Instalasi & Konfigurasi Open Source SOC di Ubuntu 26.04

Panduan Praktis Membangun SOC Modern Berbasis CLI, Agent, Sensor, Log, dan Open Source Tools

Onno W. Purbo
onno@indo.net.id
onno@itts.ac.id
Twitter onnowpurbo

Institut Teknologi Tangerang Selatan (ITTS)
Dinas KOMINFO Tangerang Selatan
OnnoCenter
2026

Lisensi: Karya ini dilisensikan di bawah **Creative Commons Attribution (CC BY) 4.0 International**. Anda bebas menggunakan, membagikan, dan mengadaptasi karya ini untuk tujuan apa pun, termasuk komersial, dengan tetap mencantumkan atribusi yang layak kepada penulis.

Disclaimer: Buku ini disusun untuk tujuan edukasi, dokumentasi, dan refleksi publik berdasarkan sumber-sumber yang dianggap kredibel pada saat penulisan. Penulis telah berupaya menjaga akurasi, namun sangat mungkin masih terdapat kekurangan atau kekeliruan; karena itu, buku ini tidak dimaksudkan sebagai satu-satunya rujukan yang final, dan penulis terbuka terhadap koreksi serta masukan yang membangun

SOC bukan dashboard; SOC adalah kemampuan membaca log, menghubungkan bukti, membuat alert, melakukan triage, dan merespons secara etis.

Admin jaringan yang hebat bukan hanya menjaga koneksi tetap hidup, tetapi mampu membaca tanda bahaya, membuktikan insiden dari log, dan merespons dengan disiplin

Kata Pengantar

Keamanan siber tidak lagi menjadi isu yang hanya dimiliki oleh organisasi besar. Setiap administrator jaringan, pengelola sistem, maupun institusi pendidikan kini menghadapi tantangan yang sama: bagaimana mendeteksi ancaman lebih awal, memahami apa yang sebenarnya terjadi di infrastruktur yang dikelola, dan mengambil keputusan berdasarkan bukti yang dapat dipertanggungjawabkan.

Buku **Instalasi & Konfigurasi Open Source SOC di Ubuntu 26.04** ini disusun untuk menjawab kebutuhan tersebut secara praktis. Keistimewaan buku ini bukan terletak pada banyaknya *tools* yang dibahas, melainkan pada pendekatan yang menempatkan pembaca sebagai seorang analis yang berpikir secara sistematis: membaca log, menghubungkan *evidence*, melakukan *triage*, membangun *timeline*, dan merespons insiden secara terukur.

Melalui pemanfaatan **Ubuntu 26.04 LTS**, **Wazuh**, **Suricata**, **Zeek**, **OpenSearch**, **MISP**, **DFIR-IRIS**, **Shuffle**, serta berbagai utilitas *command line*, pembaca diajak membangun sebuah **SOC modern berbasis open source** yang dapat dijalankan di laboratorium, kampus, sekolah, maupun organisasi dengan sumber daya terbatas. Seluruh materi dirancang dengan orientasi praktik sehingga setiap bab menghasilkan artefak nyata berupa konfigurasi, log, alert, laporan, dan dokumentasi yang dapat diverifikasi.

Semoga buku ini dapat membantu melahirkan lebih banyak administrator jaringan dan praktisi keamanan yang tidak hanya mampu mengoperasikan perangkat, tetapi juga mampu **membuktikan, menganalisis, dan mengambil keputusan berdasarkan data serta bukti digital yang sahih**.

Jakarta, Juni 2026
Onno W. Purbo

Executive Summary

Buku ini dirancang untuk menjembatani kesenjangan antara teori keamanan siber dan implementasi nyata ***Security Operations Center (SOC)*** berbasis *open source* yang dapat dibangun, dioperasikan, dan dikembangkan secara mandiri oleh administrator jaringan. Berangkat dari prinsip bahwa **SOC bukan sekadar kumpulan perangkat lunak atau dashboard yang menampilkan alert**, tetapi sebuah **disiplin operasional untuk membaca sinyal ancaman, menghubungkan bukti digital, memvalidasi indikasi serangan, dan mengambil keputusan yang cepat, terukur, serta etis**, buku ini menempatkan pembaca pada peran seorang analis dan operator SOC yang sesungguhnya. **Ubuntu 26.04 LTS** dipilih sebagai fondasi karena menawarkan stabilitas, dukungan keamanan jangka panjang hingga tahun 2031, serta ekosistem yang matang untuk membangun lingkungan pemantauan keamanan modern. Di atas fondasi tersebut, pembaca akan membangun **ekosistem SOC end-to-end** menggunakan **Wazuh** sebagai platform *SIEM/XDR*, **Security Onion** sebagai platform sensor jaringan, **Suricata** untuk deteksi berbasis aturan, **Zeek** untuk analisis metadata dan protokol jaringan, **OpenSearch** untuk analitik log dan pencarian data, **MISP** untuk *threat intelligence*, **DFIR-IRIS** atau *TheHive* untuk *case management*, serta **Shuffle** untuk otomasi dan orkestrasi respons. Yang membedakan buku ini dari kebanyakan buku SOC adalah **orientasinya yang sangat praktis, terukur, dan berbasis hasil**. Setiap bab menghasilkan **artefak teknis yang dapat diverifikasi**, mulai dari konfigurasi server, layanan yang aktif, *agent* yang terhubung, sensor yang mampu membaca trafik, log yang dapat dianalisis melalui *command line interface (CLI)*, *alert* yang dapat difilter dan dikorelasikan, hingga laporan insiden yang terdokumentasi dengan baik. Dengan pendekatan **hands-on**, pembaca tidak hanya belajar memasang *tools*, tetapi juga membangun kemampuan untuk mengubah data menjadi bukti, bukti menjadi pemahaman, dan pemahaman menjadi tindakan. Pada akhirnya, tujuan utama buku ini adalah **membentuk pola pikir investigatif, analitis, dan berbasis bukti**, sehingga administrator jaringan mampu bertransformasi menjadi praktisi keamanan yang siap menghadapi ancaman siber modern secara profesional dan bertanggung jawab.

Daftar Isi

Kata Pengantar **4**
Executive Summary **5**
Daftar Isi **6**
Bab 1 — Fondasi SOC Open Source di Ubuntu 26.04 ..... **25**
1.1 Mengapa Admin Jaringan Perlu SOC 25
Posisi Strategis Admin Jaringan 26
Ancaman yang Sering Terlihat dari SOC 27
1.2 SOC sebagai Sistem Kerja, Bukan Sekadar Tools 28
Contoh Sederhana 28
Cara Berpikir SOC 29
Perbedaan Operator Tools dan Analis SOC 30
1.3 Komponen Minimum SOC 31
1. SOC Server 31
2. Agent Endpoint 32
3. Network Sensor 32
4. Log Collector 33
5. SIEM/XDR 33
6. Dashboard 34
7. Threat Intelligence 34
8. Case Management 35
9. Automation / SOAR 35
1.4 Peta Tools Open Source 36
Tools Utama 36
Peta Peran Tools 37
Prinsip Pemilihan Tools 37
1.5 Arsitektur Mini SOC 38
Topologi Minimum 38
Topologi Sensor Jaringan 39
Arsitektur Lab Minimum 39
Jalur Data 40
Prinsip Desain Arsitektur 40
1.6 Golden Rule SOC 41
Aturan 1 — Jangan Percaya Alert Sebelum Membaca Log 41
Aturan 2 — Jangan Respons Sebelum Punya Bukti 42
Aturan 3 — Jangan Otomatisasi Sebelum Paham Risikonya 42
Golden Rule Tambahan 43
1.7 Hands-on: Membuat Desain SOC di File Teks 44
1.7.1 Membuat Struktur Folder Kerja 44
1.7.2 Membuat File Arsitektur SOC 46
1.7.3 Membaca Ulang File Desain 49
1.7.4 Membuat Checklist Awal SOC 50
1.7.5 Membuat Template Timeline 51

1.7.6 Membuat Template Incident Report..... 52
1.7.7 Membuat Catatan Desain dalam Format Markdown.....53
1.7.8 Validasi Struktur Dokumen..... 54
1.7.9 Praktik Python Ringan: Membuat Ringkasan Desain.....54
1.8 Kesalahan Umum Pemula SOC.....56
1. Terlalu Cepat Instalasi.....56
2. Terlalu Percaya Dashboard.....56
3. Tidak Mencatat Evidence..... 57
4. Mencampur Lab dan Produksi..... 57
5. Otomasi Terlalu Cepat.....57
1.9 Checklist Kompetensi Bab 1..... 58
1.10 Rangkuman Bab.....58
Output Bab..... 59
Bab 2 — Instalasi Ubuntu 26.04 untuk Server SOC..... 60
2.1 Spesifikasi Minimum Lab.....60
Minimum Pribadi.....60
Ideal untuk Praktik Serius.....61
Rekomendasi Pembagian VM.....62
Catatan Strategis.....62
2.2 Instalasi Ubuntu Server 26.04.....63
Parameter Instalasi..... 63
Hostname..... 63
User Admin.....64
Partisi..... 64
Timezone.....64
IP Static..... 65
2.3 CLI Update dan Hardening Awal.....66
Cek Versi OS.....67
Hardening Awal Minimum.....67
2.4 Setting IP Static dengan Netplan..... 68
Cek Interface..... 68
Edit File Netplan..... 68
Terapkan Konfigurasi.....69
Kesalahan Umum Netplan..... 69
Backup Sebelum Mengubah Jaringan..... 70
2.5 SSH Aman untuk Admin SOC.....71
Instalasi SSH Server..... 71
Backup Konfigurasi SSH..... 71
Edit Konfigurasi SSH.....71
Validasi Konfigurasi..... 72
Uji dari Mesin Admin..... 72
Catatan Kritis SSH..... 72
2.6 Firewall Dasar dengan UFW..... 73
Instalasi UFW..... 73

Aturan Firewall Minimum....74
Membatasi SSH dari IP Admin....74
Cek Log UFW....74
2.7 Instalasi Docker dan Compose untuk Tool Pendukung....75
Catatan Pilihan Instalasi....75
Instalasi....75
Uji Container Sederhana....76
Catatan Keamanan Docker....76
2.8 Struktur Direktori SOC Server....77
Fungsi Direktori....77
Struktur Folder Home Lab....78
Membuat README Server....78
2.9 Backup Konfigurasi Awal....79
Backup dengan Tanggal....79
Membuat Manifest Backup....80
Menghitung Hash File Backup....80
2.10 Hands-on Tambahan: Inventaris Server SOC dengan Python....81
Makna Script....82
2.11 Checklist Validasi Akhir Bab 2....83
Sistem....83
Jaringan....83
SSH....83
Firewall....83
Container Pendukung....84
Dokumentasi....84
2.12 Kesalahan Umum Bab 2....85
1. Tidak Membuat IP Static....85
2. Mengaktifkan Firewall Sebelum Membuka SSH....85
3. Salah Format YAML Netplan....85
4. Tidak Membuat Backup....86
5. Menjalankan Semua Service Sekaligus....86
2.13 Rangkuman Bab....87
Output Bab....87
Bab 3 — Wazuh Server: SIEM/XDR, Agent, dan Alert....88
3.1 Arsitektur Wazuh....88
Alur Kerja Dasar Wazuh....88
Wazuh Manager....89
Wazuh Indexer....90
Wazuh Dashboard....90
Wazuh Agent....91
Rules....91
Decoders....92
Alerts....92
API....93

3.2 Instalasi Wazuh Single Node di Ubuntu 26.04 94
Catatan Penting Sebelum Instalasi 94
Masuk ke Folder Kerja 95
Mengunduh Installer Resmi Wazuh 95
Menjalankan Instalasi All-in-One 95
Simpan Output Instalasi 96
Catatan Keamanan Dashboard 96
3.3 Cek Service Wazuh 97
Cek Wazuh Manager 97
Cek Wazuh Indexer 97
Cek Wazuh Dashboard 97
Cek Port Aktif 98
Cek Semua Service Wazuh 98
Cek Resource 98
Restart Service Jika Perlu 99
3.4 Lokasi File Penting Wazuh 99
Direktori Utama 99
Konfigurasi 99
Log Wazuh 100
Ruleset Bawaan 100
Rule Lokal 100
Decoder Lokal 100
Membuat Catatan Lokasi File 101
3.5 Akses Log Wazuh via CLI 102
Membaca Log Operasional 102
Membaca Alert Format Teks 102
Membaca Alert Format JSON 102
Membaca JSON dengan jq 102
Filter Alert Berdasarkan Level 103
Ambil Field Penting 103
Cari Alert dari Agent Tertentu 103
Cari Source IP 103
Simpan Alert Level Tinggi ke Evidence 104
3.6 Menambahkan Wazuh Agent Linux 105
Di Endpoint: Cek Status Agent 105
Catatan Instalasi Agent 105
Di Server: Cek Daftar Agent 106
Manage Agents 106
Troubleshooting Agent Pertama 107
3.7 Membaca Agent dari CLI dan Dashboard 108
Agent ID 108
Hostname 108
IP Address 109
Status Active / Disconnected 109

Last Keep Alive..... 109
Membaca dari Dashboard..... 110
3.8 Wazuh API dengan curl..... 111
Autentikasi API.....111
Query Informasi Manager.....112
Query Daftar Agent.....112
Ambil Field Agent Penting..... 112
Ekspor Daftar Agent ke JSON.....113
Catatan Token.....113
Script Python Ringan untuk Query Agent.....113
3.9 Template Catatan Instalasi Wazuh..... 115
Backup Catatan dan Konfigurasi..... 117
Hash Backup..... 117
3.10 Hands-on: Membuat Alert Pertama.....118
3.11 Checklist Validasi Akhir Bab 3..... 119
Service.....119
File dan Log.....119
Agent..... 119
API.....120
Dokumentasi..... 120
3.12 Kesalahan Umum Bab 3..... 121
1. Menganggap Instalasi Selesai Karena Dashboard Terbuka..... 121
2. Tidak Membaca alerts.json.....121
3. Agent Terhubung Tetapi Tidak Dipahami.....121
4. Token API Dibocorkan.....121
5. Tidak Backup Sebelum Custom Rule.....121
6. Membuka Dashboard Terlalu Luas.....122
3.13 Rangkuman Bab.....122
Output Bab..... 123
Bab 4 — Mengelola Agent dan Endpoint Linux..... 124
4.1 Agent sebagai Mata dan Telinga SOC..... 124
Data Penting dari Agent..... 125
Prinsip Penting..... 125
4.2 Instalasi Agent di Endpoint Ubuntu 26.04..... 126
Prasyarat Endpoint.....126
Konsep Manager IP.....127
Contoh Instalasi Agent Linux.....127
Jika Menggunakan Repository.....127
Catatan Instalasi Endpoint..... 128
4.3 File Konfigurasi Agent..... 129
Backup Sebelum Edit.....129
Membuka Konfigurasi.....129
Restart Agent..... 130
Validasi Manager IP.....130

Catatan Praktis....130
4.4 Cek Status Agent....131
Cek dari Endpoint....131
Cek dari Server Wazuh....131
Tabel Catatan Agent....132
Interpretasi Status....132
4.5 Grouping Agent....133
Mengapa Grouping Penting....133
Contoh Desain Group....134
Membuat Group....134
Menambahkan Agent ke Group....134
Konfigurasi Terpusat per Group....135
Prinsip Grouping....135
4.6 Mengaktifkan Log Collection Endpoint....136
Cek Log Lokal di Endpoint....136
Menambahkan Log Collection....136
Uji Log Autentikasi....137
Simpan Evidence....137
Prinsip Log Collection....138
4.7 File Integrity Monitoring....139
Mengapa File Ini Penting?....139
Backup Konfigurasi Agent....139
Konfigurasi FIM Lokal....140
Catatan Kritis FIM....140
4.8 Simulasi Perubahan File....141
Cek Alert di Server....141
Simpan Evidence FIM....141
Membuat Catatan FIM....142
Uji File Penting dengan Aman....142
4.9 Troubleshooting Agent....143
Alur Troubleshooting Cepat....143
1. Agent Disconnected....143
2. Salah IP Manager....144
3. Firewall Blocking....144
4. Waktu Tidak Sinkron....145
5. Service Mati....145
6. DNS Gagal....145
7. Agent Tidak Muncul di Server....146
4.10 Hands-on Python: Audit Status Agent dari File JSON....147
Penjelasan Singkat....147
4.11 Checklist Validasi Akhir Bab 4....148
Agent....148
Log Collection....148
Grouping....148

FIM..... 148
Troubleshooting..... 148
4.12 Kesalahan Umum Bab 4..... 149
1. Agent Terpasang tetapi IP Manager Salah..... 149
2. Firewall Server Menolak Agent..... 149
3. Terlalu Banyak File Dipantau FIM..... 149
4. Semua Endpoint Masuk Group Default..... 150
5. Tidak Menyimpan Evidence..... 150
4.13 Rangkuman Bab..... 151
Output Bab..... 151
Bab 5 — Sensor Jaringan: Security Onion, Suricata, Zeek..... 152
5.1 Mengapa Agent Saja Tidak Cukup..... 152
Contoh Perbedaan Pandangan..... 153
Prinsip Korelasi..... 154
5.2 Konsep Network Sensor..... 155
Management Interface..... 155
Monitoring Interface..... 156
TAP / SPAN / Mirror Port..... 156
Packet Capture..... 157
Alert..... 157
Flow..... 158
Metadata..... 158
5.3 Security Onion sebagai Platform Sensor..... 159
Peran Security Onion..... 159
Suricata di Security Onion..... 160
Zeek di Security Onion..... 160
Hunt..... 160
Alerts..... 161
Dashboards..... 161
PCAP..... 162
Sensor Deployment..... 162
5.4 Suricata untuk Deteksi Rule-Based..... 163
File Penting Suricata..... 163
Lokasi Umum File..... 163
Cek Service Suricata..... 164
Membaca EVE JSON..... 164
Membaca fast.log..... 165
Membaca suricata.yaml..... 165
Catatan Penting Rule..... 165
5.5 Zeek untuk Metadata Jaringan..... 166
Lokasi Log Zeek..... 166
conn.log..... 167
dns.log..... 167
http.log..... 168

ssl.log..... 168
files.log..... 168
notice.log..... 169
Prinsip Membaca Zeek.....169
5.6 Simulasi Port Scan di Lab Tertutup..... 170
Topologi Lab.....170
Instal Nmap di Mesin Lab.....170
Jalankan Scan Ringan..... 171
Cek Suricata.....171
Cek Zeek.....171
Catat Hasil.....172
5.7 Membaca Alert Network.....173
Field Penting Suricata..... 173
Interpretasi Cepat.....174
Jangan Langsung Menyimpulkan.....174
Pivot ke Zeek.....175
Pivot ke Endpoint..... 175
5.8 Mengambil PCAP untuk Evidence..... 176
Capture Trafik Target.....176
Baca PCAP..... 176
Capture dengan Batas Jumlah Paket.....176
Simpan ke Folder Evidence..... 177
Catatan Chain of Custody Sederhana.....177
Prinsip PCAP.....177
5.9 Hands-on Python: Ringkas Alert Suricata EVE JSON..... 178
Penjelasan Singkat.....179
Simpan Output..... 179
5.10 Template Catatan Investigasi Network Alert.....180
5.11 Checklist Validasi Akhir Bab 5..... 182
Sensor..... 182
Security Onion / Sensor Platform..... 182
Suricata..... 182
Zeek..... 182
PCAP.....182
Dokumentasi..... 183
5.12 Kesalahan Umum Bab 5..... 184
1. Sensor Tidak Melihat Trafik.....184
2. Hanya Membaca Suricata, Tidak Membaca Zeek.....184
3. Tidak Menyimpan PCAP Penting..... 184
4. Tidak Menyamakan Waktu..... 185
5. Terlalu Banyak Rule Tanpa Tuning.....185
6. Melakukan Scan ke Sistem Publik..... 185
5.13 Rangkuman Bab.....186
Output Bab..... 187

