan September 2023 – Selesai, di Laboratorium Teknik Pengukuran Besaran Listrik, Laboratorium Terpadu Jurusan Teknik Elektro, Fakultas Teknik, Universitas Lampung.

3.2 Alat dan Bahan

Adapun alat dan bahan yang digunakan pada penelitian ini adalah sebagai berikut:

1. Satu unit laptop dengan spesifikasi Intel Core i5-8250U 1.60GHz, RAM 8 GB, ROM 225 GB, HDD 1,2 TB, serta OS Windows 10 Pro 64-bit sebagai media pemodelan dan simulasi, serta *host* sebagai Administrator.
2. Perangkat lunak Oracle VirtualBox versi 7.1.6 sebagai platform dual OS yang bersifat *open source*.
3. OS Linux Ubuntu pada VirtualBox yang digunakan untuk implementasi IDS/IPS menggunakan *tools* Suricata.
4. OS Kali Linux pada VirtualBox yang telah terinstall perangkat lunak OpenVAS sebagai alat *vulnerability scanning* dan *vulnerability assessment* serta *penetration testing*.
5. Satu buah *website* dinamis yang telah ter-*hosting* sebagai target *penetration testing* dan *vulnerability assessment* serta monitoring *log* menggunakan IDS/IPS.

3.3 Tahapan Penelitian

Adapun tahap penelitian ini adalah sebagai berikut:

1. Identifikasi Masalah

Tahap identifikasi masalah penulis melakukan observasi mengenai jenis serangan yang mungkin dilakukan pada *website* dinamis.

2. Studi Literatur

Penulis mengumpulkan sumber literatur dari mulai buku, artikel, laporan terdahulu, jurnal nasional maupun internasional pada tahap studi literatur untuk menunjang keberhasilan penelitian ini.

3. Studi Terbimbing

Penulis melakukan diskusi bersama dosen pembimbing setelah mengumpulkan referensi mengenai proses pembuatan, permasalahan, pemecahan masalah dengan cara yang lebih baik. Bagaimana cara menyelesaikan penelitian dengan baik.

4. Pengumpulan data dan Pengolahan Data

Tahap pengumpulan dan pengolahan data penulis melakukan pencarian data yang akan disimulasikan, adapun data yang digunakan dalam penelitian ini adalah *Website* dinamis dengan tingkat keamanan yang rendah dengan melakukan *vulnerability Assessment*, *penetration testing* dan monitoring menggunakan IDS/IPS. Selain mencari *website* sebagai target simulasi, penulis juga melakukan persiapan terkait *tools* yang akan digunakan dalam penelitian ini.

5. Simulasi dan Analisis

Penulis melakukan simulasi serangan pada *website* yang telah dilakukan *vulnerability scanning* (pemindaian kerentanan) setelah mengumpulkan data. Simulasi serangan dilakukan untuk mendapatkan *vulnerability assessment* (penilaian kerentanan) pada *website* dengan tingkat keamanan rendah. Dari simulasi tersebut maka akan didapatkan hasil analisa kerentanan yang ada pada *website* terkait dengan jenis serangan yang dilakukan. Lalu mengimplementasikan IDS/IPS menggunakan *tools* Suricata untuk memonitoring *log* aktivitas dan memblokir paket-paket yang diduga merupakan paket berbahaya yang ada pada

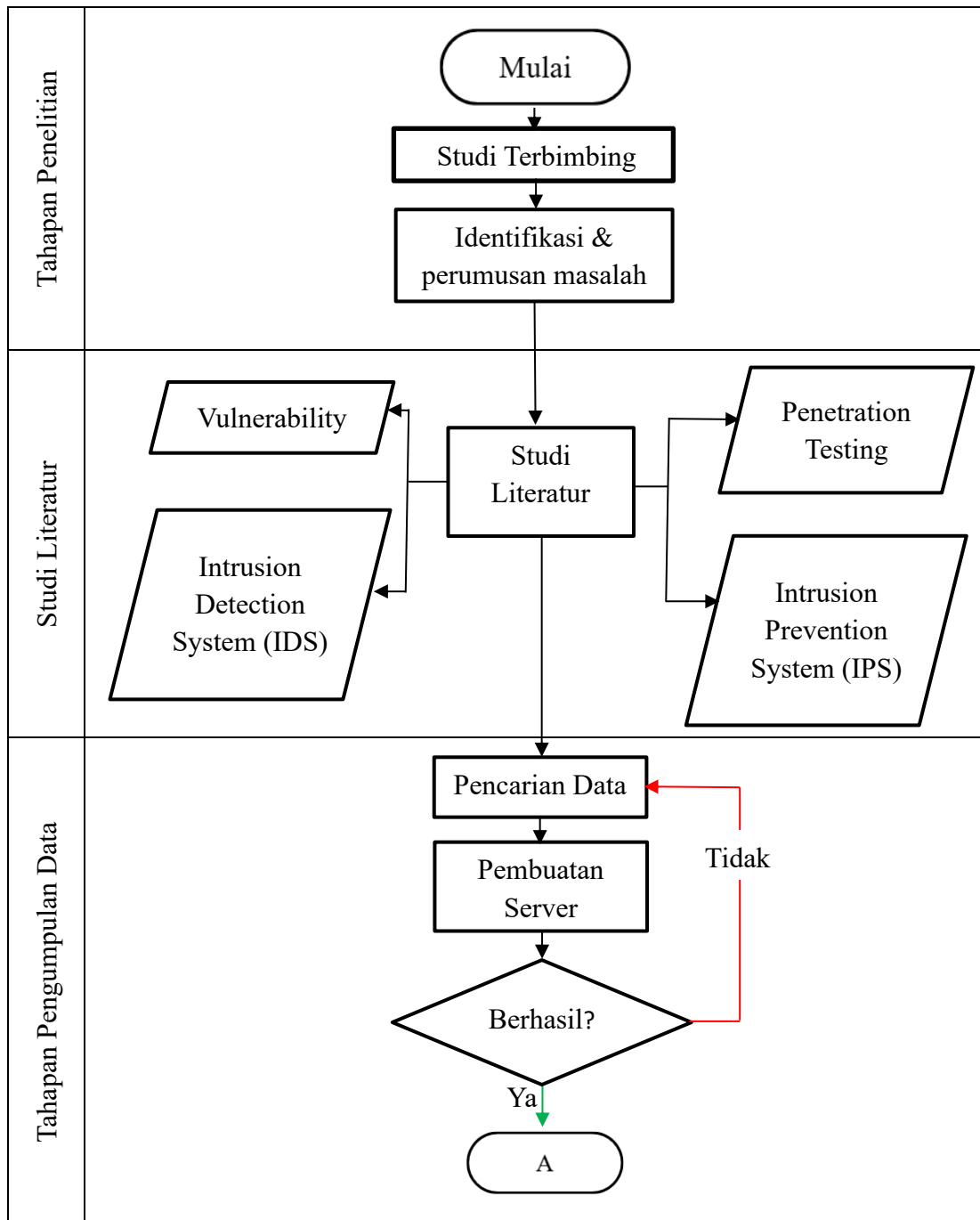
website tersebut. Lalu pesan peringatan akan dikirimkan melalui aplikasi Telegram bahwa adanya paket yang mencurigakan. Setelah mengimplementasikan IDS/IPS sebagai *server* untuk memonitoring *log* aktivitas pada *web server*, penulis melakukan simulasi serangan untuk dibandingkan dengan simulasi serangan sebelum diimplementasikan IDS/IPS pada *server* dan untuk mendapatkan pesan peringatan melalui aplikasi Telegram.