Bab 6 — Akses Log via CLI: journalctl, syslog, auditd, Wazuh, Suricata, Zeek..... 188
6.1 Prinsip Membaca Log.....188
Alur Berpikir Membaca Log..... 189
Prinsip 1 — Mulai dari Waktu..... 189
Prinsip 2 — Jangan Hanya Cari Alert.....189
Prinsip 3 — Gunakan Filter Sederhana.....190
Prinsip 4 — Simpan Evidence.....190
6.2 Log Sistem Ubuntu.....191
Membaca Semua Journal..... 191
Membaca Error Terbaru..... 191
Membaca Log SSH..... 191
Membaca Log Wazuh Agent..... 192
Membaca Log 1 Jam Terakhir..... 192
Membaca Log Hari Ini..... 192
Kombinasi Waktu.....192
Menyimpan Log Systemd.....192
Tips Praktis journalctl..... 193
6.3 Log Autentikasi.....194
Membaca Real-Time..... 194
Mencari Login Gagal..... 194
Mencari Login Berhasil.....194
Mencari Aktivitas sudo..... 195
Mencari User Tidak Valid.....195
Menghitung IP Login Gagal.....195
Versi Lebih Aman dengan grep IP.....195
6.4 Log Kernel dan Network.....196
Membaca Kernel Ring Buffer..... 196
Membaca Log Kernel dari Journal..... 196
Cek IP Address..... 196
Cek Routing.....196
Cek Port Aktif..... 197
Capture Trafik Sementara..... 197
Simpan PCAP Kecil.....197
6.5 auditd untuk Monitoring Aktivitas Sistem.....198
Instalasi auditd..... 198
Mencari Event Login.....198
Mencari Event Autentikasi.....198
Membaca Audit Log Langsung.....198
Ringkasan Audit..... 199
Prinsip auditd.....199
6.6 Membuat Rule auditd Sederhana.....200
Membuat Rule Sementara..... 200
Cek Rule.....200
Uji dengan Perubahan Aman..... 200

Mencari Rule /etc/passwd..... 200
Membuat Report File.....201
Simpan Evidence..... 201
Menghapus Rule Sementara.....201
Catatan Persisten.....201
6.7 Membaca Wazuh Alert JSON..... 202
Membaca Real-Time..... 202
Filter Alert Level Tinggi.....202
Ambil Field Penting..... 202
Cari Alert Agent Tertentu..... 202
Cari IP Mencurigakan.....203
Simpan Alert Level Tinggi.....203
Ringkas Wazuh dengan Python..... 203
6.8 Membaca Suricata Eve JSON.....205
Filter Alert..... 205
Ambil Field Penting..... 205
Cari IP Mencurigakan.....205
Cari Alert Severity Tertentu..... 206
Simpan Evidence Suricata..... 206
Ringkasan Source IP.....206
Ringkasan Signature..... 206
6.9 Membaca Zeek Log.....207
Membaca conn.log..... 207
Membaca dns.log..... 208
Membaca http.log.....208
Membaca ssl.log..... 208
Membaca notice.log..... 208
Simpan Evidence Zeek..... 209
Jika zeek-cut Tidak Ada..... 209
6.10 Korelasi Manual dengan CLI.....210
Skenario..... 210
1. Cari IP di Log Autentikasi.....210
2. Cari IP di Wazuh.....210
3. Cari IP di Suricata.....211
4. Cari IP di Zeek.....211
5. Buat Timeline.....212
6. Buat Ringkasan dengan Python.....213
Prinsip Korelasi Manual.....214
6.11 Latihan Mini: Investigasi Login Gagal dan Port Scan.....215
Langkah 1 — Siapkan Folder.....215
Langkah 2 — Cari Login Gagal.....215
Langkah 3 — Cari Alert Wazuh.....215
Langkah 4 — Cari Alert Suricata.....215
Langkah 5 — Cari Koneksi Zeek.....215

Langkah 6 — Buat Kesimpulan....216
6.12 Checklist Validasi Akhir Bab 6....217
Log Sistem.... 217
Log Autentikasi....217
Kernel dan Network....217
auditd....217
Wazuh.... 217
Suricata.... 218
Zeek.... 218
Korelasi.... 218
6.13 Kesalahan Umum Bab 6.... 219
1. Membaca Log Tanpa Rentang Waktu.... 219
2. Hanya Mengandalkan Dashboard....219
3. Grep Tanpa Memahami Format.... 219
4. Membaca JSON Tanpa jq.... 219
5. Tidak Menyimpan Evidence.... 220
6. Tidak Mencocokkan Sumber Lain.... 220
7. Lupa Timezone....220
6.14 Rangkuman Bab....221
Output Bab.... 222
Bab 7 — OpenSearch dan Dashboard SOC: Query, Index, Alert....223
7.1 Mengapa Log Perlu Diindeks.... 223
Log Mentah vs Log Terindeks.... 224
Mengapa Penting untuk SOC....224
7.2 Instalasi OpenSearch dan OpenSearch Dashboards.... 225
Dua Skenario Instalasi.... 225
Komponen yang Harus Dipahami.... 226
Port Penting....226
Password Admin.... 227
7.3 Cek Service OpenSearch....228
Cek API OpenSearch....228
Cek Port.... 228
Cek Health Cluster.... 229
Cek Log Service.... 229
Catatan Validasi....230
7.4 Index, Field, Mapping....231
Index....231
Field....231
Mapping....232
Cek Index.... 232
Cek Mapping.... 233
Cek Satu Dokumen Alert....233
7.5 Query Dasar dengan curl.... 234
Melihat Daftar Index.... 234

Melihat Cluster Health....234
Melihat Informasi Cluster....234
Query Sederhana ke Wazuh Alerts....234
Query dengan Body JSON....235
Ambil Field Penting dengan jq....235
Membuat Folder Query....236
7.6 Query Alert Severity Tinggi....237
Query Rule Level >= 10....237
Query Alert 1 Jam Terakhir....238
Query Level Tinggi dalam 1 Jam....239
Export Evidence....239
Ringkas dengan Python....240
7.7 Dashboard Minimum SOC....241
Prinsip Dashboard Minimum....241
Panel 1 — Alert per Severity....242
Panel 2 — Top Agent....242
Panel 3 — Top Source IP....243
Panel 4 — Failed Login....243
Panel 5 — FIM Alert....243
Panel 6 — Suricata Alert....244
Panel 7 — Timeline Incident....244
Catatan Desain Dashboard....245
7.8 Membuat Alert Berbasis Query....246
Prinsip Alert Berbasis Query....246
Contoh 1 — Rule Level >= 10....247
Contoh 2 — Login Gagal > 10 Kali....247
Contoh 3 — Perubahan /etc/passwd....248
Contoh 4 — Port Scan Terdeteksi....248
Contoh 5 — Koneksi ke IOC....248
Template Catatan Alert Query....249
Jangan Terlalu Agresif....249
7.9 Hands-on Python: Query OpenSearch dari Script....250
Catatan Keamanan Script....251
7.10 Troubleshooting OpenSearch dan Dashboard....252
Masalah 1 — Service Tidak Aktif....252
Masalah 2 — Port 9200 Tidak Terbuka....252
Masalah 3 — Autentikasi Gagal....252
Masalah 4 — Index Tidak Muncul....253
Masalah 5 — Dashboard Kosong....253
Masalah 6 — Query Lambat....254
7.11 Checklist Validasi Akhir Bab 7....255
Service....255
Index....255
Query....255

Dashboard....255
Alert Berbasis Query....256
Dokumentasi....256
7.12 Kesalahan Umum Bab 7....257
1. Dashboard Dibuat Sebelum Field Dipahami....257
2. Query Tidak Membatasi Waktu....257
3. Membuka Port Indexer ke Jaringan Luas....257
4. Semua Alert Dianggap Penting....257
5. Tidak Menyimpan Query....257
6. Dashboard Terlalu Ramai....258
7.13 Rangkuman Bab....258
Output Bab....259
Bab 8 — Deteksi, Rule, Decoder, FIM, Vulnerability, dan Active Response....260
8.1 Deteksi Adalah Inti SOC....260
Deteksi adalah jantung SOC....260
Dari Log ke Deteksi....261
Deteksi Bawaan vs Deteksi Lokal....261
Deteksi yang Baik....262
Kesalahan Umum....262
8.2 Wazuh Rule dan Decoder....263
Rule....263
Decoder....264
Backup Sebelum Edit....264
Edit Rule Lokal....264
Edit Decoder Lokal....264
Cek Error Setelah Edit....265
8.3 Menguji Rule dengan Logtest....266
Menjalankan Logtest....266
Contoh Rule untuk Log Aplikasi Lokal....267
Catatan Penting ID Rule....268
8.4 Custom Rule Login Gagal....269
Konsep Rule Login Gagal....269
Rule Dasar Berbasis Rule Bawaan SSH....269
Edit Rule....270
Uji dengan Logtest....270
Simpan Evidence....270
Catatan Tuning....271
8.5 FIM untuk File Penting....272
Mengapa File Ini Penting?....272
Backup Konfigurasi Agent....272
Edit Konfigurasi FIM....273
Uji Aman dengan File Dummy....273
Simpan Evidence FIM....274
Catatan FIM....274

8.6 Vulnerability Detection....275
Konsep Dasar....275
Inventaris Paket....275
CVE....276
Severity....276
False Positive....276
Query Vulnerability dari Alert....277
Template Prioritas Patch....277
8.7 Deteksi Malware Sederhana dengan YARA....278
Prinsip Aman....278
Instal YARA....278
Buat Rule YARA Dummy....279
Buat File Dummy....279
Jalankan Scan YARA....279
Simpan Evidence....280
Catatan Penting....280
8.8 Active Response....281
Mengapa Harus Hati-Hati?....281
Desain Aman untuk Lab....282
Contoh Script Active Response Aman....282
Cek Log Active Response....282
Prinsip Konfigurasi....283
Catatan Audit....283
8.9 Simulasi SSH Brute Force di Lab....284
Prasyarat....284
Cara Aman: Login Gagal Manual....284
Cek Log Endpoint....284
Cek Alert Wazuh....285
Simpan Evidence....285
Buat Catatan Simulasi....285
Etika....286
8.10 Export Alert untuk Laporan....287
Siapkan Folder Evidence....287
Export Alert Level Tinggi....287
Export Alert FIM....287
Export Alert SSH....287
Ringkas Alert dengan Python....287
Buat Hash Evidence....288
Template Laporan Singkat....289
8.11 Checklist Validasi Akhir Bab 8....290
Rule dan Decoder....290
Custom Login Gagal....290
FIM....290
Vulnerability Detection....290

YARA....291
Active Response....291
Export Laporan....291
8.12 Kesalahan Umum Bab 8....292
1. Membuat Rule Tanpa Logtest....292
2. Mengubah Rule Bawaan....292
3. Level Alert Terlalu Tinggi....292
4. FIM Terlalu Luas....292
5. Vulnerability Alert Dianggap Pasti Eksploitable....292
6. Memakai Malware Asli untuk Latihan....293
7. Active Response Terlalu Agresif....293
8. Evidence Tidak Disimpan....293
8.13 Rangkuman Bab....293
Output Bab....294
Bab 9 — Incident Handling, Case Management, MISP, dan SOAR....295
9.1 Alert Bukan Incident....295
Contoh Alert yang Belum Tentu Incident....295
Kapan Alert Naik Menjadi Incident?....296
Prinsip Kritis....296
9.2 Triage Alert....297
Checklist Triage....297
Template Triage Cepat....297
Langkah Praktis CLI....298
Hitung Percobaan Login Gagal....298
Cari Login Sukses Setelah Gagal....298
Keputusan Triage....299
9.3 Case Management dengan DFIR-IRIS atau TheHive....300
Struktur Case Minimum....301
Membuat Case Manual di File Teks....302
DFIR-IRIS atau TheHive?....303
9.4 Threat Intelligence dengan MISP....304
Istilah Penting MISP....304
Contoh IOC....304
Mengapa IOC Perlu Konteks?....305
Template IOC Lokal....305
Python: Cek IOC terhadap Evidence....306
Export JSON untuk Case....306
9.5 Mapping MITRE ATT&CK....307
Mengapa Mapping Penting?....307
Contoh Mapping Sederhana....307
Mapping untuk SSH Brute Force....308
Template Mapping....308
Prinsip Mapping....309
9.6 SOAR dengan Shuffle....310

Prinsip SOAR Aman....310
Workflow Minimum....310
Field yang Diparsing....311
Contoh Payload Alert Ringkas....311
Audit Trail....312
Template Audit Trail....312
Batas Aman....313
9.7 Playbook “SSH Brute Force Detected”....314
Tujuan Playbook....314
Langkah 1 — Alert Masuk....314
Langkah 2 — Ambil Source IP....315
Langkah 3 — Cek Jumlah Percobaan....315
Langkah 4 — Cek User Target....315
Langkah 5 — Cek Login Sukses....315
Langkah 6 — Cek Bukti Network....315
Langkah 7 — Buat Case....316
Langkah 8 — Minta Approval....317
Langkah 9 — Blokir di Lab Jika Disetujui....317
Langkah 10 — Catat Hasil....317
9.8 Hands-on Python: Membuat Draft Case dari Alert JSON....318
Makna Script....319
9.9 Checklist Validasi Akhir Bab 9....320
Triage....320
Case Management....320
MISP / Threat Intelligence....320
MITRE ATT&CK....320
SOAR....321
Playbook SSH Brute Force....321
9.10 Kesalahan Umum Bab 9....322
1. Semua Alert Dianggap Incident....322
2. Case Tanpa Timeline....322
3. IOC Ditelan Mentah-Mentah....322
4. MITRE Mapping Dipaksakan....322
5. SOAR Terlalu Agresif....323
6. Response Tanpa Audit Trail....323
9.11 Rangkuman Bab....324
Output Bab....325
Bab 10 — Proyek Akhir: Mini SOC Ubuntu 26.04 Siap Operasi....326
10.1 Spesifikasi Mini SOC....326
Spesifikasi Wajib....327
Spesifikasi Opsional....327
Topologi Minimum....328
Topologi Ideal....328
Tabel Aset Proyek....329

Struktur Folder Proyek Akhir..... 329
10.2 Skenario Uji..... 331
Skenario 1 — Login Gagal Berulang..... 331
Skenario 2 — Perubahan File Penting..... 332
Skenario 3 — Port Scan di Lab..... 333
Skenario 4 — Service Mati..... 334
Skenario 5 — Akses Web Mencurigakan..... 335
10.3 Evidence yang Harus Dikumpulkan..... 336
Prinsip Evidence..... 336
Struktur Evidence..... 336
1. Screenshot Dashboard..... 337
2. Alert JSON..... 337
3. Log CLI..... 338
4. Timeline..... 338
5. Case Note..... 339
6. IOC..... 340
7. Tindakan Respons..... 340
8. Lesson Learned..... 341
9. Hash Evidence..... 341
10.4 KPI Mini SOC..... 342
1. MTTD..... 342
2. MTTR..... 342
3. Jumlah Alert..... 342
4. Jumlah False Positive..... 343
5. Jumlah Incident Closed..... 343
6. Jumlah Rule Aktif..... 343
7. Jumlah Endpoint Monitored..... 343
Template KPI..... 344
Python: Membuat Ringkasan KPI..... 344
10.5 Format Laporan Akhir..... 346
Template Laporan Akhir..... 346
Ringkasan Arsitektur..... 348
Daftar Tools..... 348
Topologi Jaringan..... 349
Konfigurasi Utama..... 349
Hasil Alert..... 349
Lesson Learned..... 350
10.6 Rubrik Penilaian..... 351
Tabel Rubrik..... 351
Detail Penilaian..... 352
1. Instalasi Berhasil — 20%..... 352
2. Agent dan Sensor Berjalan — 15%..... 352
3. Akses Log CLI — 15%..... 353
4. Rule dan Deteksi — 20%..... 353

5. Incident Report — 15% 354
6. Dokumentasi dan Etika — 15% 354
Formula Nilai 354
10.7 Pesan Penutup 355
10.8 Checklist Proyek Akhir 356
Infrastruktur 356
Wazuh 356
Sensor Opsional 356
Skenario Uji 357
Evidence 357
Laporan 357
10.9 Template Manifest Proyek 358
10.10 Perintah Final Validasi 360
Cek Folder 360
Cek Agent 360
Cek Alert 360
Cek FIM 360
Cek Auth Log 360
Cek Dashboard / Index 360
Cek Evidence Hash 360
Cek Laporan 360
10.11 Kesalahan Umum Proyek Akhir 361
1. Terlalu Fokus Instalasi, Lupa Evidence 361
2. Dashboard Ada, CLI Tidak Bisa 361
3. Tidak Ada Case Incident 361
4. Rule Tidak Diuji 361
5. Tidak Ada Timeline 361
6. Tidak Ada Lesson Learned 361
7. Simulasi Tidak Etis 362
10.12 Rangkuman Bab 362
Output Bab 363
Lampiran A — Cheatsheet CLI Ubuntu SOC 364
A.1 systemctl — Cek dan Kelola Service 364
A.2 journalctl — Membaca Log Systemd 364
A.3 grep — Mencari Pola di Log 365
A.4 awk — Ambil Kolom Tertentu 365
A.5 jq — Membaca JSON 366
A.6 curl — Query API dan Service 366
A.7 ss — Cek Port Aktif 367
A.8 tcpdump — Capture Trafik 367
A.9 ip — Cek Jaringan Linux 368
A.10 ufw — Firewall Host 368
A.11 auditctl — Rule Audit Linux 369
A.12 ausearch — Cari Event Audit 369

Lampiran B — Port Penting SOC....370
Contoh UFW Minimum untuk Lab....371
Prinsip Port SOC....372
Lampiran C — Lokasi Log Penting....373
C.1 Ubuntu....373
C.2 Wazuh....374
C.3 Suricata....375
C.4 Zeek....376
Lampiran D — Template Incident Report....377
D.1 Template Tabel Incident Report....377
D.2 Template Markdown....377
D.3 Prinsip Incident Report....378
Lampiran E — Template Checklist Instalasi....379
E.1 Checklist Instalasi Utama....379
E.2 Checklist Keamanan Minimum....380
E.3 Checklist Setelah Reboot....380
Lampiran F — 30 Command Wajib Admin SOC....381
F.1 Daftar 30 Command....381
F.2 Command Paket Cepat untuk Triage SSH....382
F.3 Command Paket Cepat untuk FIM....383
F.4 Command Paket Cepat untuk Network Alert....383
F.5 Command Paket Cepat untuk Hash Evidence....383
F.6 Golden Rule Command Line SOC....384
Glossary....385
Referensi....393
Tentang Penulis....397

Bab 1 — Fondasi SOC Open Source di Ubuntu 26.04

Bab ini terutama gagasan bahwa SOC bukan sekadar dashboard, tetapi kemampuan membaca log, menghubungkan bukti, membuat alert, melakukan triage, dan merespons secara etis. Dokumen terlampir juga menegaskan alur minimum Bab 1: kebutuhan SOC untuk admin jaringan, SOC sebagai sistem kerja, komponen minimum, peta tools, arsitektur mini SOC, *golden rule*, dan praktik membuat desain SOC di file teks.

Secara teknis, Ubuntu 26.04 LTS layak dipakai sebagai basis lab karena rilis LTS ini mendapat pembaruan keamanan dan perbaikan penting selama lima tahun sampai April 2031. (documentation.ubuntu.com) Wazuh resmi mendefinisikan arsitekturnya sebagai kombinasi *agent*, *server*, *indexer*, dan *dashboard* untuk perlindungan SIEM/XDR. (documentation.wazuh.com) Security Onion relevan untuk visibilitas jaringan karena mengintegrasikan fungsi *network visibility*, *host visibility*, *intrusion detection*, *log management*, dan *case management*. (docs.securityonion.net) Suricata diposisikan sebagai perangkat lunak terbuka untuk analisis jaringan dan deteksi ancaman, sedangkan Zeek kuat sebagai penganalisis trafik pasif yang menghasilkan log transaksi jaringan bernilai investigatif. (Suricata) OpenSearch, MISP, DFIR-IRIS, dan Shuffle melengkapi ekosistem untuk pencarian log, intelijen ancaman, pengelolaan kasus, dan otomasi. (OpenSearch)

1.1 Mengapa Admin Jaringan Perlu SOC

Admin jaringan selama ini sering dinilai dari satu hal: **jaringan hidup atau mati**. Jika koneksi berjalan, alamat IP benar, DNS menjawab, dan layanan bisa diakses, pekerjaan dianggap selesai. Cara berpikir seperti ini tidak lagi cukup.

Di jaringan modern, koneksi yang terlihat normal bisa saja sedang dipakai untuk:

- percobaan login berulang;
- pemindaian port;
- pencurian kredensial;
- koneksi ke alamat IP mencurigakan;
- perubahan file konfigurasi;
- komunikasi malware;
- akses ilegal ke server;
- penyalahgunaan akun admin.

SOC membantu admin jaringan melihat tanda bahaya sebelum menjadi bencana.

SOC bukan hanya ruang besar dengan layar besar. SOC adalah **kemampuan operasional** untuk mengumpulkan log, membaca tanda anomali, memvalidasi alert, menghubungkan bukti, dan mengambil tindakan yang dapat dipertanggungjawabkan.

Untuk admin jaringan, SOC menjawab pertanyaan praktis:

- Siapa yang mencoba login?
- Dari IP mana percobaan itu datang?
- Berapa kali gagal?
- Apakah ada login sukses setelah gagal berulang?
- File apa yang berubah?
- Service apa yang tiba-tiba mati?
- Apakah endpoint mengirim trafik aneh?
- Apakah ada port scanning?
- Apakah alert ini benar ancaman atau hanya noise?

Tanpa SOC, admin hanya melihat gejala.

Dengan SOC, admin melihat **kronologi**.

Tanpa SOC, admin sering bereaksi setelah pengguna komplain.

Dengan SOC, admin bisa mendeteksi lebih awal.

Tanpa SOC, keputusan sering berbasis dugaan.

Dengan SOC, keputusan harus berbasis **log, alert, bukti, dan timeline**.

Posisi Strategis Admin Jaringan

Admin jaringan berada di titik paling penting dalam pertahanan digital. Ia memahami IP address, VLAN, routing, firewall, server, DNS, port, bandwidth, dan pola trafik harian. Pengetahuan ini membuat admin jaringan sangat cocok naik kelas menjadi operator SOC.

Namun ada satu perubahan cara berpikir.

Admin jaringan biasa bertanya:

> “Apakah koneksi berjalan?”

Admin SOC bertanya:

> “Apakah koneksi ini normal, sah, dan bisa dijelaskan?”

Itulah lompatan penting dari *network operation* ke *security operation*.

Ancaman yang Sering Terlihat dari SOC

Dalam lab buku ini, pembaca akan diarahkan melihat ancaman sederhana tetapi nyata:

Gejala	Sumber Bukti	Makna Keamanan
Login SSH gagal berulang	`/var/log/auth.log`, Wazuh alert	Potensi *brute force*
File `/etc/passwd` berubah	Wazuh FIM, auditd	Potensi perubahan akun
Port scan	Suricata, Zeek, tcpdump	Potensi tahap pengintaian
Service mati	`systemctl`, `journalctl`	Potensi gangguan layanan
Trafik DNS aneh	Zeek `dns.log`	Potensi komunikasi mencurigakan
Alert level tinggi	Wazuh `alerts.json`	Perlu triage
IOC cocok	MISP / feed intelijen ancaman	Perlu investigasi lanjut

Inti SOC adalah membuktikan, bukan menebak.

1.2 SOC sebagai Sistem Kerja, Bukan Sekadar Tools

Kesalahan umum pemula adalah menganggap SOC sama dengan memasang banyak perangkat lunak. Padahal tools hanya alat. SOC yang benar adalah **sistem kerja**.

Alur kerja minimum SOC:

```
log → event → alert → triage → incident → response → report → lesson learned
```

Maknanya:

- **Log** adalah catatan mentah dari sistem.
- **Event** adalah kejadian yang terekam.
- **Alert** adalah event yang dianggap penting oleh rule.
- **Triage** adalah proses memilah alert: penting atau tidak.
- **Incident** adalah kejadian yang terbukti berdampak atau berisiko.
- **Response** adalah tindakan terukur untuk membatasi dampak.
- **Report** adalah dokumentasi teknis dan manajerial.
- **Lesson learned** adalah perbaikan agar insiden tidak berulang.

Contoh Sederhana

Sebuah endpoint mencatat:

```
Failed password for invalid user admin from 203.0.113.10 port 53422 ssh2
```

Ini adalah **log**.

Jika terjadi sekali, mungkin hanya noise.

Jika terjadi 100 kali dalam 5 menit, sistem deteksi bisa mengubahnya menjadi **alert**.

Analis kemudian melakukan **triage**:

- IP sumber siapa?
- User yang dicoba apa?
- Apakah ada login sukses setelahnya?
- Endpoint mana yang diserang?
- Apakah IP tersebut muncul di sensor jaringan?
- Apakah ada perubahan file setelah login?

Jika bukti cukup, alert naik menjadi **incident**.

Setelah itu baru dilakukan **response**, misalnya:

- blokir IP di lab;
- ubah konfigurasi SSH;
- aktifkan autentikasi lebih kuat;
- periksa user baru;
- simpan evidence;
- tulis laporan.

Jangan melompat dari alert langsung ke respons.

Alert adalah sinyal.

Bukti adalah dasar keputusan.

Cara Berpikir SOC

SOC yang baik selalu bertanya:

- Apa yang terjadi?
- Kapan terjadi?
- Di host mana?
- User siapa?
- IP sumber mana?
- Port apa?
- Log mana yang mendukung?
- Apakah ada bukti dari sumber lain?
- Apa dampaknya?
- Apa tindakan paling aman?
- Apa yang harus didokumentasikan?

Pertanyaan ini lebih penting daripada hafal semua menu tools.

Perbedaan Operator Tools dan Analis SOC

Operator Tools	Analis SOC
Fokus pada dashboard	Fokus pada bukti
Percaya alert mentah	Memvalidasi alert
Reaktif	Investigatif
Menghapus masalah	Menyimpan evidence
Menjawab “ada alert”	Menjawab “apa buktinya?”
Mengejar banyak tools	Menguasai alur kerja

SOC bukan lomba memasang tools terbanyak. SOC adalah disiplin membaca tanda bahaya secara benar.

1.3 Komponen Minimum SOC

Sebuah SOC kecil tidak harus mahal dan kompleks. Untuk belajar, cukup mulai dari arsitektur minimum. Dokumen Open Source SOC Mastery juga menekankan proyek mini SOC berbasis Wazuh, OpenSearch, DFIR-IRIS, sensor jaringan/log jaringan, dan workflow otomasi sebagai konfigurasi realistis.

Komponen minimum:

- **SOC server**
- **Agent endpoint**
- **Network sensor**
- **Log collector**
- **SIEM/XDR**
- **Dashboard**
- **Threat intelligence**
- **Case management**
- **Automation / SOAR**

1. SOC Server

SOC server adalah pusat operasi. Di sinilah log diterima, alert diproses, data diindeks, dashboard dibuka, dan investigasi dilakukan.

Dalam buku ini, SOC server berbasis:

- Ubuntu Server 26.04 LTS;
- Wazuh Manager;
- Wazuh Indexer / OpenSearch;
- Wazuh Dashboard;
- tool CLI seperti `curl`, `jq`, `grep`, `awk`, `journalctl`, `systemctl`, `ss`, dan `tcpdump`.

SOC server harus diperlakukan sebagai aset kritikal.

Jika SOC server lemah, maka sistem pemantauan juga lemah.

2. Agent Endpoint

Agent adalah mata dan telinga di endpoint. Ia mengirim informasi dari mesin yang dipantau.

Contoh data dari agent:

- log autentikasi;
- perubahan file penting;
- daftar paket;
- informasi proses;
- konfigurasi sistem;
- event keamanan;
- status service.

Dalam Wazuh, agent mengirim data ke Wazuh server untuk dianalisis, lalu hasil analisis diteruskan ke indexer dan dashboard. (documentation.wazuh.com)

3. Network Sensor

Agent melihat endpoint.

Sensor melihat jaringan.

Keduanya saling melengkapi.

Sensor jaringan berguna untuk membaca:

- koneksi antar host;
- DNS query;
- HTTP request;
- TLS metadata;
- port scan;
- signature serangan;
- file transfer;
- anomali trafik.

Di buku ini, sensor jaringan diarahkan memakai:

- Security Onion;
- Suricata;
- Zeek;
- `tcpdump`.

Security Onion resmi mencakup visibilitas jaringan, deteksi intrusi, pengelolaan log, dan case management. (docs.securityonion.net) Suricata menghasilkan output EVE JSON yang dapat berisi alert, anomali, metadata, informasi file, dan record protokol. (Suricata User Guide) Zeek menghasilkan log seperti `conn.log` yang penting untuk memahami koneksi jaringan. (Dokumentasi Zeek)

4. Log Collector

Log collector mengumpulkan catatan dari berbagai sumber:

- sistem operasi;
- aplikasi;
- firewall;
- endpoint;
- sensor jaringan;
- server;
- perangkat jaringan.

Di Ubuntu, log penting biasanya berasal dari:

- `journalctl`;
- `/var/log/auth.log`;
- `/var/log/syslog`;
- `/var/log/kern.log`;
- auditd;
- rsyslog.

Prinsipnya sederhana:

log yang tidak dikumpulkan tidak bisa dianalisis.

5. SIEM/XDR

SIEM membantu mengumpulkan, mencari, menghubungkan, dan menganalisis event keamanan.

XDR memperluas deteksi dan respons lintas endpoint, server, jaringan, dan workload.

Dalam buku ini, Wazuh menjadi fondasi utama karena menyediakan komponen agent, server, indexer, dashboard, rule, decoder, alert, API, FIM, vulnerability detection, dan active response. Dokumentasi Wazuh menyatakan platform ini menyediakan perlindungan SIEM/XDR untuk endpoint dan workload. (documentation.wazuh.com)

6. Dashboard

Dashboard membantu visualisasi.

Namun dashboard bukan sumber kebenaran tunggal.

Dashboard berguna untuk:

- melihat tren alert;
- melihat top source IP;
- melihat top endpoint;
- melihat severity;
- melihat timeline;
- mempercepat triage.

Tetapi bukti teknis tetap harus dibaca dari log.

> **Dashboard mempercepat. CLI membuktikan.**

7. Threat Intelligence

Threat intelligence memberi konteks.

Contoh:

- IP berbahaya;
- domain mencurigakan;
- hash malware;
- URL phishing;
- teknik serangan;
- kelompok ancaman;
- tag kampanye.

MISP adalah perangkat lunak open source untuk mengumpulkan, menyimpan, mendistribusikan, dan berbagi indikator serta ancaman siber. (misp-project.org)

Namun intelijen ancaman tidak boleh dipercaya mentah-mentah.

IOC harus dilihat dalam konteks:

- apakah IP benar mengakses aset kita?
- kapan terjadi?
- port apa?
- protokol apa?
- ada bukti DNS?
- ada bukti endpoint?
- ada bukti file?
- ada dampak?

8. Case Management

Case management mengubah alert menjadi pekerjaan investigasi yang rapi.

Isi case minimal:

- judul;
- severity;
- waktu deteksi;
- aset terdampak;
- evidence;
- observables;
- timeline;
- catatan analis;
- task;
- rekomendasi;
- status penutupan.

DFIR-IRIS adalah platform kolaboratif untuk analis incident response agar dapat berbagi investigasi teknis. (DFIR-IRIS Docs)

Tanpa case management, investigasi mudah tercecer.

Dengan case management, setiap keputusan punya jejak.

9. Automation / SOAR

SOAR membantu otomasi.

Namun otomasi tidak boleh menggantikan penilaian analis.

Contoh otomasi aman:

- parsing alert JSON;
- mengambil source IP;
- cek reputasi IOC;
- membuat case;
- mengirim notifikasi;
- membuat template laporan;
- meminta approval sebelum respons.

Shuffle adalah platform otomasi open source yang dibangun untuk kebutuhan operasi keamanan. (GitHub)

> **Otomasi SOC harus punya kontrol manusia, log tindakan, dan batas risiko.**

1.4 Peta Tools Open Source

Bab ini tidak mengajarkan semua tools sekaligus. Bab ini memberi peta.

Instalasi dan konfigurasi detail akan dibahas bertahap pada bab berikutnya.

Tools Utama

Tool	Peran dalam SOC	Dipakai untuk
Ubuntu Server 26.04 LTS	Sistem operasi server SOC	Fondasi lab dan server
Wazuh	SIEM/XDR	Agent, alert, FIM, rule, API
Wazuh Agent	Endpoint telemetry	Kirim log dan event
OpenSearch	Pencarian dan analitik log	Index, query, dashboard
Security Onion	Platform sensor	Visibilitas jaringan dan investigasi
Suricata	Deteksi jaringan berbasis rule	Alert IDS/IPS, EVE JSON
Zeek	Metadata jaringan	`conn.log`, `dns.log`, `http.log`
auditd	Audit aktivitas Linux	Pantau file dan event sistem
rsyslog	Pengiriman log	Koleksi log sistem
Pipeline log	Pengiriman dan normalisasi log	Integrasi antar komponen
DFIR-IRIS	Case management	Incident, evidence, timeline
MISP	Threat intelligence	IOC, tag, event, feed
Shuffle	SOAR	Workflow, webhook, approval
Python	Otomasi ringan	Parsing JSON, laporan, enrichment
CLI Linux	Pembuktian cepat	`grep`, `awk`, `jq`, `curl`, `ss`, `tcpdump`

Peta Peran Tools

Endpoint Linux
└── Wazuh Agent
└── Wazuh Manager
└── Wazuh Indexer / OpenSearch
└── Dashboard
└── Triage
└── Case Management
└── Response

Network Mirror / Sensor
└── Suricata / Zeek
└── Alert / Metadata
└── OpenSearch / Security Onion
└── Investigasi
└── Evidence

Alert Valid
└── IOC / Threat Intelligence
└── MISP
└── Case
└── Shuffle Workflow
└── Approval
└── Response

Prinsip Pemilihan Tools

Gunakan tools berdasarkan fungsi, bukan gengsi.

Pilih tools yang:

- terbuka;
- bisa diaudit;
- bisa dijalankan di lab;
- punya dokumentasi;
- mendukung CLI/API;
- menghasilkan log atau JSON;
- cocok untuk pembelajaran;
- bisa dipertanggungjawabkan.

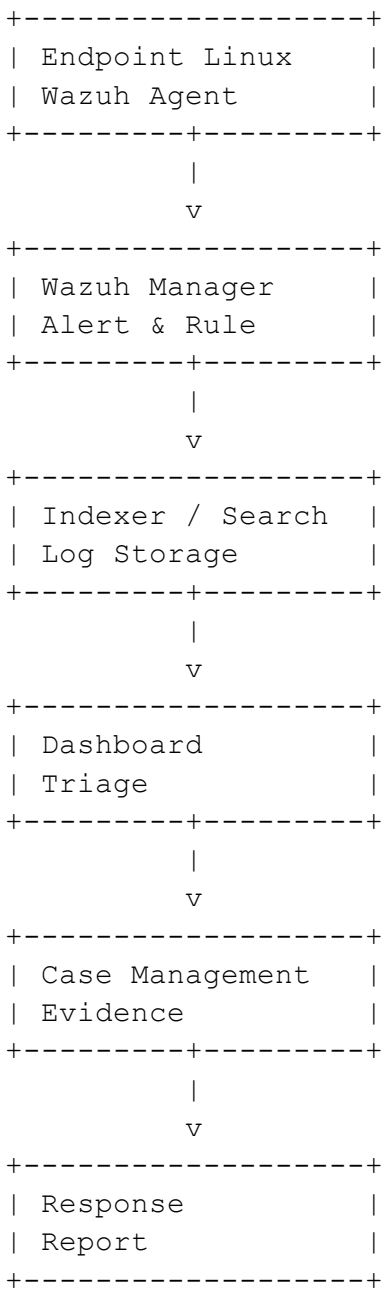
Jangan memilih tools hanya karena populer.

Pilih karena ia menjawab kebutuhan investigasi.

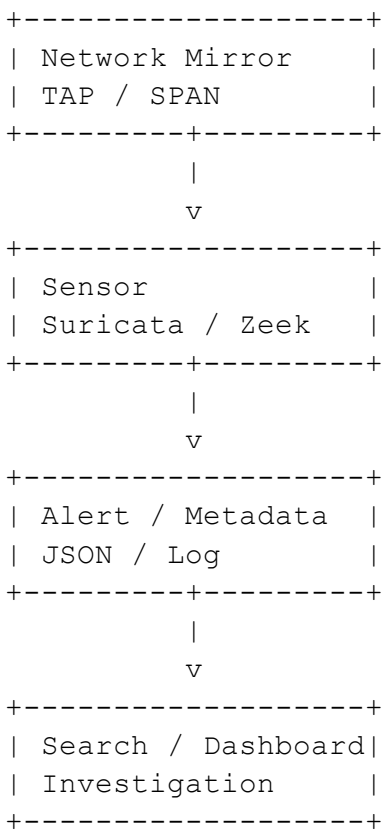
1.5 Arsitektur Mini SOC

Arsitektur mini SOC dalam buku ini dirancang agar realistis untuk admin jaringan dan lab kampus.

Topologi Minimum



Topologi Sensor Jaringan



Arsitektur Lab Minimum

Untuk Bab 1, cukup rancang dulu. Instalasi dilakukan di bab berikutnya.

Rekomendasi VM minimum:

VM	Role	Contoh IP	Fungsi
`soc-server`	SOC server	`192.168.56.10`	Wazuh, OpenSearch, dashboard
`endpoint-01`	Endpoint Linux	`192.168.56.20`	Dipasang agent
`target-lab`	Server target lab	`192.168.56.30`	Simulasi login, service, web
`sensor-01`	Sensor jaringan	`192.168.56.40`	Suricata / Zeek / tcpdump

Jika perangkat terbatas, mulai dari 2 VM:

- `soc-server`;
- `endpoint-01`.