6. Pembuatan Laporan

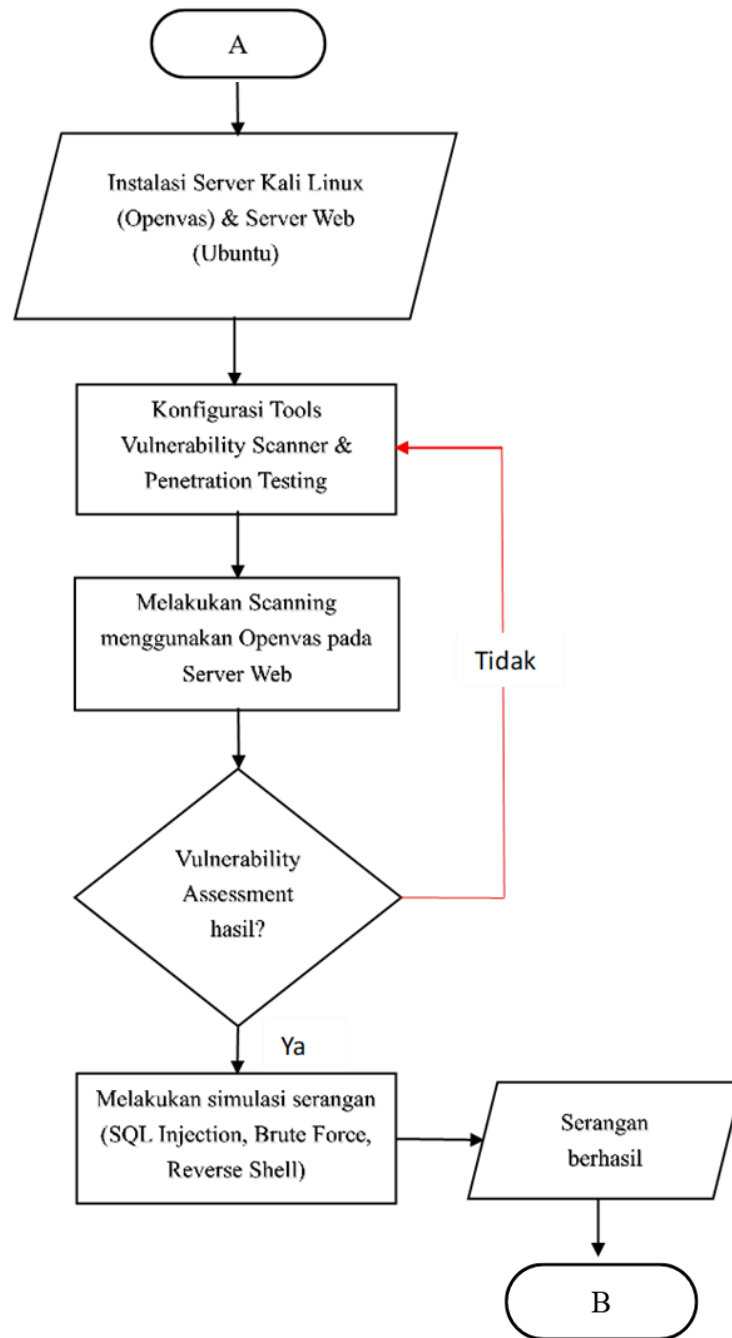
Penulis membuat laporan terkait hasil yang telah didapatkan dan sebagai sarana pertanggungjawaban terhadap penelitian yang telah dikerjakan. Laporan penelitian ini dibagi menjadi dua tahap, yaitu laporan awal yang digunakan untuk seminar proposal dan laporan akhir yang digunakan untuk seminar hasil.

3.4 Diagram Alir Penelitian

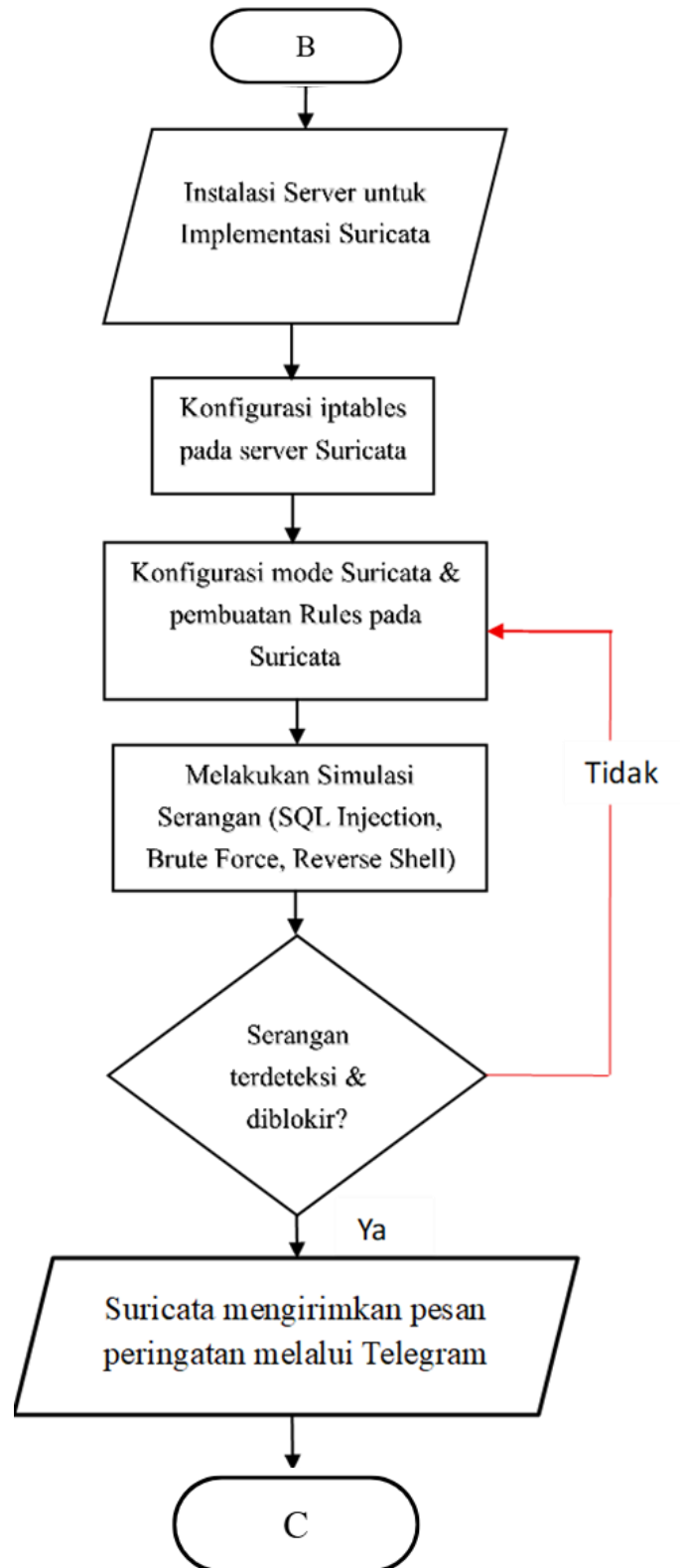
Adapun tahapan penelitian akan dijelaskan pada diagram alir penelitian yang ada pada Gambar 3.1 dibawah ini:

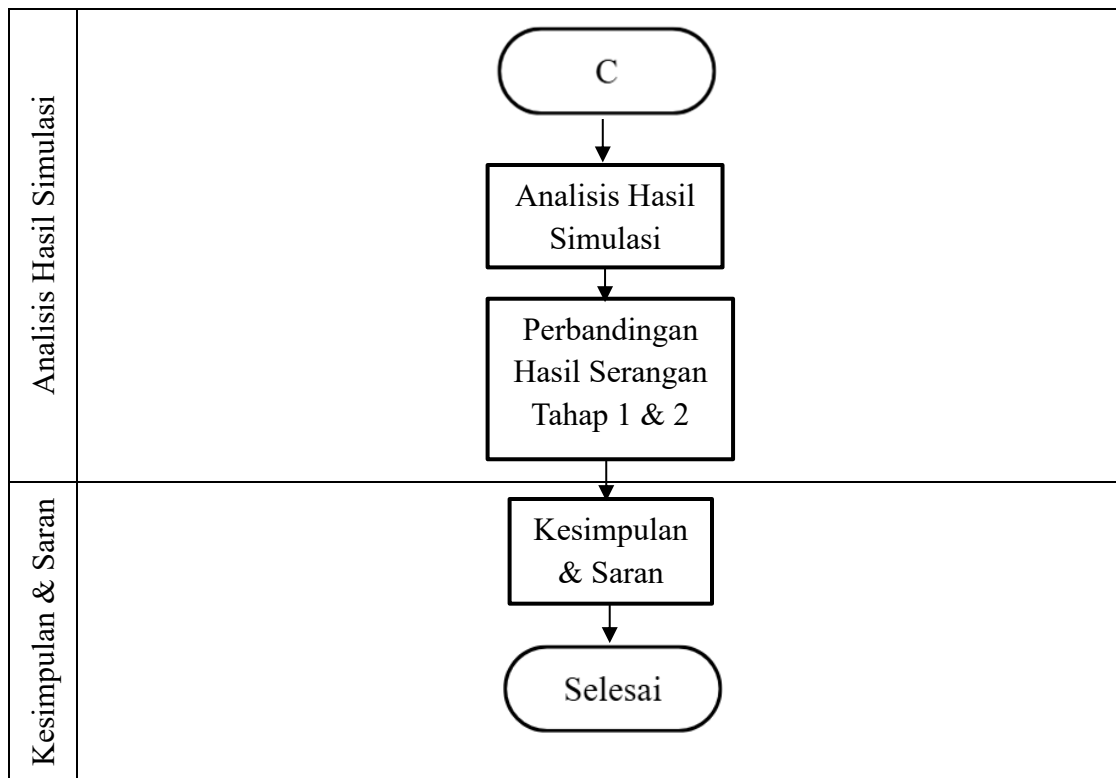


Perancangan & Simulasi Serangan Tahap 1



Perancangan & Simulasi Serangan Tahap 2





Gambar 3. 1 Diagram Alir Penelitian

3.5 Pengumpulan Data

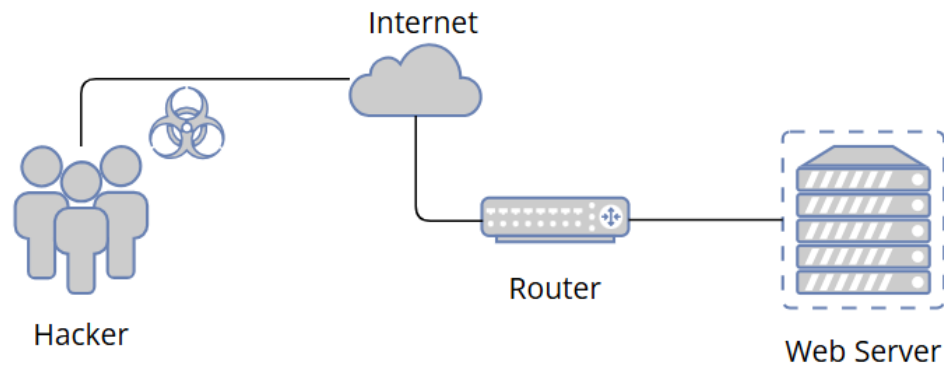
Pengumpulan data adalah suatu proses yang digunakan untuk mendapatkan dan mengumpulkan data yang diperlukan. Adapun langkah-langkah pengumpulan data pada penelitian ini adalah sebagai berikut :

1. Menyiapkan *Website* yang telah *ter-hosting* sebagai target simulasi.
2. Mencari *tools* yang relevan melalui studi literatur untuk mendukung simulasi yang akan dilakukan.
3. Meng-*install* VirtualBox untuk membuat *server* yang akan digunakan untuk simulasi. *Server* yang dibuat menggunakan OS Kali Linux untuk meng-*install tools* OpenVAS untuk *vulnerability assessment* dan OS Linux Ubuntu untuk meng-*install tools* Suricata sebagai *server* yang memonitoring *log* aktivitas dan

memblokir paket berbahaya pada *web server* serta untuk mengirimkan pesan peringatan adanya paket berbahaya yang masuk pada *web server*.

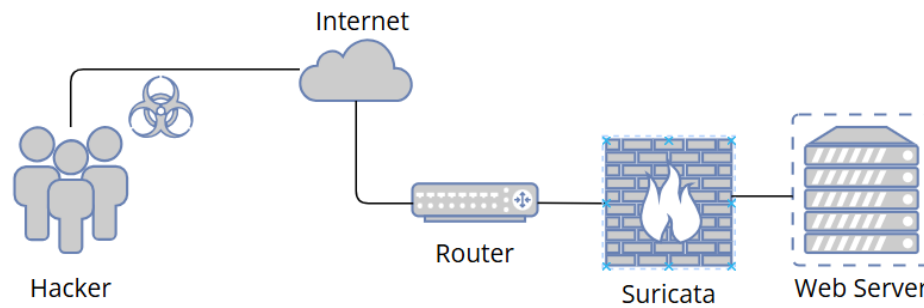
3.6 Perancangan dan Pengembangan Topologi Jaringan

Pada tahap ini, simulasi penelitian dirancang dengan tiga *server* menggunakan *virtual machine* dengan sistem operasi Linux Ubuntu dan Kali Linux. Topologi yang digunakan terdiri dari dua *server* yang bertindak sebagai IDS/IPS yang telah ter-*install tools* Suricata di sistem operasi Linux Ubuntu dan *server* untuk *vulnerability assessment* yang telah ter-*install tools* OpenVAS di sistem operasi Kali Linux, serta terdapat satu *server* sebagai *web server*.



Gambar 3. 2 Topologi Jaringan Sebelum Diimplementasikan *Server* IDS/IPS

Berdasarkan Gambar 3.2, simulasi menggunakan jaringan *virtual* sederhana untuk mensimulasikan komunikasi yang terjadi antara *host* dan *server*. Pada simulasi ini terdapat satu *host* yang bertindak sebagai *client* sekaligus *attacker* untuk melakukan penyerangan melalui sistem operasi Kali Linux. Topologi pada Gambar 3.2 merupakan jaringan sebelum diimplementasikannya *server* IDS/IPS. *Server* OpenVAS bertindak sebagai *host* yang melakukan *vulnerability scanning* dan *assessment* terhadap *web server* yang telah dilakukan penyerangan.



Gambar 3. 3 Topologi Jaringan Setelah Diimplementasikan *Server IDS/IPS*

Berdasarkan Gambar 3.3, merupakan jaringan yang telah diimplementasikan *server* IDS/IPS. *Server* IDS/IPS bertindak sebagai *host* yang akan memonitoring *log* aktivitas dan memblokir jika ada paket mencurigakan yang masuk pada *web server*, serta mengirimkan pesan peringatan melalui aplikasi Telegram.