Tambahkan sensor setelah Bab 5.

Jalur Data

SOC harus punya jalur data yang jelas.

Contoh:

```
/var/log/auth.log
  → Wazuh Agent
  → Wazuh Manager
  → alerts.json
  → OpenSearch
  → Dashboard
  → Case
  → Report
```

Contoh sensor jaringan:

```
Trafik jaringan
  → Suricata
  → eve.json
  → jq / OpenSearch
  → Alert
  → Triage
  → Evidence
```

Contoh metadata Zeek:

```
Trafik jaringan
  → Zeek
  → conn.log / dns.log / http.log
  → korelasi IP dan waktu
  → timeline investigasi
```

Prinsip Desain Arsitektur

Arsitektur mini SOC harus memenuhi syarat:

- ada sumber log;
- ada endpoint yang dipantau;
- ada server penerima log;
- ada tempat mencari log;
- ada cara membaca alert;
- ada folder evidence;
- ada format laporan;
- ada batas lab;
- ada etika penggunaan.

SOC kecil yang rapi lebih berharga daripada SOC besar yang tidak bisa dijelaskan.

1.6 Golden Rule SOC

Tiga aturan emas:

> **Jangan percaya alert sebelum membaca log.**
> **Jangan respons sebelum punya bukti.**
> **Jangan otomatisasi sebelum paham risikonya.**

Aturan 1 — Jangan Percaya Alert Sebelum Membaca Log

Alert adalah hasil rule.

Rule bisa benar.

Rule juga bisa salah.

Penyebab *false positive*:

- rule terlalu sensitif;
- aktivitas admin dianggap serangan;
- scanner internal tidak didaftarkan;
- waktu sistem tidak sinkron;
- parsing log salah;
- event normal terlihat mencurigakan;
- konteks organisasi tidak dimasukkan.

Selalu cari log mentah.

Contoh pertanyaan:

- Alert berasal dari log apa?
- Field mana yang memicu rule?
- Apakah ada bukti di host?
- Apakah ada bukti di jaringan?
- Apakah kejadian berulang?
- Apakah ada user valid?
- Apakah ada dampak?

Aturan 2 — Jangan Respons Sebelum Punya Bukti

Respons yang salah bisa lebih berbahaya daripada alert.

Contoh kesalahan:

- memblokir IP gateway;
- mematikan server produksi;
- menghapus file bukti;
- menutup akses admin sah;
- menghapus log sebelum disimpan;
- mengubah sistem tanpa catatan.

Respons harus:

- berbasis bukti;
- terukur;
- dicatat;
- bisa diaudit;
- punya alasan teknis;
- punya rencana pemulihan.

Aturan 3 — Jangan Otomatisasi Sebelum Paham Risikonya

Otomasi terlihat keren.
Namun otomasi yang salah bisa merusak layanan.

Contoh otomasi berisiko:

- auto-block semua source IP alert;
- auto-disable user;
- auto-kill proses;
- auto-delete file;
- auto-isolate host.

Untuk lab, otomasi boleh diuji.
Untuk produksi, gunakan approval.

Format aman:

```
alert masuk
→ parsing JSON
→ enrichment
→ buat case
→ minta approval
→ respons terbatas
→ catat audit trail
```

Golden Rule Tambahan

- **Tidak ada bukti, tidak ada klaim.**
- **Tidak ada timeline, investigasi lemah.**
- **Tidak ada catatan, respons tidak bisa diaudit.**
- **Tidak ada etika, lab berubah menjadi pelanggaran.**
- **Tidak ada pemahaman log, dashboard hanya pajangan.**

1.7 Hands-on: Membuat Desain SOC di File Teks

Tujuan praktik ini adalah membuat desain awal SOC sebelum instalasi.

Output praktik:

- folder kerja SOC;
- file arsitektur;
- daftar VM;
- daftar IP;
- daftar sumber log;
- alur alert;
- rencana respons;
- checklist awal.

1.7.1 Membuat Struktur Folder Kerja

Jalankan:

```
mkdir -p
~/soc-lab/{docs,configs,logs,evidence,scripts,playbooks}
```

Penjelasan per bagian:

- `mkdir` berarti membuat direktori.
- `-p` berarti buat folder induk jika belum ada, dan jangan error jika folder sudah ada.
- `~` berarti home directory user saat ini.
- `soc-lab` adalah folder utama lab SOC.
- `{docs,configs,logs,evidence,scripts,playbooks}` membuat beberapa subfolder sekaligus.

Struktur yang dibuat:

```
~/soc-lab/
├── docs
├── configs
├── logs
├── evidence
├── scripts
└── playbooks
```

Makna folder:

Folder	Isi
`docs`	Catatan desain, arsitektur, laporan
`configs`	Salinan konfigurasi penting
`logs`	Log latihan yang disalin
`evidence`	Bukti insiden
`scripts`	Script Python / shell
`playbooks`	Alur respons insiden

Cek hasil:

```
ls -lah ~/soc-lab
```

Penjelasan:

- `ls` menampilkan isi folder.
- `-l` menampilkan format panjang.
- `-a` menampilkan file tersembunyi.
- `-h` membuat ukuran mudah dibaca manusia.
- `~/soc-lab` adalah folder yang dicek.

1.7.2 Membuat File Arsitektur SOC

Jalankan:

```
nano ~/soc-lab/docs/arsitektur-soc.txt
```

Penjelasan:

- `nano` membuka editor teks terminal.
- `~/soc-lab/docs/arsitektur-soc.txt` adalah file desain arsitektur SOC.

Isi file:

```
# Arsitektur Mini SOC Ubuntu 26.04

## 1. Tujuan
Membangun Mini SOC open source untuk monitoring endpoint
Linux,
membaca log, menghasilkan alert, melakukan triage,
menyimpan evidence,
dan membuat laporan incident response.

## 2. Daftar Mesin
1. soc-server
   - Role: Wazuh Manager, Indexer, Dashboard
   - IP: 192.168.56.10
   - OS: Ubuntu Server 26.04 LTS

2. endpoint-01
   - Role: Endpoint Linux dengan Wazuh Agent
   - IP: 192.168.56.20
   - OS: Ubuntu Server 26.04 LTS

3. target-lab
   - Role: Server uji untuk simulasi login gagal dan
service
   - IP: 192.168.56.30
   - OS: Ubuntu Server 26.04 LTS

4. sensor-01
   - Role: Network sensor Suricata / Zeek
   - IP: 192.168.56.40
   - OS: Ubuntu Server 26.04 LTS

## 3. Sumber Log
- /var/log/auth.log
- /var/log/syslog
- journalctl
```

```
- Wazuh alerts.json
- Suricata eve.json
- Zeek conn.log
- Zeek dns.log
- auditd

## 4. Alur Endpoint
Endpoint Linux
→ Wazuh Agent
→ Wazuh Manager
→ Wazuh Indexer / OpenSearch
→ Dashboard
→ Triage
→ Case
→ Response
→ Report

## 5. Alur Sensor Jaringan
Network mirror / sensor
→ Suricata / Zeek
→ Alert / metadata
→ Search / dashboard
→ Investigasi
→ Evidence
→ Report

## 6. Skenario Deteksi Awal
1. Login SSH gagal berulang
2. Perubahan file penting
3. Port scan di lab tertutup
4. Service mati
5. Trafik DNS mencurigakan

## 7. Golden Rule
- Jangan percaya alert sebelum membaca log.
- Jangan respons sebelum punya bukti.
- Jangan otomatisasi sebelum paham risikonya.

## 8. Rencana Evidence
Evidence disimpan di:
~/soc-lab/evidence/

Jenis evidence:
- alert JSON
- log CLI
- timeline
- catatan analis
- IOC
```

```
- screenshot dashboard bila diperlukan

## 9. Rencana Response
Respons awal:
1. Validasi alert
2. Cari bukti tambahan
3. Buat timeline
4. Simpan evidence
5. Buat case
6. Minta approval jika perlu tindakan blocking
7. Tulis laporan
8. Buat lesson learned
```

Simpan di nano:

```
Ctrl+O → Enter → Ctrl+X
```

Makna:

- `Ctrl+O` menyimpan file.
- `Enter` menyetujui nama file.
- `Ctrl+X` keluar dari editor.

1.7.3 Membaca Ulang File Desain

Jalankan:

```
cat ~/soc-lab/docs/arsitektur-soc.txt
```

Penjelasan:

- `cat` menampilkan isi file.
- file yang dibaca adalah desain SOC yang baru dibuat.

Gunakan `less` jika file panjang:

```
less ~/soc-lab/docs/arsitektur-soc.txt
```

Penjelasan:

- `less` membuka file agar bisa di scroll.
- tekan `q` untuk keluar.

1.7.4 Membuat Checklist Awal SOC

Jalankan:

```
nano ~/soc-lab/docs/checklist-awal-soc.txt
```

Isi:

```
# Checklist Awal Mini SOC

## Infrastruktur
[ ] SOC server ditentukan
[ ] Endpoint ditentukan
[ ] Target lab ditentukan
[ ] Sensor jaringan ditentukan
[ ] IP address dicatat
[ ] Segmentasi lab ditentukan
[ ] Akses internet dibatasi sesuai kebutuhan
[ ] Dashboard tidak diekspos publik

## Log
[ ] /var/log/auth.log tersedia
[ ] journalctl bisa dibaca
[ ] syslog tersedia
[ ] Wazuh alert direncanakan
[ ] Suricata eve.json direncanakan
[ ] Zeek log direncanakan
[ ] auditd direncanakan

## Evidence
[ ] Folder evidence dibuat
[ ] Format nama file evidence dibuat
[ ] Timeline disiapkan
[ ] Template incident report disiapkan

## Etika
[ ] Semua simulasi dilakukan di lab sendiri
[ ] Tidak menyerang sistem publik
[ ] Tidak memakai malware asli
[ ] Tidak menyimpan data sensitif tanpa izin
[ ] Semua respons dicatat
```

1.7.5 Membuat Template Timeline

Jalankan:

```
nano ~/soc-lab/docs/template-timeline.txt
```

Isi:

```
# Template Timeline Incident

Incident ID:
Judul:
Tanggal:
Analis:

| Waktu | Sumber Data | Event | Bukti | Catatan |
|------|-------------|-------|-------|---------|
|      |             |       |       |         |
|      |             |       |       |         |
|      |             |       |       |         |

Kesimpulan awal:

Tindakan:

Lesson learned:
```

Timeline penting karena SOC bekerja berdasarkan urutan kejadian.
Tanpa timeline, investigasi mudah kabur.

1.7.6 Membuat Template Incident Report

Jalankan:

```
nano ~/soc-lab/docs/template-incident-report.txt
```

Isi:

```
# Template Incident Report

## 1. Ringkasan
Tuliskan ringkasan singkat insiden.

## 2. Waktu Deteksi
Tanggal:
Jam:
Zona waktu:

## 3. Sumber Alert
Tool:
Rule:
Severity:
File log:

## 4. Aset Terdampak
Hostname:
IP:
Role:

## 5. Bukti
- Alert JSON:
- Log CLI:
- Sensor:
- IOC:
- Timeline:

## 6. Analisis
Apa yang terjadi?
Mengapa dianggap penting?
Apakah ada dampak?

## 7. Respons
Tindakan yang dilakukan:
Approval:
Hasil:

## 8. Rekomendasi
Perbaikan konfigurasi:
Perbaikan monitoring:
Perbaikan proses:

## 9. Lesson Learned
Apa yang harus diperbaiki agar kejadian tidak berulang?
```

1.7.7 Membuat Catatan Desain dalam Format Markdown

Markdown memudahkan dokumentasi.

Jalankan:

```
nano ~/soc-lab/docs/arsitektur-soc.md
```

Isi:

````
# Arsitektur Mini SOC Ubuntu 26.04

## Tujuan
Membangun SOC kecil berbasis open source untuk belajar
monitoring,
deteksi, triage, evidence, response, dan report.

## Komponen
- Ubuntu Server 26.04 LTS
- Wazuh
- Wazuh Agent
- OpenSearch
- Security Onion
- Suricata
- Zeek
- auditd
- rsyslog
- DFIR-IRIS
- MISP
- Shuffle
- Python
- CLI Linux

## Alur Utama
```text
log → event → alert → triage → incident → response →
report → lesson learned

## Golden Rule
   1. Jangan percaya alert sebelum membaca log.
   2. Jangan respons sebelum punya bukti.
   3. Jangan otomatisasi sebelum paham risikonya.
````
````

1.7.8 Validasi Struktur Dokumen

Jalankan:

```
find ~/soc-lab -maxdepth 2 -type f
```

Penjelasan:

- `find` mencari file/folder.
- `~/soc-lab` adalah lokasi pencarian.
- `-maxdepth 2` membatasi kedalaman pencarian sampai dua level.
- `-type f` hanya menampilkan file.

Hasil yang diharapkan:

```
/home/user/soc-lab/docs/arsitektur-soc.txt
/home/user/soc-lab/docs/checklist-awal-soc.txt
/home/user/soc-lab/docs/template-timeline.txt
/home/user/soc-lab/docs/template-incident-report.txt
/home/user/soc-lab/docs/arsitektur-soc.md
```

1.7.9 Praktik Python Ringan: Membuat Ringkasan Desain

Python dipakai untuk membantu dokumentasi, bukan menggantikan analisis.
Buat script:

```
nano ~/soc-lab/scripts/ringkas_desain.py
```

Isi:

```
from pathlib import Path

base = Path.home() / "soc-lab"
docs = base / "docs"

files = [
    "arsitektur-soc.txt",
    "checklist-awal-soc.txt",
    "template-timeline.txt",
    "template-incident-report.txt",
]

print("Ringkasan Dokumen SOC")
print("=" * 30)

for name in files:
    path = docs / name
    if path.exists():
        lines =
path.read_text(encoding="utf-8").splitlines()
```

```
            print(f"{name}: {len(lines)} baris")
        else:
            print(f"{name}: belum ada")
```

Jalankan:

```
python3 ~/soc-lab/scripts/ringkas_desain.py
```

Penjelasan script:

- `from pathlib import Path` memanggil modul untuk mengelola path file.
- `Path.home()` mengambil folder home user.
- `base` menunjuk ke folder `soc-lab`.
- `docs` menunjuk ke folder dokumentasi.
- `files` berisi daftar file yang akan dicek.
- `path.exists()` mengecek apakah file ada.
- `read_text()` membaca isi file.
- `splitlines()` memecah isi file menjadi baris.
- `len(lines)` menghitung jumlah baris.

Output contoh:

```
Ringkasan Dokumen SOC
==============================
arsitektur-soc.txt: 72 baris
checklist-awal-soc.txt: 35 baris
template-timeline.txt: 17 baris
template-incident-report.txt: 39 baris
```

Makna praktik ini:

- pembaca mulai membangun kebiasaan dokumentasi;
- Python dipakai untuk validasi sederhana;
- semua artefak tersimpan rapi;
- lab siap dilanjutkan ke Bab 2.

1.8 Kesalahan Umum Pemula SOC

1. Terlalu Cepat Instalasi

Banyak pemula langsung memasang tools tanpa desain.

Akibatnya:

- IP tidak jelas;
- role VM tidak jelas;
- log tidak tahu dari mana;
- dashboard ada tetapi kosong;
- alert muncul tetapi tidak dipahami;
- evidence tidak tersimpan.

Desain dulu, instal kemudian.

2. Terlalu Percaya Dashboard

Dashboard bagus untuk ringkasan.

Namun investigasi tetap harus turun ke log.

Minimal kuasai:

```
journalctl
grep
awk
jq
tail
less
ss
tcpdump
curl
```

3. Tidak Mencatat Evidence

Alert tanpa evidence lemah.

Evidence minimum:

- waktu kejadian;
- alert JSON;
- log mentah;
- hostname;
- IP;
- user;
- rule;
- severity;
- tindakan;
- hasil.

4. Mencampur Lab dan Produksi

Lab harus terpisah.
Jangan melakukan simulasi serangan ke sistem publik.
Jangan memakai kredensial asli.
Jangan memakai data sensitif.
Jangan menjalankan respons otomatis di sistem penting tanpa approval.

5. Otomasi Terlalu Cepat

Sebelum otomasi, pahami:

- input alert;
- field JSON;
- kondisi trigger;
- dampak action;
- cara rollback;
- audit trail;
- approval.

SOAR yang baik bukan yang paling agresif, tetapi yang paling terkendali.

1.9 Checklist Kompetensi Bab 1

Setelah menyelesaikan bab ini, pembaca harus mampu:

- menjelaskan mengapa admin jaringan perlu SOC;
- membedakan log, event, alert, triage, incident, response, report, dan lesson learned;
- menyebutkan komponen minimum SOC;
- menjelaskan peran Wazuh, OpenSearch, Security Onion, Suricata, Zeek, MISP, DFIR-IRIS, Shuffle, auditd, rsyslog, dan Python;
- menggambar arsitektur mini SOC;
- membuat folder kerja SOC;
- membuat file desain SOC;
- membuat checklist awal;
- membuat template timeline;
- membuat template incident report;
- menjalankan script Python sederhana untuk validasi dokumen;
- memahami *golden rule* SOC;
- memahami etika lab.

1.10 Rangkuman Bab

SOC bukan produk. SOC adalah **disiplin kerja**.

Admin jaringan membutuhkan SOC karena ancaman modern sering muncul sebagai tanda kecil: login gagal, port scan, file berubah, service mati, DNS aneh, atau trafik yang tidak biasa. Tanda kecil ini harus dikumpulkan, dibaca, dikorelasikan, dan dibuktikan.

Bab ini membangun fondasi sebelum instalasi:

- **log adalah bahan mentah**;
- **alert adalah sinyal**;
- **triage adalah proses berpikir**;
- **incident membutuhkan bukti**;
- **response harus terukur**;
- **report harus bisa diaudit**;
- **lesson learned adalah perbaikan sistem**.

Mini SOC open source tidak harus besar. Yang penting rapi, legal, terdokumentasi, dan bisa dijelaskan.

Output Bab

Setelah Bab 1, pembaca memiliki:

- folder kerja `~/soc-lab/`;
- file `arsitektur-soc.txt`;
- file `checklist-awal-soc.txt`;
- file `template-timeline.txt`;
- file `template-incident-report.txt`;
- file `arsitektur-soc.md`;
- script `ringkas_desain.py`;
- peta komponen SOC;
- desain awal mini SOC;
- prinsip investigasi berbasis bukti.

Kalimat kunci Bab 1:

> **Admin jaringan yang hebat bukan hanya menjaga koneksi tetap hidup, tetapi mampu membaca tanda bahaya, membuktikan insiden dari log, dan merespons dengan disiplin.**

Bab 2 — Instalasi Ubuntu 26.04 untuk Server SOC

Bab 2 yang memuat spesifikasi lab, instalasi server, update awal, IP static, SSH, UFW, Docker/Compose, struktur direktori, dan backup konfigurasi. Bab ini juga selaras dengan prinsip dari dokumen **Open Source SOC Mastery**: lab SOC harus dibangun bertahap, aman, terdokumentasi, tidak mengekspos dashboard ke internet, dan seluruh eksperimen dilakukan di lingkungan legal serta tertutup.