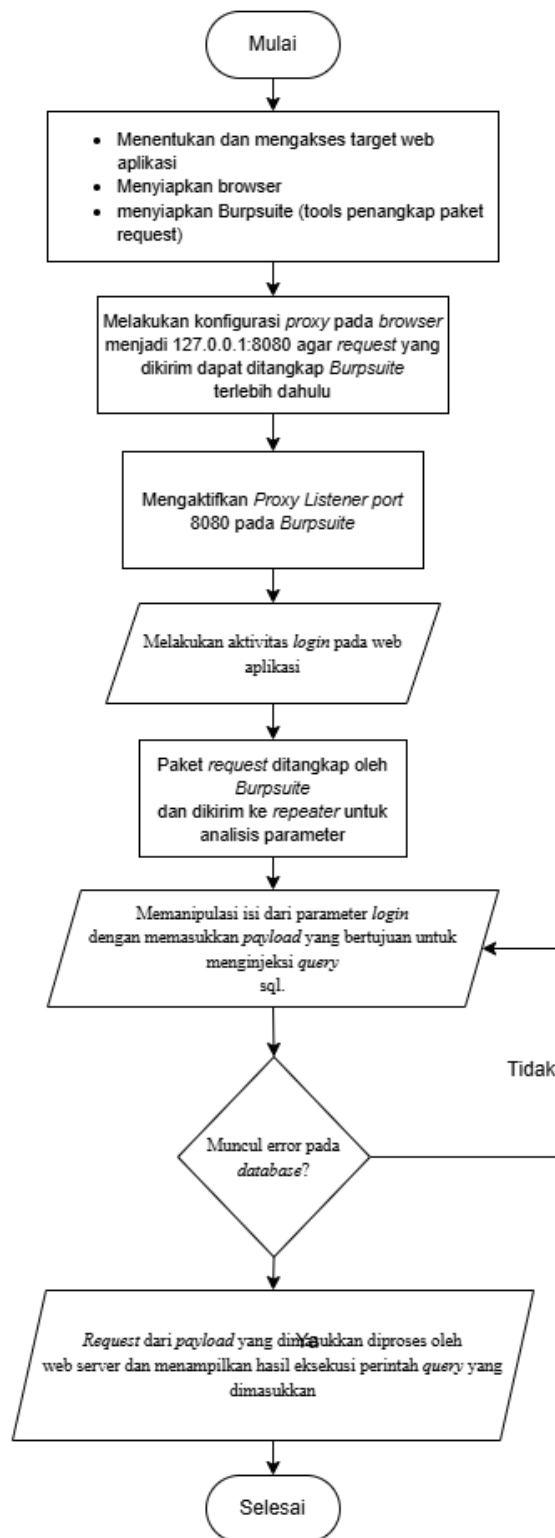
3.7 *Penetration Testing*

Penelitian ini menggunakan metode *Penetration Testing* untuk menguji tingkat keamanan *web application* target terhadap berbagai jenis serangan yang umum terjadi pada sistem berbasis *web*. Pengujian dilakukan dengan mensimulasikan serangan secara terkontrol pada *web application dummy* dengan tujuan untuk mengidentifikasi celah keamanan, menganalisis dampak serangan, serta mengevaluasi kelemahan sistem baik dari sisi aplikasi maupun *server*. Dalam penelitian ini digunakan tiga teknik *penetration testing*, yaitu *SQL Injection*, *Brute Force*, dan *Reverse Shell*, yang dilakukan secara bertahap dan saling berkaitan.

3.7.1 *Penetration Testing Menggunakan Teknik SQL Injection*

Teknik *SQL Injection* digunakan sebagai tahap awal dalam proses *penetration testing* untuk menguji keamanan mekanisme validasi *input* pada *web application*, khususnya pada parameter *login*. Serangan ini dilakukan dengan memanipulasi *input* yang dikirimkan ke *server* untuk mengetahui apakah aplikasi web rentan terhadap injeksi *query* SQL. Tujuan dari pengujian ini adalah untuk mengidentifikasi kemungkinan *bypass* autentikasi atau kesalahan penanganan *query* yang dapat dimanfaatkan oleh penyerang.

Pengujian *SQL Injection* dilakukan dengan bantuan tools *BurpSuite* untuk melakukan *intercept*, analisis, dan manipulasi *request* HTTP yang dikirimkan dari *browser* ke *server*. Respon *server* kemudian dianalisis untuk menentukan apakah *input* yang diberikan diproses secara tidak aman oleh sistem.



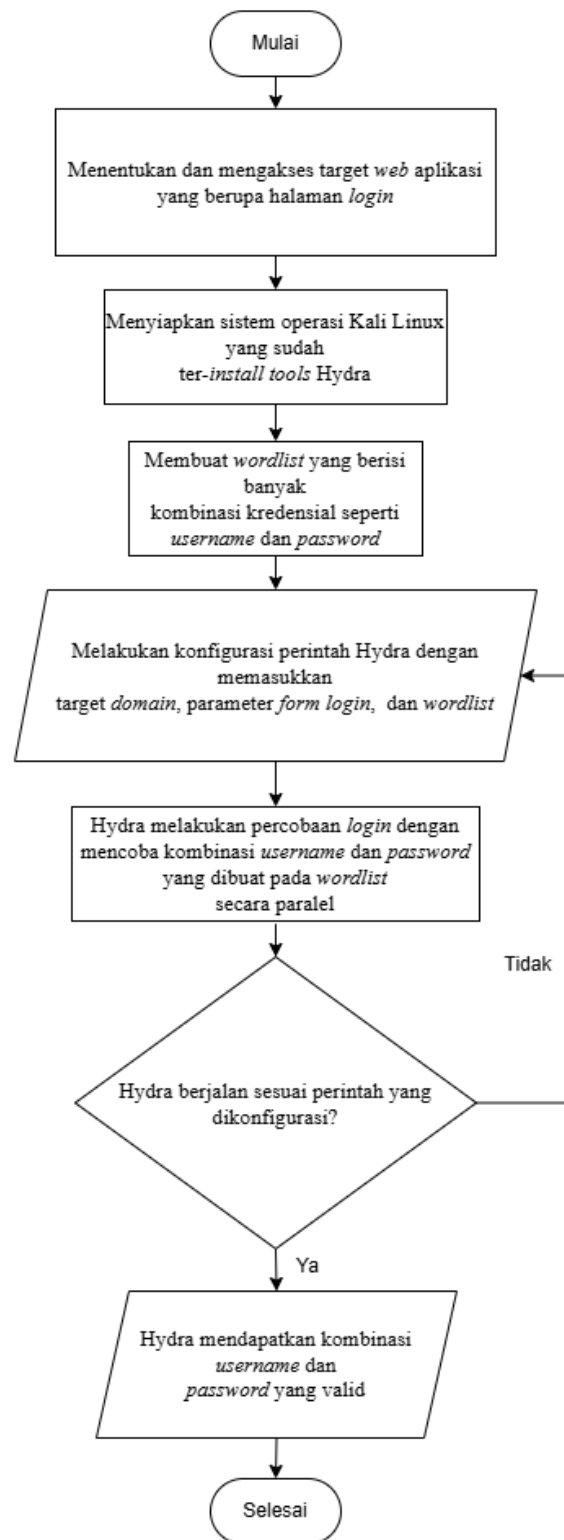
Gambar 3. 4 Diagram Alir *Penetration Testing SQL Injection*

Diagram alir *penetration testing* menggunakan teknik *SQL Injection* menggambarkan tahapan pengujian keamanan yang dimulai dari persiapan lingkungan uji hingga analisis hasil eksploitasi. Proses diawali dengan konfigurasi *browser* dan *BurpSuite* sebagai *proxy* untuk melakukan *intercept request* HTTP yang dikirimkan ke *server web application*. Selanjutnya, dilakukan pengujian pada parameter *login* dengan memanipulasi *input* untuk mengidentifikasi adanya celah pada mekanisme validasi *input*. Respon yang diberikan oleh *server* dianalisis untuk menentukan apakah aplikasi *web* rentan terhadap *SQL Injection*, seperti munculnya *error database* atau keberhasilan *bypass* autentikasi. Diagram ini menunjukkan alur sistematis dalam mengidentifikasi kelemahan *input handling* pada aplikasi *web* sebagai tahap awal *penetration testing*.

3.7.2 *Penetration Testing Menggunakan Teknik Brute Force*

Teknik *Brute Force* digunakan untuk menguji kekuatan mekanisme autentikasi *login* pada *web application*. Serangan ini dilakukan dengan mencoba berbagai kombinasi *username* dan *password* secara otomatis menggunakan *wordlist*. Tujuan dari pengujian ini adalah untuk mengetahui apakah sistem memiliki mekanisme pengamanan yang memadai terhadap percobaan *login* berulang, seperti pembatasan jumlah percobaan *login* atau deteksi aktivitas mencurigakan.

Pada penelitian ini, serangan *Brute Force* dilakukan menggunakan *tools Hydra* pada sistem operasi Kali Linux. *Hydra* digunakan untuk melakukan percobaan *login* secara paralel pada layanan HTTP dengan parameter *login* yang telah diidentifikasi sebelumnya. Hasil dari serangan ini berupa kredensial valid yang selanjutnya digunakan pada tahap *penetration testing* berikutnya.



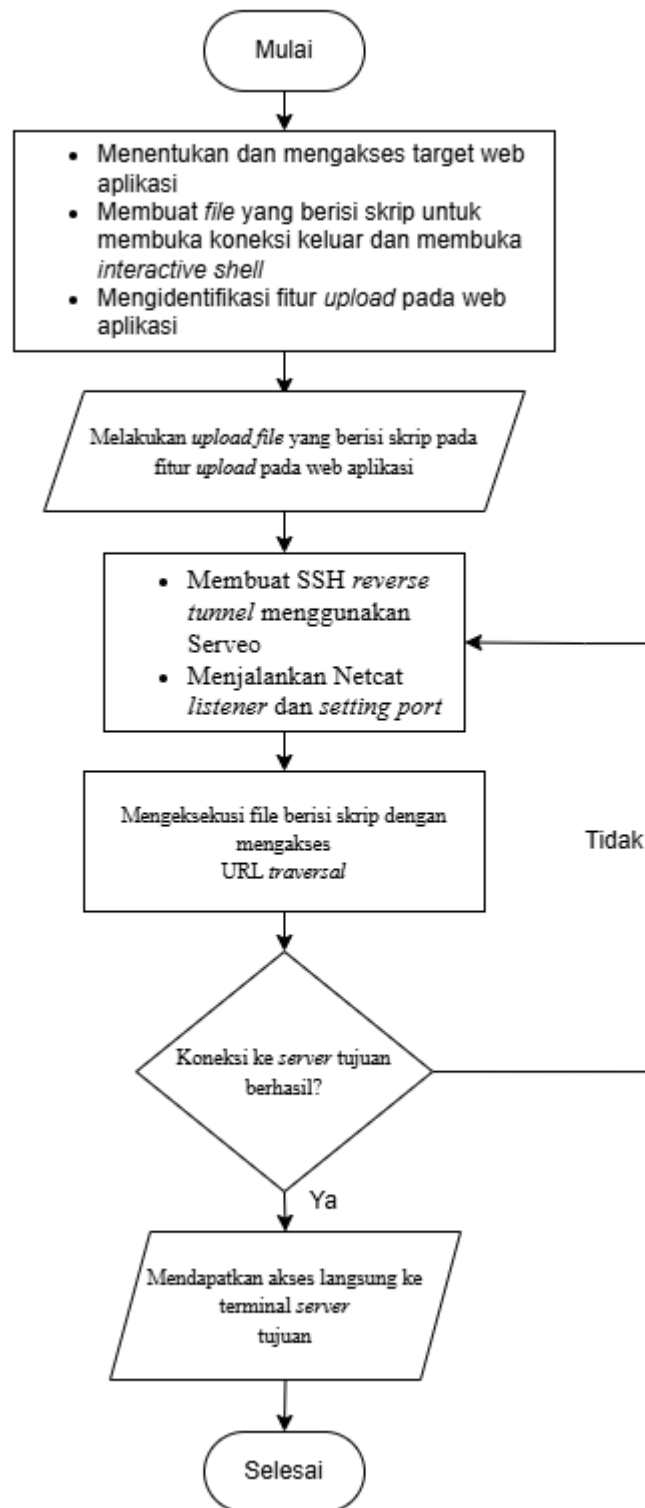
Gambar 3. 5 Diagram Alir *Penetration Testing Brute Force*

Diagram alir *penetration testing* menggunakan teknik *Brute Force* menunjukkan proses pengujian keamanan autentikasi *login* pada *web application*. Tahapan dimulai dengan penentuan target dan identifikasi parameter *login* yang digunakan oleh sistem. Selanjutnya dilakukan persiapan *wordlist username* dan *password*, serta konfigurasi *tools* Hydra pada sistem operasi Kali Linux. Serangan dilakukan dengan mencoba kombinasi kredensial secara otomatis dan paralel terhadap layanan HTTP. Respon *server* dianalisis untuk menentukan keberhasilan atau kegagalan proses *login*. Diagram ini memperlihatkan bagaimana teknik *brute force* dapat digunakan untuk mengevaluasi kekuatan mekanisme autentikasi dan mendeteksi kelemahan sistem terhadap percobaan *login* berulang.

3.7.3 *Penetration Testing Menggunakan Teknik Reverse Shell*

Reverse Shell digunakan sebagai tahap *post-exploitation* dalam penelitian ini, yaitu setelah kredensial valid berhasil diperoleh melalui serangan sebelumnya. Teknik ini bertujuan untuk mendapatkan akses langsung ke sistem *server* yang meng-*hosting web application* dengan memanfaatkan kelemahan pada fitur *upload file* dan konfigurasi *server*.

Pengujian dilakukan dengan mengunggah *file* berbahaya yang berisi *payload reverse shell* ke dalam *web application*, kemudian mengeksekusi *file* tersebut agar *server* target melakukan koneksi balik ke *server* penyerang. Setelah koneksi berhasil terbentuk, peneliti memperoleh akses *shell* ke sistem *server* dan melakukan analisis lanjutan, termasuk peningkatan hak akses (*privilege escalation*) untuk mengetahui dampak keamanan yang ditimbulkan.



Gambar 3. 6 Diagram Alir *Penetration Testing Reverse Shell*

Diagram alir *penetration testing* menggunakan teknik *Reverse Shell* menggambarkan tahapan *post-exploitation* setelah kredensial valid berhasil diperoleh. Proses dimulai dengan persiapan *payload reverse shell* yang kemudian diunggah ke *web application* melalui fitur *upload file*. Setelah *payload* berhasil diunggah dan dieksekusi, *server* target melakukan koneksi balik ke *server* penyerang yang telah menyiapkan *listener*. Koneksi ini memberikan akses *shell* ke sistem *server* sebagai *user web server*. Diagram selanjutnya menunjukkan tahapan peningkatan kualitas *shell* menjadi *pseudo-terminal* serta pemeriksaan hak akses pengguna untuk melakukan *privilege escalation*. Diagram ini menegaskan dampak serius dari keberhasilan *reverse shell*, yaitu potensi pengambilalihan penuh terhadap sistem *server*.

3.8 Tahapan Simulasi

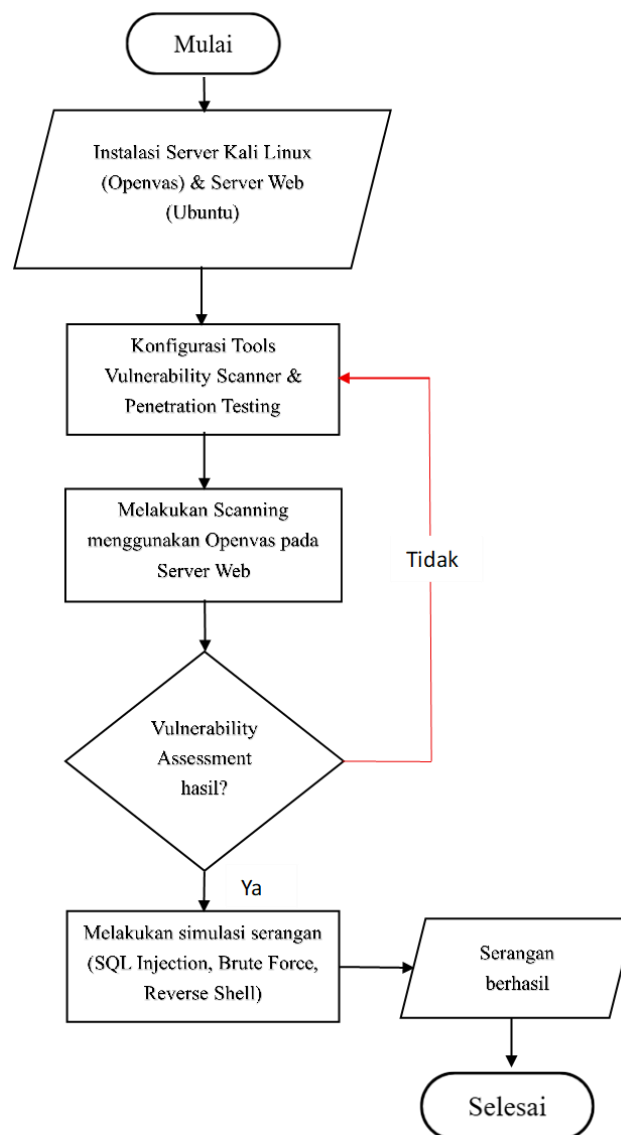
Tahapan simulasi dilakukan dengan dua skenario uji coba untuk mendapatkan hasil perbandingan setelah dilakukan serangan pada *web server* saat sebelum dan sesudah diimplementasikan IDS/IPS, karena tugas dari IDS/IPS adalah memonitoring dan memblokir jika ada paket mencurigakan yang masuk pada *server*, serta mengirimkan pesan peringatan pada administrator bahwa ada paket mencurigakan yang mencoba untuk masuk.