Ubuntu 26.04 LTS resmi mendapat dukungan keamanan dan perbaikan penting selama lima tahun sampai April 2031, sehingga layak dijadikan fondasi server SOC jangka menengah. (Dokumentasi Ubuntu) Ubuntu Server memakai Netplan untuk konfigurasi jaringan berbasis YAML, OpenSSH dikonfigurasi melalui `/etc/ssh/sshd_config` atau direktori konfigurasi modular, dan UFW adalah alat firewall default yang menyederhanakan pengelolaan aturan jaringan host. (Ubuntu)

2.1 Spesifikasi Minimum Lab

Server SOC bukan server biasa. Ia akan menerima log, menyimpan alert, menjalankan indexer, membuka dashboard, menjalankan agent manager, menerima koneksi endpoint, dan menjadi pusat investigasi.

Karena itu, spesifikasi tidak boleh terlalu minimalis.

Prinsip utama:

> **SOC server yang lambat akan membuat investigasi lambat. Investigasi lambat membuat respons terlambat.**

Minimum Pribadi

Untuk belajar mandiri:

Komponen	Minimum
RAM	16 GB
Storage kosong	150 GB
CPU	4 core
VM minimum	3 VM

VM minimum:

- `soc-server`
- `endpoint-01`
- `target-lab`

Konfigurasi ini cukup untuk:

- instalasi Ubuntu Server;
- konfigurasi SSH;
- konfigurasi IP static;
- instalasi Wazuh single-node pada bab berikutnya;
- endpoint dengan Wazuh Agent;
- simulasi login gagal;
- simulasi perubahan file;
- latihan membaca log.

Ideal untuk Praktik Serius

Untuk pembaca yang ingin membangun portofolio SOC:

Komponen	Ideal
RAM	32–64 GB
Storage	SSD 500 GB–1 TB
CPU	8 core atau lebih
VM	4 VM atau lebih

VM ideal:

- `soc-server`
- `endpoint-01`
- `target-lab`
- `sensor-01`

Dengan 4 VM, pembaca bisa memisahkan:

- server SOC;
- endpoint yang dipantau;
- target simulasi;
- sensor jaringan.

Rekomendasi Pembagian VM

VM	Hostname	RAM	CPU	Storage	Role
1	soc-server	8–16 GB	4 core	100–200 GB	Server SOC
2	endpoint-01	2–4 GB	1–2 core	25–40 GB	Endpoint agent
3	target-lab	2–4 GB	1–2 core	25–40 GB	Target simulasi
4	sensor-01	4–8 GB	2–4 core	50–100 GB	Sensor jaringan

Jika mesin terbatas, mulai dari 3 VM.

Sensor jaringan dapat ditambahkan pada Bab 5.

Catatan Strategis

Jangan memaksakan semua tools aktif sekaligus jika hardware terbatas.

Urutan yang lebih sehat:

1. Ubuntu Server stabil.
2. SSH aman.
3. IP static benar.
4. Firewall aktif.
5. Struktur direktori rapi.
6. Backup konfigurasi dibuat.
7. Baru instal tools SOC.

Fondasi buruk akan membuat seluruh SOC rapuh.

2.2 Instalasi Ubuntu Server 26.04

Instalasi server SOC harus rapi sejak awal.

Keputusan kecil seperti hostname, user admin, timezone, IP, dan partisi akan mempengaruhi dokumentasi, troubleshooting, dan investigasi.

Parameter Instalasi

Gunakan parameter berikut:

Komponen	Rekomendasi
OS	Ubuntu Server 26.04 LTS
Hostname	soc-server
User admin	socadmin
Timezone	Asia/Jakarta
SSH server	Aktif
IP	Static
Update awal	Wajib
Dashboard publik	Tidak boleh diekspos langsung

Hostname

Hostname harus jelas.
Contoh:

```
soc-server
endpoint-01
target-lab
sensor-01
```

Jangan gunakan nama acak seperti:

```
ubuntu
server
test
user-pc
```

Hostname yang jelas memudahkan:

- membaca log;
- mencari alert;
- membuat laporan;
- membuat timeline;
- membedakan aset.

User Admin

Gunakan user khusus admin SOC.
Contoh:

```
socadmin
```

Hindari memakai nama pribadi untuk lab yang akan dibagikan.
Alasannya:

- lebih rapi untuk dokumentasi;
- lebih aman saat screenshot;
- lebih mudah disanitasi;
- tidak membocorkan identitas pribadi.

Partisi

Untuk lab kecil, partisi otomatis cukup.

Untuk lab serius, pisahkan ruang log bila memungkinkan.

Rekomendasi sederhana:

Mount Point	Fungsi
`/`	Sistem utama
`/var`	Log dan data service
`/opt`	Tools SOC
`/home`	Dokumen kerja admin

Jika tidak ingin kompleks, gunakan partisi otomatis. Namun pastikan storage cukup.

Timezone

Timezone harus benar.
SOC sangat bergantung pada waktu.

Jika waktu salah:

- timeline kacau;
- alert sulit dikorelasikan;
- evidence lemah;
- laporan tidak dipercaya.

Timezone buku ini:

```
Asia/Jakarta
```

IP Static

Server SOC sebaiknya memakai IP static.

Jangan mengandalkan alamat dinamis untuk server SOC.

Alasannya:

- endpoint perlu tahu alamat manager;
- dashboard perlu alamat tetap;
- firewall lebih mudah diatur;
- dokumentasi lebih stabil;
- troubleshooting lebih cepat.

Contoh skema IP lab:

Hostname	IP
soc-server	192.168.56.10
endpoint-01	192.168.56.20
target-lab	192.168.56.30
sensor-01	192.168.56.40
Gateway lab	192.168.56.1

Sesuaikan dengan jaringan masing-masing.

2.3 CLI Update dan Hardening Awal

Setelah instalasi selesai, login ke server.
Lakukan update awal:

```
sudo apt update
```

Penjelasan:

- sudo menjalankan perintah dengan hak admin.
- apt adalah manajer paket.
- update memperbarui daftar paket dari repository.

Lanjutkan upgrade:

```
sudo apt full-upgrade -y
```

Penjelasan:

- full-upgrade memperbarui paket dan menangani perubahan dependensi.
- -y menjawab “yes” otomatis.

Instal tool dasar:

```
sudo apt install -y curl wget vim nano git jq htop net-tools
tcpdump unzip gnupg ca-certificates lsb-release tree
```

Penjelasan paket:

Paket	Fungsi
`curl`	Mengakses API / URL dari CLI
`wget`	Mengunduh file
`vim`	Editor teks terminal
`nano`	Editor teks sederhana
`git`	Mengambil source code / konfigurasi
`jq`	Membaca dan memfilter JSON
`htop`	Memantau proses
`net-tools`	Tool jaringan klasik
`tcpdump`	Capture trafik jaringan
`unzip`	Ekstrak file ZIP
`gnupg`	Verifikasi kunci paket
`ca-certificates`	Sertifikat TLS
`lsb-release`	Informasi rilis sistem
`tree`	Melihat struktur direktori

Set timezone:

```
sudo timedatectl set-timezone Asia/Jakarta
```

Cek waktu:

```
timedatectl
```

Output yang diharapkan memuat:

```
Time zone: Asia/Jakarta
```

Cek Versi OS

Jalankan:

```
cat /etc/os-release
```

Cek kernel:

```
uname -a
```

Cek hostname:

```
hostnamectl
```

Hardening Awal Minimum

Hardening awal bukan berarti membuat server sempurna.
Hardening awal berarti menutup risiko paling dasar.

Lakukan:

```
sudo apt autoremove -y
sudo apt autoclean
```

Cek user aktif:

```
who
```

Cek port aktif:

```
sudo ss -tulpn
```

Cek service berjalan:

```
systemctl --type=service --state=running
```

Catatan penting: jangan mematikan service yang belum dipahami. Catat dulu, baru putuskan.

2.4 Setting IP Static dengan Netplan

Ubuntu Server memakai Netplan untuk konfigurasi jaringan berbasis file YAML. Dokumentasi Ubuntu menjelaskan bahwa konfigurasi jaringan Ubuntu ditangani melalui Netplan, yang mendefinisikan konfigurasi interface dalam file YAML. (Ubuntu)

Cek Interface

Jalankan:

```
ip a
```

Cari nama interface.
Contoh nama interface:

```
ens18
ens33
enp0s3
eth0
```

Cek route:

```
ip route
```

Contoh output:

```
default via 192.168.56.1 dev ens18
192.168.56.0/24 dev ens18 proto kernel scope link src
192.168.56.10
```

Edit File Netplan

Buka file konfigurasi:

```
sudo nano /etc/netplan/01-soc.yaml
```

Contoh konfigurasi:

```
network:
  version: 2
  ethernets:
    ens18:
      dhcp4: false
      addresses:
        - 192.168.56.10/24
      routes:
        - to: default
          via: 192.168.56.1
      nameservers:
        addresses:
          - 1.1.1.1
          - 9.9.9.9
```

Sesuaikan:

- ens18 dengan nama interface;
- 192.168.56.10/24 dengan IP server SOC;
- 192.168.56.1 dengan gateway;
- DNS dengan kebijakan jaringan masing-masing.

Terapkan Konfigurasi

Jalankan:

```
sudo netplan apply
```

Cek ulang:

```
ip a
ip route
```

Tes koneksi ke gateway:

```
ping -c 4 192.168.56.1
```

Tes koneksi keluar jika lab diizinkan akses internet:

```
ping -c 4 1.1.1.1
```

Tes resolusi DNS:

```
ping -c 4 ubuntu.com
```

Kesalahan Umum Netplan

Masalah	Penyebab Umum	Solusi
Server tidak bisa diakses	Salah interface	Cek ip a
Gateway tidak jalan	Salah via	Cek ip route
DNS gagal	Nameserver salah	Cek /etc/resolv.conf
YAML error	Spasi salah	Gunakan indentasi konsisten
IP bentrok	IP dipakai host lain	Cek jaringan sebelum apply

YAML sangat sensitif terhadap spasi. Jangan memakai tab.

Backup Sebelum Mengubah Jaringan

Sebelum edit Netplan lanjutan:

```
sudo cp -a /etc/netplan /etc/netplan.backup.$(date
+%F-%H%M)
```

Penjelasan:

- cp -a menyalin lengkap beserta atribut.
- $(date +%F-%H%M) menambahkan tanggal dan jam.
- backup berguna jika konfigurasi salah.

2.5 SSH Aman untuk Admin SOC

SSH adalah pintu utama admin ke server SOC.

Jika SSH lemah, server SOC mudah menjadi target.

Ubuntu mendokumentasikan konfigurasi OpenSSH server melalui file `/etc/ssh/sshd_config` atau snippet di `/etc/ssh/sshd_config.d/`. (Ubuntu)

Instalasi SSH Server

Jalankan:

```
sudo apt install -y openssh-server
```

Aktifkan service:

```
sudo systemctl enable --now ssh
```

Cek status:

```
sudo systemctl status ssh
```

Cek port:

```
sudo ss -tulpn | grep ssh
```

Output biasanya menunjukkan port 22.

Backup Konfigurasi SSH

Sebelum edit:

```
sudo cp -a /etc/ssh/sshd_config
/etc/ssh/sshd_config.backup.$(date +%F-%H%M)
```

Edit Konfigurasi SSH

Buka file:

```
sudo nano /etc/ssh/sshd_config
```

Konfigurasi dasar yang disarankan:

```
PermitRootLogin no
PasswordAuthentication yes
PubkeyAuthentication yes
X11Forwarding no
MaxAuthTries 3
ClientAliveInterval 300
ClientAliveCountMax 2
```

Penjelasan:

Parameter	Makna
PermitRootLogin no	Root tidak boleh login langsung
PasswordAuthentication yes	Password masih diizinkan untuk lab awal
PubkeyAuthentication yes	Login berbasis key diizinkan
X11Forwarding no	Fitur grafis jarak jauh dimatikan
MaxAuthTries 3	Batas percobaan login
ClientAliveInterval 300	Cek koneksi idle
ClientAliveCountMax 2	Putuskan sesi idle setelah batas tertentu

Untuk server produksi, arahkan ke autentikasi berbasis key dan batasi sumber IP admin.

Validasi Konfigurasi

Sebelum restart:

```
sudo sshd -t
```

Jika tidak ada output, konfigurasi valid.

Restart SSH:

```
sudo systemctl restart ssh
```

Cek kembali:

```
sudo systemctl status ssh
```

Uji dari Mesin Admin

Dari mesin lain:

```
ssh socadmin@192.168.56.10
```

Jika berhasil, SSH siap.

Catatan Kritis SSH

Jangan mengunci diri sendiri.

Sebelum menutup sesi lama:

- buka sesi SSH baru;
- pastikan login berhasil;
- baru tutup sesi lama.

Kesalahan SSH bisa membuat server tidak bisa diakses dari jarak jauh.

2.6 Firewall Dasar dengan UFW

Firewall host wajib aktif.

UFW adalah alat default Ubuntu untuk mengelola firewall berbasis host dan dirancang agar aturan firewall lebih mudah dikelola dari CLI. (Ubuntu)

Instalasi UFW

Jalankan:

```
sudo apt install -y ufw
```

Set kebijakan default:

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
```

Makna:

- koneksi masuk ditolak;
- koneksi keluar diizinkan.

Izinkan SSH:

```
sudo ufw allow ssh
```

Aktifkan:

```
sudo ufw enable
```

Cek status:

```
sudo ufw status verbose
```

Aturan Firewall Minimum

Untuk Bab 2, cukup buka SSH.

```
22/tcp ALLOW
```

Port lain dibuka pada bab yang relevan.
Contoh nanti:

Layanan	Port Umum	Dibuka Saat
SSH	22/tcp	Bab 2
Wazuh Agent	1514/tcp/udp	Bab 3–4
Wazuh Enrollment	1515/tcp	Bab 3–4
Wazuh API	55000/tcp	Bab 3
OpenSearch	9200/tcp	Bab 7
Dashboard	sesuai konfigurasi	Bab 7

Jangan membuka semua port dari awal.

Firewall yang baik mengikuti kebutuhan, bukan kebiasaan.

Membatasi SSH dari IP Admin

Jika IP admin diketahui:

```
sudo ufw delete allow ssh
sudo ufw allow from 192.168.56.1 to any port 22 proto
tcp
sudo ufw status numbered
```

Makna:

- hapus aturan SSH umum;
- izinkan SSH hanya dari IP admin;
- tampilkan aturan bernomor.

Cek Log UFW

Aktifkan logging:

```
sudo ufw logging on
```

Baca log:

```
sudo journalctl -k | grep UFW
```

Atau:

```
sudo grep UFW /var/log/syslog
```

Log firewall penting untuk SOC.
Firewall bukan hanya alat blokir.

Firewall juga sumber evidence.

2.7 Instalasi Docker dan Compose untuk Tool Pendukung

Beberapa tools SOC pendukung sering lebih mudah dijalankan dalam container, terutama untuk lab.

Docker Compose membantu mendefinisikan banyak service, network, dan volume dalam satu file YAML. Dokumentasi Docker menjelaskan Compose sebagai alat untuk menjalankan aplikasi multi-container dengan satu file konfigurasi. (Docker Documentation)

Catatan Pilihan Instalasi

Ada dua jalur umum:

- paket dari repository distro;
- repository upstream dari proyek Docker.

Outline buku ini memakai paket distro agar sederhana untuk lab:

```
sudo apt install -y docker.io docker-compose-v2
```

Dokumentasi Docker upstream memakai paket Compose plugin bernama docker-compose-plugin, dan perintah modernnya adalah docker compose version. (Docker Documentation)

Untuk buku ini:

- gunakan paket distro untuk lab;
- gunakan dokumentasi resmi masing-masing tool saat masuk instalasi produksi;
- catat versi agar troubleshooting mudah.

Instalasi

Jalankan:

```
sudo apt install -y docker.io docker-compose-v2
```

Aktifkan service:

```
sudo systemctl enable --now docker
```

Tambahkan user ke group docker:

```
sudo usermod -aG docker $USER
```

Penjelasan:

- usermod mengubah konfigurasi user.
- -aG menambahkan user ke group tanpa menghapus group lain.
- docker adalah group yang boleh menjalankan docker.
- $USER adalah user aktif saat ini.

Agar group aktif, logout lalu login kembali.

Atau sementara:

```
newgrp docker
```

Cek versi:

```
docker version
docker compose version
```

Cek service:

```
sudo systemctl status docker
```

Uji Container Sederhana

Jalankan:

```
docker run --rm hello-world
```

Penjelasan:

- docker run menjalankan container.
- --rm menghapus container setelah selesai.
- hello-world adalah image uji.

Jika berhasil, Docker siap.

Catatan Keamanan Docker

Docker memudahkan lab, tetapi punya risiko.

Perhatikan:

- user dalam group `docker` punya akses sangat kuat;
- jangan menjalankan container tidak jelas;
- jangan menaruh token dalam file compose;
- jangan membuka port dashboard ke internet;
- gunakan volume terpisah;
- catat versi image;
- backup file compose.

Container bukan pengganti hardening. Container tetap harus diawasi.

2.8 Struktur Direktori SOC Server

Struktur direktori membuat server mudah dipelihara.
Tanpa struktur, konfigurasi tercecer.

Buat direktori utama:

```
sudo mkdir -p
/opt/soc/{wazuh,opensearch,securityonion,misp,dfir-iris,
shuffle,logs,backup}
```

Ubah kepemilikan:

```
sudo chown -R $USER:$USER /opt/soc
```

Cek:

```
tree /opt/soc
```

Output yang diharapkan:

```
/opt/soc
├── backup
├── dfir-iris
├── logs
├── misp
├── opensearch
├── securityonion
├── shuffle
└── wazuh
```

Fungsi Direktori

Direktori	Fungsi
`/opt/soc/wazuh`	Catatan dan konfigurasi Wazuh
`/opt/soc/opensearch`	Konfigurasi dan catatan indexer
`/opt/soc/securityonion`	Catatan sensor jaringan
`/opt/soc/misp`	Catatan threat intelligence
`/opt/soc/dfir-iris`	Catatan case management
`/opt/soc/shuffle`	Catatan otomasi
`/opt/soc/logs`	Salinan log latihan
`/opt/soc/backup`	Backup konfigurasi penting

Struktur Folder Home Lab

Dari Bab 1, pembaca sudah punya:

```
mkdir -p
~/soc-lab/{docs,configs,logs,evidence,scripts,playbooks}
```

Hubungan dua folder:

Lokasi	Fungsi
`/opt/soc`	Struktur server dan tool
`~/soc-lab`	Dokumentasi, latihan, evidence, script

Prinsip:

- konfigurasi sistem disimpan di `/opt/soc`;
- catatan belajar dan evidence disimpan di `~/soc-lab`;
- backup penting disalin ke dua lokasi jika perlu.

Membuat README Server

Jalankan:

```
nano /opt/soc/README.md
```

Isi:

```
# SOC Server Ubuntu 26.04

## Host
- Hostname: soc-server
- IP: 192.168.56.10
- Timezone: Asia/Jakarta
- Role: SOC server

## Komponen Direncanakan
- Wazuh
- OpenSearch
- Security Onion / Suricata / Zeek
- MISP
- DFIR-IRIS
- Shuffle
- Python scripts