3.8.1 Simulasi Serangan Tahap 1

Adapun tahapan yang dilakukan pada simulasi serangan tahap 1 adalah sebagai berikut:

1. Melakukan instalasi dan konfigurasi *client server* untuk *penetration testing* pada sistem operasi Kali Linux dan *server web* menggunakan sistem operasi Ubuntu 22.04 LTS.

2. Melakukan konfigurasi pada *tools* OpenVAS untuk melakukan *vulnerability scanning* dan *assessment*.
3. Melakukan *scanning* menggunakan *tools* OpenVAS untuk mendapatkan nilai kerentanan pada *website* yang menjadi target simulasi.
4. Melakukan simulasi serangan pada *web application* target menggunakan jenis serangan *SQL Injection*, *Brute Force* dan *Reverse Shell*.

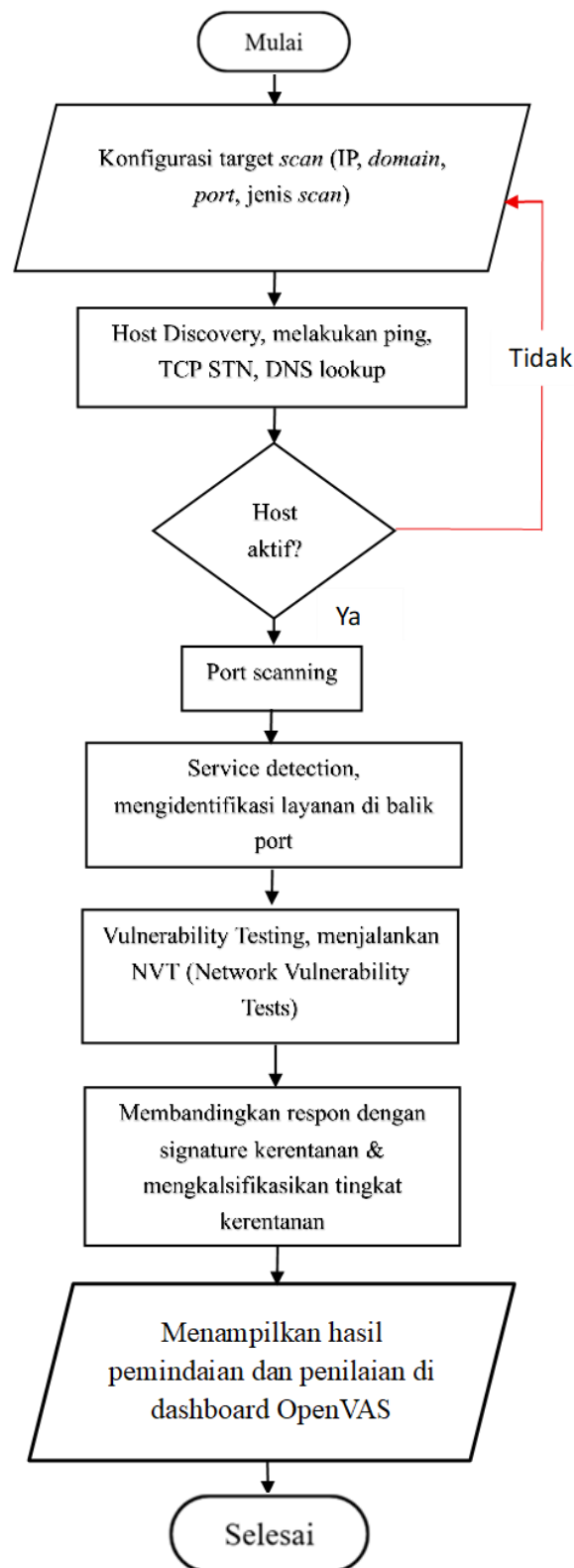


Gambar 3. 7 Diagram Alir Simulasi Tahap 1

Pada gambar 3.7 merupakan diagram alir simulasi serangan tahap 1 yang dilakukan untuk mendapatkan hasil dari kerentanan *website* target dan apakah simulasi serangan ini dapat berhasil terhadap *website* yang diserang.

Pada skenario tahap pertama terdapat tahap melakukan penilaian kerentanan menggunakan *tools* OpenVAS. Penilaian kerentanan ini memiliki beberapa tahap pemindaian yang diuraikan seperti pada Gambar 3.8. Adapun tahapan yang dilakukan pada proses pemindaian kerentanan menggunakan OpenVAS adalah sebagai berikut:

1. Melakukan Konfigurasi *Target Scan* yang meliputi pengaturan IP, *domain*, *port*, dan jenis scan yang akan dijalankan.
2. Melakukan *Host Discovery* dengan menjalankan perintah *ping*, TCP STN, dan DNS *lookup* untuk menemukan *host* target.
3. Memastikan *host* aktif, jika tidak aktif kembali ke langkah konfigurasi, jika aktif proses dilanjutkan dengan *Port Scanning*.
4. Melakukan *Service detection* untuk mengidentifikasi layanan yang berjalan di balik *port* yang terbuka.
5. Melakukan *Vulnerability Testing* dengan menjalankan NVT (*Network Vulnerability Tests*) untuk mencari kerentanan.
6. Membandingkan respon dengan *signature* kerentanan yang ada dan melakukan klasifikasi tingkat kerentanan.
7. Menampilkan hasil pemindaian dan penilaian di *dashboard* OpenVAS.



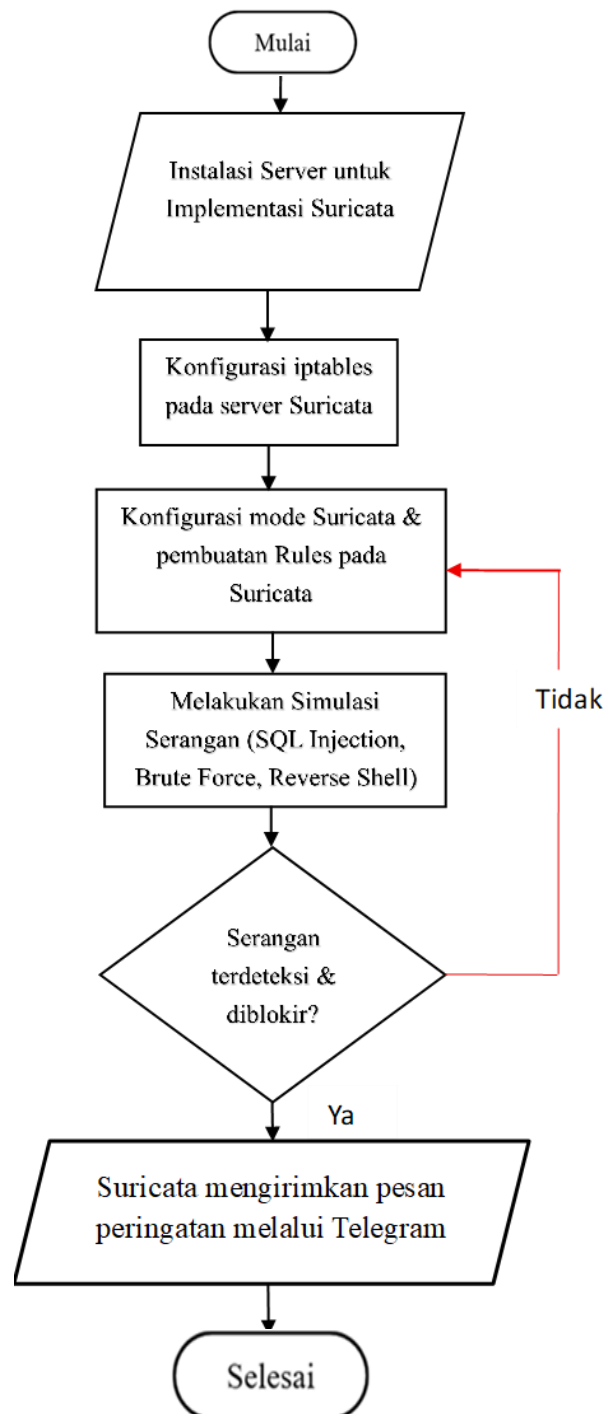
Gambar 3. 8 Diagram alir pemindaian Openvas