## Catatan Keamanan
- SSH dibatasi
- UFW aktif
- Dashboard tidak diekspos publik
- Backup konfigurasi dibuat
- Semua perubahan dicatat
```

Dokumentasi kecil seperti ini sangat berharga saat troubleshooting.

2.9 Backup Konfigurasi Awal

Backup konfigurasi awal wajib dilakukan sebelum masuk Bab 3.
Buat folder backup:

```
mkdir -p ~/soc-lab/backup
```

Salin konfigurasi jaringan:

```
sudo cp -a /etc/netplan ~/soc-lab/backup/
```

Salin konfigurasi SSH:

```
sudo cp -a /etc/ssh ~/soc-lab/backup/
```

Salin konfigurasi firewall:

```
sudo cp -a /etc/ufw ~/soc-lab/backup/
```

Cek hasil:

```
tree ~/soc-lab/backup
```

Backup dengan Tanggal

Agar lebih rapi:

```
BACKUP_DIR=~/soc-lab/backup/initial-$(date +%F-%H%M)
mkdir -p "$BACKUP_DIR"
sudo cp -a /etc/netplan "$BACKUP_DIR/"
sudo cp -a /etc/ssh "$BACKUP_DIR/"
sudo cp -a /etc/ufw "$BACKUP_DIR/"
sudo chown -R $USER:$USER "$BACKUP_DIR"
tree "$BACKUP_DIR"
```

Penjelasan:

- BACKUP_DIR=... membuat variabel lokasi backup.
- $(date +%F-%H%M) membuat timestamp.
- mkdir -p membuat folder.
- cp -a menyalin lengkap.
- chown -R mengubah kepemilikan agar bisa dibaca user.
- tree menampilkan struktur backup.

Membuat Manifest Backup

Buat file catatan:

```
nano ~/soc-lab/backup/README-backup.txt
```

Isi:

```
Backup Konfigurasi Awal SOC Server

Tanggal:
Hostname:
IP:
Timezone:
Admin:

Isi backup:
- /etc/netplan
- /etc/ssh
- /etc/ufw

Tujuan:
Backup ini dipakai untuk rollback jika konfigurasi
jaringan, SSH,
atau firewall salah setelah instalasi tool SOC.
```

Menghitung Hash File Backup

Untuk evidence dan integritas, gunakan hash.
Contoh:

```
find ~/soc-lab/backup -type f -exec sha256sum {} \; >
~/soc-lab/backup/SHA256SUMS.txt
```

Penjelasan:

- find mencari semua file.
- -type f hanya file.
- sha256sum membuat hash.
- hasil disimpan ke SHA256SUMS.txt.

Cek:

```
cat ~/soc-lab/backup/SHA256SUMS.txt
```

SOC yang baik tidak hanya menyimpan file. SOC yang baik tahu apakah file berubah.

2.10 Hands-on Tambahan: Inventaris Server SOC dengan Python

Python dipakai untuk membantu dokumentasi server.
Buat script:

```
nano ~/soc-lab/scripts/inventaris_server.py
```

Isi:

```
from pathlib import Path
import json
import platform
import socket
import subprocess
from datetime import datetime

def run(command):
    result = subprocess.run(
        command,
        shell=True,
        text=True,
        capture_output=True
    )
    return result.stdout.strip()

data = {
    "timestamp": datetime.now().isoformat(),
    "hostname": socket.gethostname(),
    "os": platform.platform(),
    "kernel": platform.release(),
    "ip_address": run("hostname -I"),
    "timezone": run("timedatectl | grep 'Time zone'"),
    "disk": run("df -h /"),
    "memory": run("free -h"),
    "running_services_sample": run("systemctl --type=service
--state=running --no-pager | head -20"),
    "listening_ports": run("ss -tulpn"),
}

output_dir = Path.home() / "soc-lab" / "docs"
output_dir.mkdir(parents=True, exist_ok=True)

output_file = output_dir / "inventaris-server.json"
output_file.write_text(json.dumps(data, indent=2),
encoding="utf-8")

print(f"Inventaris disimpan ke: {output_file}")
```

Jalankan:

```
python3 ~/soc-lab/scripts/inventaris_server.py
```

Baca hasil:

```
jq . ~/soc-lab/docs/inventaris-server.json
```

Makna Script

Script ini mencatat:

- waktu inventaris;
- hostname;
- OS;
- kernel;
- IP address;
- timezone;
- disk;
- memori;
- sample service aktif;
- port aktif.

Ini menjadi baseline awal sebelum instalasi SOC tools.

Baseline penting karena kita harus tahu kondisi server sebelum berubah.

2.11 Checklist Validasi Akhir Bab 2

Gunakan checklist ini sebelum lanjut ke Bab 3.

Sistem

- Ubuntu Server 26.04 LTS terpasang.
- Hostname benar.
- User admin dibuat.
- Timezone `Asia/Jakarta`.
- Update awal selesai.
- Paket dasar terpasang.
- Server bisa reboot normal.

Jaringan

- Interface dikenali.
- IP static aktif.
- Gateway benar.
- DNS berfungsi.
- Server bisa ping gateway.
- Server bisa akses repository jika internet diizinkan.
- Konfigurasi Netplan dibackup.

SSH

- OpenSSH server aktif.
- Root login langsung dinonaktifkan.
- Konfigurasi SSH valid dengan `sshd -t`.
- Login dari mesin admin berhasil.
- Konfigurasi SSH di backup.

Firewall

- UFW aktif.
- Default incoming deny.
- Default outgoing allow.
- SSH diizinkan.
- Status UFW sudah dicek.
- Konfigurasi UFW di backup.

Container Pendukung

- Docker terpasang.
- Service Docker aktif.
- User admin masuk group docker.
- `docker version` berhasil.
- `docker compose version` berhasil.
- Uji container berhasil.

Dokumentasi

- `/opt/soc` dibuat.
- `~/soc-lab/backup` dibuat.
- README server dibuat.
- Inventaris server dibuat.
- Hash backup dibuat.
- Semua IP, hostname, dan role dicatat.

2.12 Kesalahan Umum Bab 2

1. Tidak Membuat IP Static

Akibat:

- agent gagal terhubung;
- dashboard berubah alamat;
- dokumentasi kacau;
- firewall sulit dikontrol.

SOC server wajib punya alamat tetap.

2. Mengaktifkan Firewall Sebelum Membuka SSH

Akibat:

- admin terkunci dari server;
- perlu akses konsol lokal;
- waktu troubleshooting terbuang.

Urutan benar:

```
allow SSH → enable firewall → test SSH baru
```

3. Salah Format YAML Netplan

YAML gagal karena:

- tab;
- indentasi tidak konsisten;
- nama interface salah;
- gateway salah.

Selalu cek:

```
ip a
ip route
sudo netplan apply
```

4. Tidak Membuat Backup

Tanpa backup, rollback sulit.

Minimal backup:

- `/etc/netplan;`
- `/etc/ssh;`
- `/etc/ufw.`

5. Menjalankan Semua Service Sekaligus

SOC dibangun bertahap.

Jangan langsung menjalankan Wazuh, OpenSearch, sensor, threat intelligence, case management, dan SOAR sekaligus jika server belum stabil.

Stabil dulu. Baru kompleks.

2.13 Rangkuman Bab

Bab ini membangun fondasi server SOC.

Fokusnya bukan teori panjang, tetapi praktik yang rapi:

- menentukan spesifikasi lab;
- memasang Ubuntu Server 26.04;
- melakukan update awal;
- mengatur timezone;
- membuat IP static;
- mengamankan SSH;
- mengaktifkan UFW;
- memasang Docker dan Compose;
- membuat struktur direktori SOC;
- melakukan backup konfigurasi awal;
- membuat inventaris server dengan Python.

Server SOC harus diperlakukan sebagai aset kritikal.
Ia bukan sekadar mesin lab.
Ia adalah pusat log, alert, evidence, triage, dan response.
Jika server ini tidak rapi, seluruh SOC akan sulit dipercaya.

Output Bab

Setelah menyelesaikan Bab 2, pembaca memiliki:

- Ubuntu Server 26.04 LTS aktif;
- hostname dan user admin yang jelas;
- timezone `Asia/Jakarta`;
- IP static;
- SSH aktif dan lebih aman;
- firewall UFW aktif;
- Docker dan Compose siap untuk tool pendukung;
- struktur direktori `/opt/soc`;
- backup konfigurasi awal;
- inventaris server dalam JSON;
- baseline sistem sebelum instalasi Wazuh.

Kalimat kunci Bab 2:

> **SOC yang kuat dimulai dari server yang stabil, IP yang jelas, SSH yang aman, firewall yang terkendali, dan dokumentasi yang bisa dipercaya.**

Bab 3 — Wazuh Server: SIEM/XDR, Agent, dan Alert

Bab 3 yang menempatkan Wazuh sebagai fondasi SIEM/XDR, agent, alert, CLI, dan API. Dalam dokumen terlampir, target Bab 3 sangat jelas: **Wazuh server berjalan, agent pertama terhubung, alert bisa dibaca melalui dashboard dan CLI**.

Wazuh resmi menjelaskan bahwa model *all-in-one deployment* memasang server, indexer, dan dashboard pada satu server, cocok untuk lab dan lingkungan kecil. (Dokumentasi Wazuh) Dokumentasi *quickstart* Wazuh juga menyediakan instalasi komponen sentral dalam satu host memakai *installation assistant*. (Dokumentasi Wazuh) Bab ini tidak hanya mengejar "berhasil instal", tetapi memastikan pembaca memahami service, file, log, agent, alert, dan API.

3.1 Arsitektur Wazuh

Wazuh bukan satu program tunggal. Wazuh adalah kumpulan komponen yang bekerja sebagai sistem monitoring, deteksi, dan respons.

Komponen utama:

- **Wazuh Manager**
- **Wazuh Indexer**
- **Wazuh Dashboard**
- **Wazuh Agent**
- **Rules**
- **Decoders**
- **Alerts**
- **API**

Alur Kerja Dasar Wazuh

```
Endpoint Linux
→ Wazuh Agent
→ Wazuh Manager
→ Rules / Decoders
→ Alerts
→ Wazuh Indexer
→ Wazuh Dashboard
→ Triage
→ Report
```

Makna alur:

- endpoint menghasilkan log;
- agent mengirim data;
- manager menganalisis;
- decoder membaca struktur log;
- rule menentukan apakah event penting;
- alert dibuat;
- indexer menyimpan dan membuat data dapat dicari;
- dashboard membantu visualisasi;
- analis tetap membaca log untuk pembuktian.

Dashboard mempercepat. CLI membuktikan.

Wazuh Manager

Wazuh Manager adalah pusat analisis.

Fungsinya:

- menerima data dari agent;
- memproses log;
- menjalankan decoder;
- menjalankan rule;
- membuat alert;
- mengelola agent;
- menyediakan API;
- menjadi pusat kendali deteksi.

Dokumentasi Wazuh menjelaskan bahwa Wazuh server menganalisis data yang diterima dari agent dan memicu alert ketika ancaman terdeteksi. (Dokumentasi Wazuh)

Wazuh Indexer

Wazuh Indexer menyimpan dan mengindeks data keamanan.

Fungsinya:

- menyimpan alert;
- membuat alert bisa dicari;
- mendukung query;
- mendukung dashboard;
- menyimpan data untuk investigasi.

Indexer adalah komponen berat. Karena itu, Bab 2 sudah menekankan RAM, CPU, storage, dan struktur direktori.

Dokumentasi Wazuh menyebut indexer dapat dipasang sebagai single-node atau multi-node cluster, dengan rekomendasi hardware berbeda untuk kebutuhan produksi. (Dokumentasi Wazuh)

Wazuh Dashboard

Wazuh Dashboard adalah antarmuka visual.

Fungsinya:

- melihat status agent;
- melihat alert;
- mencari event;
- melihat severity;
- melihat FIM;
- melihat vulnerability;
- membantu triage.

Namun dashboard bukan pengganti log.

Alert yang tampil di dashboard tetap harus divalidasi melalui CLI.

Wazuh Agent

Wazuh Agent dipasang di endpoint.

Agent mengirim:

- log autentikasi;
- event sistem;
- perubahan file;
- informasi paket;
- informasi konfigurasi;
- informasi proses;
- event keamanan.

Agent adalah mata dan telinga SOC di endpoint.

Tanpa agent, server SOC buta terhadap aktivitas host.

Rules

Rules adalah logika deteksi.

Contoh rule:

- login gagal;
- perubahan file penting;
- user baru dibuat;
- sudo gagal;
- paket rentan;
- aktivitas mencurigakan.

Rule menentukan apakah event biasa naik menjadi alert.

Decoders

Decoders membaca format log.

Log mentah:

```
Failed password for invalid user admin from 192.0.2.10
port 53422 ssh2
```

Decoder membantu memecah log menjadi field:

- user;
- source IP;
- port;
- program;
- waktu;
- jenis event.

Tanpa decoder, rule sulit bekerja dengan akurat.

Alerts

Alerts adalah hasil deteksi.

Alert bukan otomatis incident.

Alert harus di triage.

Pertanyaan dasar:

- alert apa?
- level berapa?
- agent mana?
- user siapa?
- IP sumber mana?
- waktu kejadian?
- log mentahnya apa?
- ada bukti tambahan?

API

Wazuh API memungkinkan admin membaca dan mengelola Wazuh dari CLI atau script.

API dapat dipakai untuk:

- mengambil token;
- membaca daftar agent;
- membaca status manager;
- query event;
- integrasi otomasi;
- membuat script Python.

Wazuh API memakai JWT token; dokumentasi resminya menyarankan parameter `raw=true` saat autentikasi via `curl` agar token lebih mudah dipakai. (Dokumentasi Wazuh)

3.2 Instalasi Wazuh Single Node di Ubuntu 26.04

Pada lab buku ini, gunakan pendekatan **single node / all-in-one**.

Artinya:

- Wazuh Manager;
- Wazuh Indexer;
- Wazuh Dashboard;

berjalan di satu server.

Ini cocok untuk:

- lab pribadi;
- kelas;
- pelatihan;
- proof-of-concept;
- mini SOC;
- portofolio teknis.

Untuk produksi besar, pisahkan komponen dan ikuti desain kapasitas yang lebih serius.

Catatan Penting Sebelum Instalasi

Pastikan Bab 2 sudah selesai:

- Ubuntu Server 26.04 aktif;
- IP static benar;
- hostname benar;
- timezone Asia/Jakarta;
- SSH aktif;
- UFW aktif;
- backup konfigurasi dibuat;
- storage cukup;
- RAM cukup.

Cek ulang:

```
hostnamectl
timedatectl
ip a
ip route
df -h
free -h
sudo ufw status verbose
```

Masuk ke Folder Kerja

```
mkdir -p /opt/soc/wazuh
cd /opt/soc/wazuh
```

Penjelasan:

- /opt/soc/wazuh menjadi folder catatan instalasi Wazuh.
- semua file installer dan catatan disimpan rapi.

Mengunduh Installer Resmi Wazuh

Gunakan installer resmi dari dokumentasi Wazuh.

```
curl -sO
https://packages.wazuh.com/4.14/wazuh-install.sh
```

Catatan:

- cek versi terbaru di dokumentasi resmi sebelum praktik;
- contoh ini memakai jalur dokumentasi Wazuh 4.x;
- jangan mengambil installer dari sumber tidak jelas.

Ubah permission:

```
chmod +x wazuh-install.sh
```

Penjelasan:

- chmod +x membuat file bisa dieksekusi.

Menjalankan Instalasi All-in-One

```
sudo ./wazuh-install.sh -a
```

Penjelasan:

- sudo menjalankan sebagai admin.
- ./wazuh-install.sh menjalankan script installer.
- -a mengaktifkan mode all-in-one sesuai quickstart Wazuh.

Dokumentasi Wazuh menyatakan quickstart memasang komponen sentral Wazuh pada host yang sama memakai installation assistant. (Dokumentasi Wazuh)

Simpan Output Instalasi

Saat instalasi selesai, catat:

- URL dashboard;
- user admin;
- password;
- versi komponen;
- lokasi file kredensial;
- error jika ada.

Buat catatan:

```
nano ~/soc-lab/docs/catatan-instalasi-wazuh.txt
```

Isi awal:

```
# Catatan Instalasi Wazuh

Tanggal:
Hostname:
IP SOC Server:
Versi Wazuh:
Mode instalasi: all-in-one / single host

Komponen:
- Wazuh Manager:
- Wazuh Indexer:
- Wazuh Dashboard:

Akses Dashboard:
URL:
User:
Password:
Catatan error:
Solusi:
```

Jangan hanya screenshot password. Tulis catatan teknis yang rapi dan simpan aman.

Catatan Keamanan Dashboard

Untuk lab lokal:

- dashboard boleh diakses dari jaringan lab;
- jangan buka dashboard ke internet;
- gunakan firewall;
- ganti password default;
- batasi IP admin;
- catat siapa yang punya akses.

SOC yang baru dipasang sering justru menjadi target karena dashboard dibuka terlalu bebas.

3.3 Cek Service Wazuh

Instalasi belum selesai hanya karena script selesai.

Admin harus memeriksa service.

Cek Wazuh Manager

```
sudo systemctl status wazuh-manager
```

Yang dicari:

- status active;
- tidak ada error fatal;
- service berjalan setelah reboot.

Cek Wazuh Indexer

```
sudo systemctl status wazuh-indexer
```

Yang dicari:

- status active;
- indexer tidak crash;
- memori cukup;
- port berjalan.

Cek Wazuh Dashboard

```
sudo systemctl status wazuh-dashboard
```

Yang dicari:

- status active;
- dashboard bisa menerima koneksi;
- tidak ada error konfigurasi.

Cek Port Aktif

```
sudo ss -tulpn
```

Atau filter:

```
sudo ss -tulpn | grep -E '1514|1515|55000|9200|443|5601'
```

Port umum yang perlu dipahami:

Port	Fungsi Umum
1514	Komunikasi agent ke manager
1515	Enrollment agent
55000	Wazuh API
9200	Indexer API
443 / port dashboard	Akses dashboard

Catatan:

- port bisa berbeda tergantung konfigurasi;
- jangan buka semua port ke semua jaringan;
- gunakan firewall sesuai kebutuhan.

Cek Semua Service Wazuh

```
systemctl list-units --type=service | grep wazuh
```

Cek Resource

```
free -h
df -h
htop
```

Wazuh Indexer dapat memakai resource cukup besar.
Jika server lambat:

- cek RAM;
- cek disk;
- cek CPU;
- cek log service;
- kurangi VM lain;
- tambah resource.

Restart Service Jika Perlu

```
sudo systemctl restart wazuh-manager
sudo systemctl restart wazuh-indexer
sudo systemctl restart wazuh-dashboard
```

Gunakan restart setelah paham penyebab masalah.
Jangan restart membabi buta saat investigasi.

3.4 Lokasi File Penting Wazuh

Admin SOC wajib tahu lokasi file.

Dashboard membantu, tetapi troubleshooting sering selesai dari CLI.

Direktori Utama

```
sudo ls -lah /var/ossec/
```

Isi penting biasanya mencakup:

- konfigurasi;
- log;
- ruleset;
- binary;
- queue;
- integrasi.

Konfigurasi

```
sudo ls -lah /var/ossec/etc/
```

File penting:

```
/var/ossec/etc/ossec.conf
```

Fungsi:

- konfigurasi manager;
- konfigurasi global;
- modul;
- log collection;
- FIM;
- integrasi.

Log Wazuh