3.8.2 Simulasi Serangan Tahap 2

Adapun tahapan yang dilakukan pada simulasi serangan tahap 2 adalah sebagai berikut:

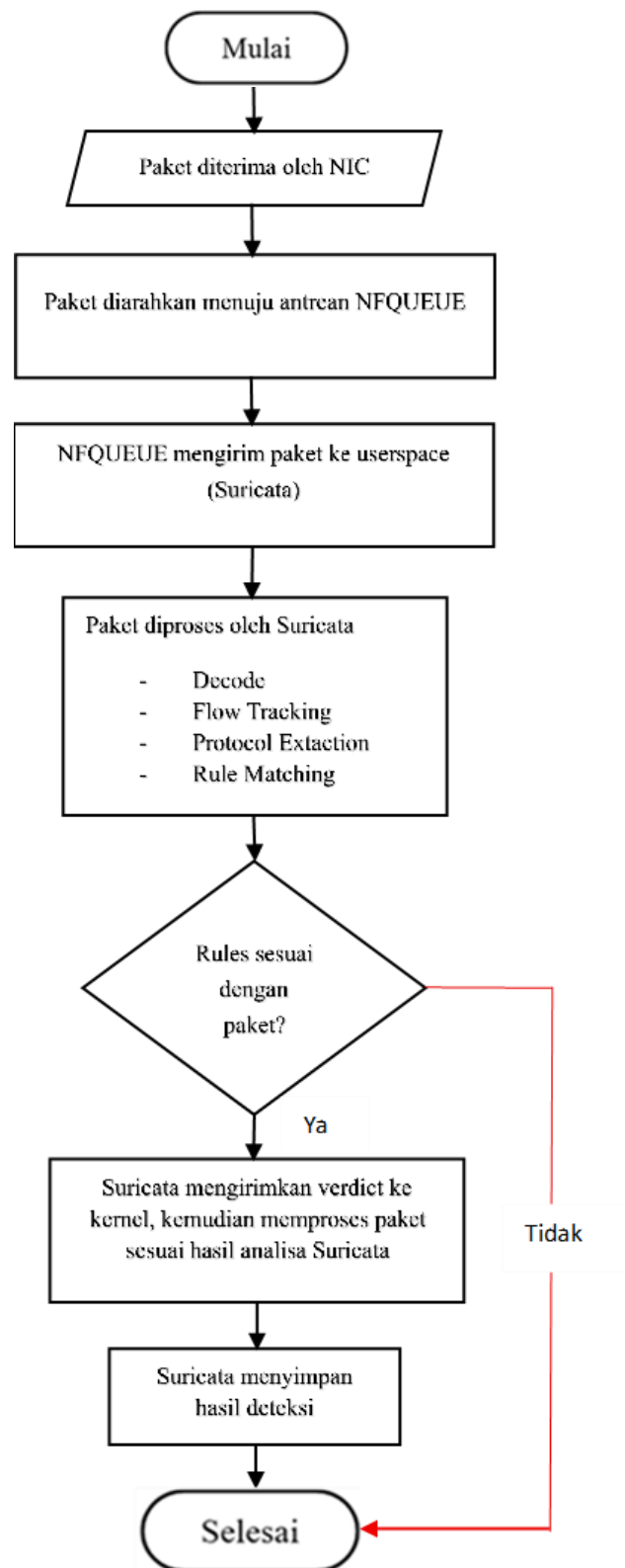
1. Melakukan instalasi *server* untuk Suricata untuk memonitoring *log* aktivitas dan juga sebagai pengaman untuk *web server*.
2. Melakukan konfigurasi *iptables* pada Suricata.
3. Melakukan konfigurasi Suricata agar *Intrusion Prevention System* (IPS) dapat memblokir jika ada paket mencurigakan yang disebabkan oleh adanya serangan dari pihak *attacker*, serta *Intrusion Detection System* (IDS) dapat mengirimkan pesan peringatan pada administrator melalui aplikasi Telegram, dan juga membuat *rules* pada Suricata.
4. Melakukan Simulasi serangan pada *web application* target menggunakan jenis serangan *SQL Injection*, *Brute Force* dan *Reverse Shell*.
5. Ketika serangan berhasil dideteksi dan diblokir, Suricata mencatat hasil deteksi dan mengirimkan pesan peringatan ke Telegram.

Pada simulasi tahap 2 ini dapat dilihat pada diagram alir simulasi pada Gambar 3.9 berikut ini:



Gambar 3. 9 Diagram Alir Simulasi Tahap 2

Pada serangan tahap kedua ini, Suricata akan melakukan proses pemindaian paket yang masuk dengan proses kerja sistem yang diawali dengan tahap inisialisasi sebelum sistem melakukan pemeriksaan terhadap paket data yang masuk. Setiap paket dianalisis untuk menentukan kesesuaian dengan aturan (rules) yang telah dikonfigurasi pada Suricata. Apabila terdapat kecocokan, maka sistem akan mengambil tindakan sesuai kebijakan yang ditetapkan. Sebaliknya, jika tidak ditemukan kecocokan, paket akan diteruskan dan proses dinyatakan selesai. Sistem kerja Suricata ditunjukkan oleh diagram alir pada Gambar 3.10.



Gambar 3. 10 Diagram alir sistem kerja Suricata

3.9 Indikator Pencapaian

Adapun indikator pencapaian pada penelitian ini adalah sebagai berikut:

1. IDS dapat mendeteksi serangan dan paket mencurigakan yang masuk ke *web server*.
2. IPS dapat memblokir paket mencurigakan yang akan masuk ke *web server*.
3. Mengirimkan pesan peringatan adanya paket mencurigakan yang akan masuk ke *web server*.

V KESIMPULAN

5.1 Kesimpulan

Penelitian ini menunjukkan bahwa *web server* target memiliki celah keamanan serius, terbukti dari keberhasilan berbagai jenis serangan seperti *SQL Injection*, *Brute Force*, dan *Reverse Shell*. Serangan *SQL Injection* berhasil mengeksploitasi parameter *login* dengan menyisipkan *payload* melalui *HTTP request* dan serangan *brute force* menggunakan *Hydra* berhasil memperoleh kredensial *login* yang valid. Serangan *reverse shell* bahkan memungkinkan penyerang memperoleh akses penuh terhadap sistem, termasuk eskalasi hak akses menjadi *root*. Setelah penerapan sistem deteksi dan pencegahan intrusi (IDS/IPS) berbasis Suricata, hasil serangan berubah signifikan di mana serangan *SQL Injection* dan *brute force* berhasil dideteksi dan diblokir melalui konfigurasi *rules* berbasis *signature*, membuktikan bahwa Suricata mampu memberikan perlindungan aktif terhadap ancaman siber jika dikonfigurasi dengan tepat.

Berdasarkan hasil evaluasi, sistem deteksi memiliki akurasi sebesar 90% dengan *False Positive Rate* (FPR) 0% yang menunjukkan tidak ada aktivitas normal yang keliru terdeteksi sebagai serangan. Nilai *False Negative Rate* (FNR) sebesar 16,67% menunjukkan adanya serangan yang lolos dari deteksi, yaitu pada skenario *multiple login failed* yang tidak memenuhi ambang batas *rule* Suricata. Temuan ini mengindikasikan bahwa walaupun Suricata efektif dalam mendeteksi sebagian besar serangan, diperlukan optimasi *rule* untuk menekan potensi *False Negative*. Secara keseluruhan, penerapan IDS/IPS Suricata terbukti mampu meningkatkan keamanan

server secara signifikan, menjaga kestabilan sistem, dan menjadi lapisan pertahanan penting terhadap ancaman siber. Walaupun Suricata berhasil mendeteksi mayoritas serangan dengan akurasi tinggi, hasil evaluasi masih menunjukkan adanya *False Negative* pada skenario *login* gagal berulang, yang menandakan bahwa tidak semua serangan dapat terdeteksi secara optimal.

.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan mengenai “Implementasi IDS/IPS Suricata untuk Deteksi dan Pencegahan Serangan pada *Web Server*”, saran yang dapat dilakukan terhadap pengembangan penelitian selanjutnya adalah:

1. Mengoptimalkan konfigurasi *rules* Suricata khususnya untuk mendeteksi serangan *multiple login failed*, agar dapat mengurangi *False Negative Rate* (FNR) yang saat ini tercatat sebesar 16,67%.
2. Melakukan pembaruan *rules* secara rutin dan menyesuaikan konfigurasi dengan karakteristik lalu lintas jaringan dan sistem yang digunakan, sehingga akurasi deteksi tetap tinggi.
3. Mengintegrasikan IDS/IPS dengan sistem log terpusat seperti *ELK Stack* untuk mendukung analisis lalu lintas secara *real-time* dan mempercepat proses investigasi insiden.
4. Menerapkan *praktik secure coding* pada pengembangan aplikasi, seperti validasi input dan penggunaan *prepared statements* untuk mencegah serangan langsung pada sisi aplikasi.

DAFTAR PUSTAKA

- [1] D. P. Putranto, J. Jayanta, and B. Hananto, "Analisis Keamanan Website Leads UPNVJ Terhadap Serangan SQL Injection dan Sniffing Attack," *Inform. J. Ilmu Komput.*, vol. 18, no. 3, p. 230, 2022, doi: 10.52958/iftk.v18i3.4690.
- [2] W. G. J. Halfond, J. Viegas, and A. Orso, "A Classification of SQL Injection Attacks and Countermeasures," Tech. Rep., Georgia Institute of Technology, Atlanta, GA, USA, 2006.
- [3] M. I. Irawan and U. Y. K. Hediyanto, "Implementasi Keamanan Jaringan Pada Cloudfri Dengan Metode Hardening," *e-Proceeding Eng.*, vol. 9, no. 2, p. 644, 2022.
- [4] S. Umrao, M. Kaur, and G. K. Gupta, "Vulnerability Assessment and Penetration Testing," *Int. J. Comput. Commun. Technol.*, vol. 3, no. 6-8, pp. 71-74, 2012, doi: 10.47893/ijcct.2016.1367.
- [5] C. Arfanudin, B. Sugiantoro, and Y. Prayudi, "Analisis Serangan Router Dengan Security Information and Event Management (Siem) Dan Implikasinya Pada Indeks Keamanan Informasi Analysis of Router Attack With Security Information and Event Management and Implications (Siem) in Information Security Index," *CyberSecurity dan Forensik Digital.*, vol. 2, no. 1, pp. 1-7, 2019.
- [6] B. A. A. Al'Aziz, P. Sukarno, and A. A. Wardana, "Blacklisted IP distribution system to handle DDoS attacks on IPS Snort based on Blockchain," in *Proc. 6th Int. Technol. Inf. Semin. (ITIS)*, 2020, pp. 41-45, doi: 10.1109/ITIS50118.2020.9320996.
- [7] D. Borbor, L. Wang, S. Jajodia, and A. Singhal, "Surviving unpatchable

- vulnerabilities through heterogeneous network hardening options,” *J. Comput. Secur.*, vol. 26, no. 6, pp. 761–789, 2018, doi: 10.3233/JCS-171106.
- [8] Y. Khera, D. Kumar, S. Sujay, and N. Garg, “Analysis and Impact of Vulnerability Assessment and Penetration Testing,” in *Proc. Int. Conf. Mach. Learn., Big Data, Cloud Parallel Comput. (COMITCon)*, 2019, pp. 525–530, doi: 10.1109/COMITCon.2019.8862224.
 - [9] F. Sirait, M. Sofyan, and K. Putra, “Implementasi Metode Vulnerability dan Hardening Pada Sistem Keamanan Jaringan,” *J. Teknol. Elektro*, vol. 9, no. 1, p. 16, 2018.
 - [10] H. A. Hendri, S. Aprianti, and Riska, “Analisa Keamanan Jaringan Komputer Menggunakan Sistem Deteksi Intrusi Shorewall,” *J. Amplifier*, vol. 13, no. 1, pp. 33–39, 2023, doi: 10.33369/jamplifier.v13i1.27888.
 - [11] Y. Maryati, “Analisis Keamanan Jaringan Komputer dengan Penerapan Teknologi Firewall dan Intrusion Detection System (IDS),” *J. Bahana Manaj. Pendidik.*, vol. 3, no. 5, 2023.
 - [12] Y. Nakagawa, Y. Kazato, and Y. Nakatani, “Inspecting intrusion prevention system signatures for false blocking using set theory,” in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2020, pp. 1–6, doi: 10.1109/ICCWorkshops49005.2020.9145103.
 - [13] M. Syani and A. M. Ropi, “Analisis dan Implementasi Network Security System Menggunakan Teknik Host-Based Intrusion Detection System (HIDS) Berbasis Cloud Computing,” in *Proc. Semin. Nas. Telekomun. dan Inform. (SELISIK), Bandung, Indonesia*, 2018, pp. 199–204.
 - [14] A. Rusli, Husain, L. Widyawati, M. Awzar, and M. Innudin, “Analisa penerapan honeypot Cowrie dan IPS untuk meningkatkan keamanan web server,” in *Proc. Semin. Nas. CORISINDO, Mataram, Indonesia*, 2025, pp. 294–300.
 - [15] Networking Security, vol. 2025, no. 3, pp. 1–44, 2025.
 - [16] Sowmyashree A. and H. S. Guruprasad, “Evaluation and Analysis of Vulnerability Scanners: Nessus and OpenVAS,” *Int. Res. J. Eng. Technol.*, vol. 7, no. 5, pp. 2068–2073, 2020.

- [17] Z. Munawar, and N. I. Putri, “Keamanan Jaringan Komputer Pada Era Big Data,” *J. Sist. Inform.*, vol. 2, no. 1, pp. 14–20, 2020.
- [18] D. Laksmiati, “Vulnerability Assessment Pada Situs WwW.Hatsehat.Com Menggunakan Openvas,” *J. Akrab Juara*, vol. 5, no. 3, pp. 240–246, 2020.
- [19] P. Mell, K. Scarfone, and S. Romanosky, “CVSS v2 Complete Documentation,” FIRST, 2007. [Online]. Available: <https://www.first.org/cvss/v2/cvss-v2-guide.pdf>
- [20] FIRST, “Common Vulnerability Scoring System Version 3.1: Specification Document,” 2019. [Online]. Available: https://www.first.org/cvss/v3-1/cvss-v31-user-guide_r1.pdf
- [21] M. Alhamed and M. M. H. Rahman, “A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions,” *Appl. Sci.*, vol. 13, no. 12, Art. no. 6986, 2023, doi: 10.3390/app13126986.
- [22] A. R. Muhammad, P. Sukarno, and A. A. Wardana, “Integrated Security Information and Event Management (SIEM) with Intrusion Detection System (IDS) for Live Analysis based on Machine Learning,” *Procedia Comput. Sci.*, vol. 217, pp. 1406–1415, 2022, doi: 10.1016/j.procs.2022.12.339.
- [23] T. Prasetyo, “Pengamanan Jaringan Komputer Dengan Intrusion Prevention System (IPS) Berbasis Sms Gateway,” *Teknologipintar.org*, vol. 2, no. 6, pp. 1–13, 2022.
- [24] F. Tanang Anugrah, S. Ikhwan, and J. Gusti, “Implementasi Intrusion Prevention System (IPS) Menggunakan Suricata Untuk Serangan SQL Injection,” *Techné J. Ilm. Elektrotek.*, vol. 21, no. 2, pp. 199–210, 2022, doi: 10.31358/techne.v21i2.320.
- [25] O. Rivaldi and N. L. Marpaung, “Penerapan Sistem Keamanan Jaringan Menggunakan Intrusion Prevention System Berbasis Suricata,” *INOVTEK Polbeng Seri Inform.*, vol. 8, no. 1, pp. 141, 2023, doi: 10.35314/isi.v8i1.3269.
- [26] A. A. Salih and A. M. Abdulazeez, “Evaluation of Classification Algorithms for Intrusion Detection System: A Review,” *J. Soft Comput. Data Min.*, vol. 2, no. 1, pp. 31–40, 2021, doi: 10.30880/jscdm.2021.02.01.004.