```
sudo ls -lah /var/ossec/logs/
```

Lokasi penting:

```
/var/ossec/logs/ossec.log
/var/ossec/logs/alerts/alerts.log
/var/ossec/logs/alerts/alerts.json
```

Makna:

File	Fungsi
`ossec.log`	Log operasional Wazuh
`alerts.log`	Alert format teks
`alerts.json`	Alert format JSON

Ruleset Bawaan

```
sudo ls -lah /var/ossec/ruleset/
```

Direktori ini berisi ruleset bawaan.
Jangan sembarang mengubah file bawaan.

Rule Lokal

```
sudo ls -lah /var/ossec/etc/rules/
```

Lokasi rule lokal:

```
/var/ossec/etc/rules/local_rules.xml
```

Rule lokal dibahas lebih dalam pada Bab 8.

Decoder Lokal

```
sudo ls -lah /var/ossec/etc/decoders/
```

Lokasi decoder lokal:

```
/var/ossec/etc/decoders/local_decoder.xml
```

Decoder juga dibahas lebih dalam pada Bab 8.

Membuat Catatan Lokasi File

```
nano ~/soc-lab/docs/lokasi-file-wazuh.txt
```

Isi:

```
# Lokasi File Penting Wazuh

Direktori utama:
- /var/ossec/

Konfigurasi:
- /var/ossec/etc/ossec.conf

Log operasional:
- /var/ossec/logs/ossec.log

Alert:
- /var/ossec/logs/alerts/alerts.log
- /var/ossec/logs/alerts/alerts.json

Rule lokal:
- /var/ossec/etc/rules/local_rules.xml

Decoder lokal:
- /var/ossec/etc/decoders/local_decoder.xml
```

Catatan:

Jangan mengubah rule bawaan tanpa backup.
Gunakan rule lokal untuk custom detection.

3.5 Akses Log Wazuh via CLI

Bagian ini inti Bab 3.

Admin harus bisa membaca log tanpa bergantung dashboard.

Membaca Log Operasional

```
sudo tail -f /var/ossec/logs/ossec.log
```

Penjelasan:

- tail membaca bagian akhir file.
- -f mengikuti perubahan file secara real-time.
- ossec.log berisi aktivitas service Wazuh.

Gunakan ini saat:

- agent gagal connect;
- manager restart;
- konfigurasi error;
- rule bermasalah;
- service tidak normal.

Membaca Alert Format Teks

```
sudo tail -f /var/ossec/logs/alerts/alerts.log
```

Format teks mudah dibaca manusia.
Cocok untuk pemula.

Membaca Alert Format JSON

```
sudo tail -f /var/ossec/logs/alerts/alerts.json
```

Format JSON cocok untuk:

- filtering;
- parsing;
- API;
- otomasi;
- laporan;
- Python.

Membaca JSON dengan jq

```
sudo jq . /var/ossec/logs/alerts/alerts.json | less
```

Penjelasan:

- jq . merapikan JSON.
- less membuat output bisa di scroll.

Filter Alert Berdasarkan Level

```
sudo jq 'select(.rule.level >= 10)'
/var/ossec/logs/alerts/alerts.json
```

Makna:

- tampilkan alert dengan level 10 ke atas;
- cocok untuk mencari alert prioritas tinggi.

Ambil Field Penting

```
sudo jq '{time: .timestamp, agent: .agent.name, level:
.rule.level, description: .rule.description}'
/var/ossec/logs/alerts/alerts.json
```

Output contoh:

```
{
  "time": "2026-06-02T10:10:00.000+0700",
  "agent": "endpoint-01",
  "level": 10,
  "description": "Multiple authentication failures"
}
```

Cari Alert dari Agent Tertentu

```
sudo jq 'select(.agent.name=="endpoint-01")'
/var/ossec/logs/alerts/alerts.json
```

Cari Source IP

```
sudo jq 'select(.data.srcip=="192.0.2.10")'
/var/ossec/logs/alerts/alerts.json
```

Field bisa berbeda tergantung decoder.
Selalu lihat struktur JSON asli.

Simpan Alert Level Tinggi ke Evidence

```
mkdir -p ~/soc-lab/evidence

sudo jq 'select(.rule.level >= 10)'
/var/ossec/logs/alerts/alerts.json | tee
~/soc-lab/evidence/wazuh-high-alerts.json
```

Penjelasan:

- mkdir -p membuat folder evidence.
- jq memfilter alert.
- tee menampilkan sekaligus menyimpan output.

Evidence harus disimpan sebelum respons.

3.6 Menambahkan Wazuh Agent Linux

Agent pertama adalah bukti bahwa SOC mulai hidup.

Target:

- endpoint terpasang agent;
- agent connect ke manager;
- server bisa melihat agent;
- alert bisa muncul.

Di Endpoint: Cek Status Agent

Setelah instalasi agent sesuai dokumentasi resmi Wazuh, cek:

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

Cek log agent:

```
sudo tail -f /var/ossec/logs/ossec.log
```

Yang dicari:

- agent berhasil start;
- agent menemukan manager;
- tidak ada error koneksi;
- tidak ada masalah key/enrollment.

Catatan Instalasi Agent

Instalasi agent biasanya membutuhkan:

- alamat IP Wazuh Manager;
- enrollment;
- paket agent;
- restart service.

Contoh variabel penting:

```
WAZUH_MANAGER="192.168.56.10"
```

Ikuti perintah resmi dari dokumentasi Wazuh sesuai versi yang dipakai. Jangan menyalin perintah lama tanpa cek versi.

Di Server: Cek Daftar Agent

```
sudo /var/ossec/bin/agent_control -l
```

Perintah ini menampilkan agent yang terdaftar.
Cek agent aktif:

```
sudo /var/ossec/bin/agent_control -lc
```

Makna umum:

- -l menampilkan daftar agent;
- -lc menampilkan agent yang aktif/terhubung.

Manage Agents

```
sudo /var/ossec/bin/manage_agents
```

Tool ini dipakai untuk manajemen agent secara interaktif.
Namun untuk lab modern, enrollment dan dashboard sering lebih nyaman. Tetap penting memahami CLI karena troubleshooting sering memakai terminal.

Troubleshooting Agent Pertama

Masalah umum:

Gejala	Kemungkinan Penyebab	Cek
Agent disconnected	IP manager salah	`grep manager /var/ossec/etc/ossec.conf`
Agent tidak muncul	Enrollment gagal	`agent_control -l`
Port tertutup	Firewall	`sudo ufw status`
Waktu tidak sinkron	Timezone / NTP	`timedatectl`
Service mati	Agent error	`systemctl status wazuh-agent`
DNS gagal	Resolver salah	`ping, resolvectl`

Tes koneksi dari endpoint ke server:

```
ping -c 4 192.168.56.10
```

Cek port enrollment:

```
nc -vz 192.168.56.10 1515
```

Cek port komunikasi agent:

```
nc -vz 192.168.56.10 1514
```

Jika nc belum ada:

```
sudo apt install -y netcat-openbsd
```

3.7 Membaca Agent dari CLI dan Dashboard

Agent bukan hanya “connected”.
Admin harus memahami data agent.

CLI:

```
sudo /var/ossec/bin/agent_control -lc
```

Data yang harus dipahami:

- **agent ID**
- **hostname**
- **IP address**
- **status active / disconnected**
- **last keep alive**

Agent ID

Agent ID adalah identitas agent di Wazuh.
Contoh:

```
ID: 001
```

Agent ID dipakai saat:

- membaca alert;
- mencari host;
- troubleshooting;
- grouping agent;
- membuat laporan.

Hostname

Hostname harus konsisten dengan Bab 2.

Contoh:

```
endpoint-01
```

Jika hostname tidak jelas, laporan incident akan membingungkan.

IP Address

IP agent penting untuk korelasi.
Contoh:

```
192.168.56.20
```

Catat IP di:

```
nano ~/soc-lab/docs/daftar-agent-wazuh.txt
```

Isi:

```
# Daftar Agent Wazuh
```

Agent ID	Hostname	IP	Role	Status	Catatan
001	endpoint-01	192.168.56.20	endpoint lab	active	agent pertama

Status Active / Disconnected

Status `active` berarti agent sedang terhubung.

Status `disconnected` perlu investigasi.

Pertanyaan:

- endpoint hidup?
- agent service aktif?
- IP berubah?
- firewall blocking?
- waktu sinkron?
- manager aktif?

Last Keep Alive

Last keep alive menunjukkan kapan agent terakhir berkomunikasi.

Jika terlalu lama:

- agent mungkin mati;
- endpoint mati;
- jaringan putus;
- firewall menolak;
- service error.

Membaca dari Dashboard

Di dashboard, cek:

- daftar agent;
- status;
- alert per agent;
- severity;
- FIM;
- inventory;
- vulnerability.

Namun dashboard harus dipakai sebagai peta, bukan satu-satunya bukti.

Selalu cocokkan dengan CLI:

```
sudo /var/ossec/bin/agent_control -lc
sudo tail -f /var/ossec/logs/alerts/alerts.json
```

3.8 Wazuh API dengan curl

API membuat Wazuh bisa diotomasi.

Bab ini mulai dari `curl`.

Bab berikutnya bisa berkembang ke Python.

Autentikasi API

Contoh dasar dari outline:

```
curl -k -u wazuh:wazuh
https://localhost:55000/security/user/authenticate
```

Namun dokumentasi Wazuh menyarankan penggunaan raw=true agar token dikembalikan sebagai teks mentah dan mudah disimpan ke variabel. (Dokumentasi Wazuh)

Gunakan:

```
TOKEN=$(curl -k -u wazuh:wazuh -X POST
"https://localhost:55000/security/user/authenticate?raw=
true")
```

Penjelasan:

- TOKEN=$(...) menyimpan output ke variabel.
- curl melakukan request HTTP.
- -k menerima sertifikat self-signed lab.
- -u wazuh:wazuh memakai basic auth.
- -X POST memakai metode POST.
- raw=true membuat token lebih mudah dipakai.

Cek token:

```
echo $TOKEN
```

Jangan tampilkan token di layar publik.

Query Informasi Manager

```
curl -k -X GET
"https://localhost:55000/manager/info?pretty=true" \
  -H "Authorization: Bearer $TOKEN"
```

Makna:

- API dipanggil ke Wazuh Manager;
- token dipakai sebagai bearer token;
- pretty=true merapikan output JSON.

Query Daftar Agent

```
curl -k -X GET
"https://localhost:55000/agents?pretty=true" \
  -H "Authorization: Bearer $TOKEN"
```

Filter dengan jq:

```
curl -k -X GET
"https://localhost:55000/agents?pretty=true" \
  -H "Authorization: Bearer $TOKEN" | jq .
```

Ambil Field Agent Penting

```
curl -k -X GET
"https://localhost:55000/agents?pretty=true" \
  -H "Authorization: Bearer $TOKEN" \
| jq '.data.affected_items[] | {id: .id, name: .name,
ip: .ip, status: .status}'
```

Output contoh:

```
{
  "id": "001",
  "name": "endpoint-01",
  "ip": "192.168.56.20",
  "status": "active"
}
```

Ekspor Daftar Agent ke JSON

```
mkdir -p ~/soc-lab/evidence/api
curl -k -X GET
"https://localhost:55000/agents?pretty=true" \
  -H "Authorization: Bearer $TOKEN" \
  -o ~/soc-lab/evidence/api/wazuh-agents.json
```

Cek:

```
jq . ~/soc-lab/evidence/api/wazuh-agents.json | less
```

Catatan Token

JWT token Wazuh API punya masa berlaku. Dokumentasi API menyebut token JWT memiliki durasi default 900 detik. (Dokumentasi Wazuh)

Jika request gagal karena token kadaluarsa, ambil token baru.

Script Python Ringan untuk Query Agent

Buat script:

```
nano ~/soc-lab/scripts/wazuh_list_agents.py
```

Isi:

```
import json
import ssl
import urllib.request
import base64

WAZUH_URL = "https://localhost:55000"
USERNAME = "wazuh"
PASSWORD = "wazuh"

ctx = ssl._create_unverified_context()

auth =
base64.b64encode(f"{USERNAME}:{PASSWORD}".encode()).deco
de()

req = urllib.request.Request(
    f"{WAZUH_URL}/security/user/authenticate?raw=true",
    method="POST",
    headers={"Authorization": f"Basic {auth}"}
)

with urllib.request.urlopen(req, context=ctx) as
response:
```

```
        token = response.read().decode().strip()

    req = urllib.request.Request(
        f"{WAZUH_URL}/agents?pretty=true",
        headers={"Authorization": f"Bearer {token}"}
    )

    with urllib.request.urlopen(req, context=ctx) as
    response:
        data = json.loads(response.read().decode())

    for agent in data["data"]["affected_items"]:
        print(agent.get("id"), agent.get("name"),
    agent.get("ip"), agent.get("status"))
```

Jalankan:

```
python3 ~/soc-lab/scripts/wazuh_list_agents.py
```

Catatan:

- script ini untuk lab;
- tidak memakai library tambahan;
- sertifikat self-signed dilewati dengan konteks SSL tidak diverifikasi;
- untuk produksi, validasi sertifikat harus benar.

3.9 Template Catatan Instalasi Wazuh

Dokumentasi adalah bagian dari SOC.
Tanpa catatan, instalasi sulit diaudit.

Buat template:

```
nano ~/soc-lab/docs/template-catatan-instalasi-wazuh.txt
```

Isi:

```
# Catatan Instalasi Wazuh

## 1. Identitas Server
Tanggal instalasi:
Nama admin:
Hostname:
IP server:
Timezone:
OS:
Kernel:

## 2. Versi Komponen
Wazuh Manager:
Wazuh Indexer:
Wazuh Dashboard:
Wazuh Agent pertama:

## 3. Mode Instalasi
[ ] All-in-one
[ ] Multi-node
[ ] Lab
[ ] Produksi

Catatan mode:

## 4. Port Aktif
Port:
Service:
Scope akses:
Catatan firewall:

## 5. Lokasi File Penting
Direktori utama:
Konfigurasi:
Log manager:
Alert teks:
Alert JSON:
Rule lokal:
Decoder lokal:
```

```
## 6. Daftar Agent
Agent ID:
Hostname:
IP:
Role:
Status:
Last keep alive:

## 7. Akses Dashboard
URL:
User admin:
Lokasi penyimpanan kredensial:
Catatan perubahan password:

## 8. API
URL API:
Port:
Metode autentikasi:
Catatan token:

## 9. Error Saat Instalasi
Error:
Waktu:
Log terkait:
Analisis:
Solusi:
Status:

## 10. Validasi Akhir
[ ] Service manager aktif
[ ] Service indexer aktif
[ ] Service dashboard aktif
[ ] Port dicek
[ ] Agent pertama terhubung
[ ] Alert JSON bisa dibaca
[ ] API bisa diakses
[ ] Backup konfigurasi dibuat
```

Backup Catatan dan Konfigurasi

```
mkdir -p ~/soc-lab/backup/wazuh
sudo cp -a /var/ossec/etc ~/soc-lab/backup/wazuh/
sudo cp -a /var/ossec/logs/ossec.log
~/soc-lab/backup/wazuh/ossec.log.initial 2>/dev/null || true
sudo chown -R $USER:$USER ~/soc-lab/backup/wazuh
```

Penjelasan:

- 2>/dev/null || true mencegah perintah gagal jika file belum ada;
- backup awal berguna sebelum custom rule di Bab 8.

Hash Backup

```
find ~/soc-lab/backup/wazuh -type f -exec sha256sum {} \; >
~/soc-lab/backup/wazuh/SHA256SUMS.txt
```

3.10 Hands-on: Membuat Alert Pertama

Setelah agent terhubung, buat event sederhana dari endpoint.
Di endpoint:

```
sudo -k
sudo ls /root
```

Masukkan password salah beberapa kali jika aman di lab.
Atau cek autentikasi SSH dari host lab lain:

```
ssh user_salah@192.168.56.20
```

Lakukan hanya pada lab sendiri.
Di Wazuh server:

```
sudo tail -f /var/ossec/logs/alerts/alerts.json | jq .
```

Cari alert:

```
sudo jq 'select(.rule.description |
test("authentication|login|password"; "i"))'
/var/ossec/logs/alerts/alerts.json
```

Simpan evidence:

```
mkdir -p ~/soc-lab/evidence/wazuh
sudo jq 'select(.rule.description |
test("authentication|login|password"; "i"))' \
  /var/ossec/logs/alerts/alerts.json \
  | tee ~/soc-lab/evidence/wazuh/auth-alerts.json
```

Buat catatan:

```
nano ~/soc-lab/evidence/wazuh/catatan-alert-pertama.txt
```

Isi:

```
# Catatan Alert Pertama Wazuh

Tanggal:
Analis:
Endpoint:
IP endpoint:
Skenario:
Waktu simulasi:

Alert:
Rule:
Level:
Deskripsi:
User:
Source IP:

Bukti:
```

```
- alerts.json:
- auth.log:
- dashboard:
```

Kesimpulan:

```
Alert berhasil muncul dan dapat dibaca dari CLI.
```

Tujuan praktik ini bukan menyerang. Tujuannya membuktikan alur log → alert → evidence.

3.11 Checklist Validasi Akhir Bab 3

Service

- `wazuh-manager` aktif.
- `wazuh-indexer` aktif.
- `wazuh-dashboard` aktif.
- Port dicek dengan `ss`.
- Resource server dicek.

File dan Log

- `/var/ossec/` dikenali.
- `/var/ossec/etc/` dikenali.
- `/var/ossec/logs/ossec.log` bisa dibaca.
- `/var/ossec/logs/alerts/alerts.log` bisa dibaca.
- `/var/ossec/logs/alerts/alerts.json` bisa dibaca.
- `jq` bisa memfilter alert.

Agent

- Agent Linux terpasang.
- Agent service aktif.
- Agent muncul di server.
- Agent status `active`.
- Agent ID dicatat.
- Hostname dan IP dicatat.

API

- API bisa autentikasi.
- Token berhasil diambil.
- Query manager berhasil.
- Query agent berhasil.
- Output API bisa dibaca dengan `jq`.
- Export JSON berhasil.

Dokumentasi

- Catatan instalasi Wazuh dibuat.
- Lokasi file penting dicatat.
- Daftar agent dibuat.
- Evidence alert pertama disimpan.
- Backup konfigurasi awal Wazuh dibuat.
- Hash backup dibuat.

3.12 Kesalahan Umum Bab 3

1. Menganggap Instalasi Selesai Karena Dashboard Terbuka

Dashboard terbuka belum tentu SOC siap.

Harus dicek:

```
sudo systemctl status wazuh-manager
sudo systemctl status wazuh-indexer
sudo systemctl status wazuh-dashboard
sudo ss -tulpn
```

2. Tidak Membaca alerts.json

Banyak pemula hanya melihat tampilan visual.

Padahal alerts.json adalah sumber penting untuk:

- filtering;
- automation;
- evidence;
- export;
- Python;
- SOAR.

3. Agent Terhubung Tetapi Tidak Dipahami

Agent active belum cukup.

Admin harus mencatat:

- agent ID;
- hostname;
- IP;
- role;
- status;
- last keep alive.

4. Token API Dibocorkan

Token API adalah akses.
Jangan tampilkan token di screenshot publik.
Jangan simpan token di file terbuka.

5. Tidak Backup Sebelum Custom Rule

Bab 8 akan mengubah rule dan decoder.
Sebelum itu, backup konfigurasi awal harus ada.

6. Membuka Dashboard Terlalu Luas

Dashboard SOC adalah aset sensitif.
Batasi akses.
Gunakan firewall.
Jangan ekspos ke jaringan publik.

3.13 Rangkuman Bab

Bab ini membangun fondasi Wazuh sebagai pusat SIEM/XDR mini SOC.

Pembaca tidak hanya menjalankan installer, tetapi memahami:

- Wazuh Manager sebagai pusat analisis;
- Wazuh Indexer sebagai penyimpan dan pencari data;
- Wazuh Dashboard sebagai visualisasi;
- Wazuh Agent sebagai sumber telemetry endpoint;
- Rules dan decoders sebagai logika deteksi;
- Alerts sebagai sinyal investigasi;
- API sebagai pintu otomasi;
- CLI sebagai alat pembuktian.

SOC yang matang tidak berhenti pada “service aktif”.

SOC yang matang mampu menjawab:

- agent mana yang mengirim alert;
- alert berasal dari log apa;
- rule apa yang memicu;
- level berapa;
- bukti mentahnya di mana;
- apakah perlu triage;
- apakah layak menjadi incident.

Output Bab

Setelah menyelesaikan Bab 3, pembaca memiliki:

- Wazuh server berjalan;
- Wazuh Manager aktif;
- Wazuh Indexer aktif;
- Wazuh Dashboard aktif;
- agent Linux pertama terhubung;
- alert dapat dibaca dari CLI;
- `alerts.json` dapat difilter dengan `jq`;
- Wazuh API dapat diakses dengan `curl`;
- daftar agent terdokumentasi;
- evidence alert pertama tersimpan;
- template catatan instalasi Wazuh siap dipakai.

Kalimat kunci Bab 3:

> **Wazuh bukan sekadar dashboard SOC; Wazuh adalah mesin pembuktian yang mengubah log endpoint menjadi alert yang bisa dibaca, difilter, diaudit, dan dijadikan dasar respons.**

Bab 4 — Mengelola Agent dan Endpoint Linux

Bab 4 yang menempatkan agent sebagai komponen utama endpoint monitoring, meliputi instalasi agent, konfigurasi manager IP, pengecekan service, grouping agent, log collection, FIM, simulasi perubahan file, dan troubleshooting.

Secara teknis, dokumentasi resmi Wazuh menjelaskan bahwa Wazuh Agent berjalan di endpoint yang dimonitor dan mengirim data ke Wazuh Manager secara hampir real-time melalui kanal terenkripsi dan terautentikasi. (Dokumentasi Wazuh) Wazuh juga mendukung konfigurasi agent secara lokal melalui `/var/ossec/etc/ossec.conf` maupun terpusat melalui `agent.conf`, termasuk untuk *file integrity monitoring*, *log data collection*, *system inventory*, dan konfigurasi per group. (Dokumentasi Wazuh)

4.1 Agent sebagai Mata dan Telinga SOC

SOC server tidak bisa melihat endpoint tanpa agent.

> **Agent adalah mata dan telinga SOC di mesin Linux.**

Ia membaca aktivitas lokal, mengirim telemetry, lalu Wazuh Manager menganalisisnya menjadi event dan alert.

Agent mengirim:

- **log autentikasi**;
- **perubahan file**;
- **informasi paket**;
- **konfigurasi sistem**;
- **proses**;
- **security event**.

Tanpa agent, SOC hanya melihat sisi server.

Dengan agent, SOC bisa melihat apa yang terjadi di endpoint.

Data Penting dari Agent

Data	Contoh	Nilai SOC
Log autentikasi	login gagal, sudo gagal	Deteksi brute force
Perubahan file	/etc/passwd berubah	Deteksi perubahan akun
Informasi paket	daftar paket terpasang	Deteksi kerentanan
Konfigurasi sistem	SSH config, user, service	Audit keamanan
Proses	proses aktif	Deteksi aktivitas aneh
Security event	rule match, FIM, SCA	Alert dan triage

Prinsip Penting

Endpoint adalah tempat serangan sering terlihat pertama kali.

Contoh:

- attacker mencoba SSH;
- user privilege berubah;
- file konfigurasi dimodifikasi;
- paket rentan terpasang;
- service penting mati;
- script mencurigakan dijalankan.

Agent membantu SOC menjawab:

- host mana yang bermasalah?
- user siapa yang terlibat?
- kapan kejadian terjadi?
- file apa yang berubah?
- apakah alert muncul berulang?
- apakah endpoint masih aktif?

SOC tanpa agent seperti admin jaringan tanpa log endpoint.

4.2 Instalasi Agent di Endpoint Ubuntu 26.04

Instalasi agent harus mengikuti dokumentasi resmi sesuai versi Wazuh yang digunakan. Dokumentasi Wazuh menyediakan deployment variables untuk memudahkan instalasi, pendaftaran, dan konfigurasi agent Linux. (Dokumentasi Wazuh)

Topik utama:

- repository;
- package install;
- konfigurasi manager IP;
- enrollment;
- restart service.

Prasyarat Endpoint

Di endpoint Ubuntu 26.04, pastikan:

- hostname jelas;
- IP benar;
- bisa ping ke Wazuh Manager;
- waktu sinkron;
- DNS berfungsi jika memakai repository internet;
- firewall tidak memblokir komunikasi ke server SOC.

Cek hostname:

```
hostnamectl
```

Cek IP:

```
ip a
```

Cek route:

```
ip route
```

Cek waktu:

```
timedatectl
```

Tes koneksi ke Wazuh Manager:

```
ping -c 4 <IP_WAZUH_SERVER>
```

Contoh:

```
ping -c 4 192.168.56.10
```

Konsep Manager IP

Agent harus tahu alamat Wazuh Manager.
Contoh:

```
WAZUH_MANAGER="192.168.56.10"
```

Jika IP salah, agent tidak akan terkoneksi.

Kesalahan paling sering pada agent adalah salah IP manager.

Contoh Instalasi Agent Linux

Contoh alur umum:

```
curl -sO
https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-age
nt/wazuh-agent_<VERSI>_amd64.deb

sudo WAZUH_MANAGER="192.168.56.10" dpkg -i
wazuh-agent_<VERSI>_amd64.deb
```

Catatan:

- ganti <VERSI> sesuai dokumentasi resmi Wazuh;
- jangan memakai versi lama tanpa validasi;
- untuk lab, gunakan versi yang sama dengan server Wazuh;
- untuk produksi, ikuti kebijakan update organisasi.

Aktifkan agent:

```
sudo systemctl daemon-reload
sudo systemctl enable --now wazuh-agent
```

Cek status:

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

Jika Menggunakan Repository

Alur berbasis repository biasanya mencakup:

- menambahkan repository Wazuh;
- memasang package agent;
- menetapkan manager IP;
- menjalankan service.

Selalu cek dokumentasi Wazuh versi terbaru sebelum menjalankan perintah instalasi karena nama paket, repository, dan versi dapat berubah. (Dokumentasi Wazuh)

Catatan Instalasi Endpoint

Buat catatan di endpoint:

```
sudo mkdir -p /opt/soc-agent/docs
sudo nano /opt/soc-agent/docs/catatan-agent.txt
```

Isi:

```
# Catatan Wazuh Agent Endpoint

Hostname:
IP endpoint:
Role endpoint:
IP Wazuh Manager:
Tanggal instalasi:
Versi agent:
Status service:
Catatan error:
Solusi:
```

Instalasi tanpa catatan akan menyulitkan troubleshooting.

4.3 File Konfigurasi Agent

File konfigurasi utama agent Linux:

```
/var/ossec/etc/ossec.conf
```

Dokumentasi Wazuh menyebut `ossec.conf` sebagai file konfigurasi utama pada manager dan juga penting pada agent Linux. File ini berada di `/var/ossec/etc/ossec.conf`, dan Wazuh menyarankan backup sebelum perubahan karena konfigurasi salah dapat mencegah service berjalan. (Dokumentasi Wazuh)

Backup Sebelum Edit

Jalankan:

```
sudo cp -a /var/ossec/etc/ossec.conf
/var/ossec/etc/ossec.conf.backup.$(date +%F-%H%M)
```

Penjelasan:

- cp -a menyalin file lengkap beserta atribut;
- $(date +%F-%H%M) menambahkan tanggal dan jam;
- backup memudahkan rollback.

Membuka Konfigurasi

```
sudo nano /var/ossec/etc/ossec.conf
```

Cari bagian server:

```
sudo grep -n "server" /var/ossec/etc/ossec.conf
```

Contoh bagian penting:

```
<client>
  <server>
    <address>192.168.56.10</address>
    <port>1514</port>
    <protocol>tcp</protocol>
  </server>
</client>
```

Makna:

- <address> adalah IP Wazuh Manager;
- <port> adalah port komunikasi agent;
- <protocol> adalah protokol komunikasi.

Restart Agent

Setelah perubahan:

```
sudo systemctl restart wazuh-agent
```

Cek status:

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

Cek log:

```
sudo tail -f /var/ossec/logs/ossec.log
```

Validasi Manager IP

Gunakan:

```
sudo grep -n "<address>" /var/ossec/etc/ossec.conf
```

Jika IP salah, edit ulang.

Catatan Praktis

Jangan mengedit banyak bagian sekaligus.

Urutan aman:

1. backup;
2. edit satu bagian;
3. restart;
4. cek log;
5. cek status di server;
6. catat hasil.

4.4 Cek Status Agent

Setelah agent dipasang, jangan langsung percaya bahwa ia aktif.

Validasi dari endpoint dan server.

Cek dari Endpoint

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

Yang dicari:

- active (running);
- tidak ada error fatal;
- service aktif setelah reboot.

Pantau log service:

```
sudo journalctl -u wazuh-agent -f
```

Pantau log agent:

```
sudo tail -f /var/ossec/logs/ossec.log
```

Jika agent berhasil komunikasi, biasanya terlihat aktivitas koneksi ke manager.

Cek dari Server Wazuh

Di Wazuh server:

```
sudo /var/ossec/bin/agent_control -l
```

Cek agent aktif:

```
sudo /var/ossec/bin/agent_control -lc
```

Data penting:

- agent ID;
- hostname;
- IP address;
- status;
- last keep alive.

Tabel Catatan Agent

Buat file:

```
nano ~/soc-lab/docs/daftar-agent-linux.txt
```

Isi:

```
# Daftar Agent Linux
```

Agent ID	Hostname	IP	Role	Status	Last Keep Alive	Catatan
001	endpoint-01	192.168.56.20	endpoint admin	active	-	agent pertama

Interpretasi Status

Status	Makna	Tindakan
active	Agent terkoneksi	Lanjut cek log
disconnected	Agent tidak kontak manager	Cek network, service, IP
tidak muncul	Enrollment gagal	Cek instalasi dan manager IP
sering putus	Network / firewall / waktu	Cek koneksi dan log

Agent aktif bukan akhir pekerjaan. Agent aktif adalah awal monitoring.

4.5 Grouping Agent

Jika endpoint makin banyak, konfigurasi satu per satu akan melelahkan.

Wazuh mendukung **agent group** untuk konfigurasi terpusat. Dokumentasi Wazuh menjelaskan bahwa agent dapat dimasukkan ke group dan setiap group dapat memiliki konfigurasi unik melalui centralized configuration. (Dokumentasi Wazuh)

Materi group:

- agent group untuk server web;
- agent group untuk database;
- agent group untuk endpoint admin;
- konfigurasi per group.

Mengapa Grouping Penting

Tanpa group:

- konfigurasi tersebar;
- sulit audit;
- rawan salah;
- perubahan harus dilakukan manual;
- endpoint berbeda diperlakukan sama.

Dengan group:

- server web punya konfigurasi khusus;
- database punya log khusus;
- endpoint admin punya monitoring lebih ketat;
- rule dan log collection lebih rapi.

Contoh Desain Group

Group	Endpoint	Fokus Monitoring
web-server	server aplikasi web	log web, file web, service web
database-server	server database	log database, akses admin
admin-endpoint	mesin admin	SSH, sudo, file konfigurasi
default	endpoint umum	baseline monitoring

Membuat Group

Di Wazuh server:

```
sudo /var/ossec/bin/agent_groups -a -g web-server
sudo /var/ossec/bin/agent_groups -a -g database-server
sudo /var/ossec/bin/agent_groups -a -g admin-endpoint
```

Cek group:

```
sudo /var/ossec/bin/agent_groups -l
```

Menambahkan Agent ke Group

Contoh agent ID 001 masuk group admin-endpoint:

```
sudo /var/ossec/bin/agent_groups -a -i 001 -g
admin-endpoint
```

Cek:

```
sudo /var/ossec/bin/agent_groups -s -i 001
```

Konfigurasi Terpusat per Group

Lokasi umum konfigurasi terpusat:

```
/var/ossec/etc/shared/<nama_group>/agent.conf
```

Contoh:

```
sudo nano
/var/ossec/etc/shared/admin-endpoint/agent.conf
```

Contoh isi:

```
<agent_config>
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/auth.log</location>
  </localfile>

  <syscheck>
    <directories check_all="yes"
realtime="yes">/etc/ssh/sshd_config</directories>
    <directories check_all="yes"
realtime="yes">/etc/sudoers</directories>
  </syscheck>
</agent_config>
```

Restart manager:

```
sudo systemctl restart wazuh-manager
```

Restart agent di endpoint:

```
sudo systemctl restart wazuh-agent
```

Prinsip Grouping

- jangan masukkan semua endpoint ke group yang sama;
- jangan monitoring terlalu banyak file tanpa alasan;
- prioritaskan file dan log yang kritikal;
- dokumentasikan perubahan group;
- uji di satu endpoint sebelum diterapkan ke banyak endpoint.

Grouping membuat SOC lebih terukur.

4.6 Mengaktifkan Log Collection Endpoint

Wazuh Agent dapat mengumpulkan log dari file lokal endpoint melalui konfigurasi `localfile`. Dokumentasi Wazuh menjelaskan bahwa `localfile` digunakan untuk mengonfigurasi pengumpulan log dari file, event, maupun output command. (Dokumentasi Wazuh) Untuk Linux endpoint, Wazuh Agent dapat mengumpulkan log dari file yang ditentukan dalam `ossec.conf` dan meneruskannya ke Wazuh server untuk dianalisis. (Dokumentasi Wazuh)

Contoh log:

- `/var/log/auth.log`
- `/var/log/syslog`
- `/var/log/kern.log`
- `/var/log/nginx/access.log`
- `/var/log/nginx/error.log`

Cek Log Lokal di Endpoint

```
sudo tail -f /var/log/auth.log
sudo tail -f /var/log/syslog
sudo journalctl -xe
```

Makna:

- auth.log berisi aktivitas autentikasi;
- syslog berisi log sistem umum;
- journalctl -xe melihat event systemd terbaru dengan detail.

Menambahkan Log Collection

Edit agent:

```
sudo nano /var/ossec/etc/ossec.conf
```

Tambahkan di dalam <ossec_config>:

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/auth.log</location>
</localfile>

<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/syslog</location>
</localfile>

<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/kern.log</location>
</localfile>
```

Jika endpoint menjalankan server web berbasis Nginx:

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/nginx/access.log</location>
</localfile>

<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/nginx/error.log</location>
</localfile>
```

Restart agent:

```
sudo systemctl restart wazuh-agent
```

Cek log agent:

```
sudo tail -f /var/ossec/logs/ossec.log
```

Uji Log Autentikasi

Di endpoint:

```
sudo -k
sudo ls /root
```

Masukkan password salah satu atau dua kali di lab.
Di server:

```
sudo tail -f /var/ossec/logs/alerts/alerts.json | jq .
```

Cari alert autentikasi:

```
sudo jq 'select(.rule.description |
test("authentication|password|sudo|login"; "i"))' \
/var/ossec/logs/alerts/alerts.json
```

Simpan Evidence

```
mkdir -p ~/soc-lab/evidence/agent-log
sudo jq 'select(.rule.description |
test("authentication|password|sudo|login"; "i"))' \
/var/ossec/logs/alerts/alerts.json \
| tee ~/soc-lab/evidence/agent-log/auth-events.json
```

Prinsip Log Collection

- kumpulkan log yang relevan;
- jangan kumpulkan semuanya tanpa strategi;
- pahami format log;
- uji satu sumber log dulu;
- dokumentasikan file yang dipantau.

Log collection yang baik bukan yang paling banyak, tetapi yang paling berguna untuk investigasi.

4.7 File Integrity Monitoring

FIM, atau *File Integrity Monitoring*, memantau perubahan file penting.

Dokumentasi Wazuh menjelaskan bahwa FIM dikonfigurasi melalui blok `<syscheck>` dan dapat memantau direktori atau file pada endpoint Linux. (Dokumentasi Wazuh)

Pantau file penting:

- `/etc/passwd`
- `/etc/shadow`
- `/etc/sudoers`
- `/etc/ssh/sshd_config`
- `/var/www/html`

Mengapa File Ini Penting?

File / Direktori	Risiko Jika Berubah
`/etc/passwd`	User baru, perubahan akun
`/etc/shadow`	Perubahan password hash
`/etc/sudoers`	Eskalasi privilege
`/etc/ssh/sshd_config`	SSH dibuat lebih lemah
`/var/www/html`	Web defacement / web shell

Backup Konfigurasi Agent

```
sudo cp -a /var/ossec/etc/ossec.conf
/var/ossec/etc/ossec.conf.before-fim.$(date +%F-%H%M)
```

Konfigurasi FIM Lokal

Edit:

```
sudo nano /var/ossec/etc/ossec.conf
```

Cari blok:

```
<syscheck>
```

Tambahkan:

```
<directories check_all="yes"
realtime="yes">/etc/passwd</directories>
<directories check_all="yes"
realtime="yes">/etc/shadow</directories>
<directories check_all="yes"
realtime="yes">/etc/sudoers</directories>
<directories check_all="yes"
realtime="yes">/etc/ssh/sshd_config</directories>
<directories check_all="yes"
realtime="yes">/var/www/html</directories>
```

Restart agent:

```
sudo systemctl restart wazuh-agent
```

Cek:

```
sudo tail -f /var/ossec/logs/ossec.log
```

Catatan Kritis FIM

Jangan memantau seluruh filesystem secara membabi buta.

Dampaknya:

- terlalu banyak alert;
- CPU meningkat;
- storage membengkak;
- analis lelah;
- alert penting tertutup noise.

Pilih file yang benar-benar kritikal.

FIM harus tajam, bukan ramai.

4.8 Simulasi Perubahan File

Simulasi harus dilakukan di lab sendiri.
Jangan mengubah file penting sembarangan tanpa rencana rollback.

Untuk uji aman, gunakan file dummy:

```
sudo touch /etc/test-soc.conf
```

Edit file:

```
sudo nano /etc/test-soc.conf
```

Isi:

```
uji perubahan file untuk FIM SOC
```

Hapus file:

```
sudo rm /etc/test-soc.conf
```

Cek Alert di Server

Di Wazuh server:

```
sudo tail -f /var/ossec/logs/alerts/alerts.json | jq .
```

Cari alert FIM:

```
sudo jq 'select(.rule.groups[]? == "syscheck")'
/var/ossec/logs/alerts/alerts.json
```

Atau cari berdasarkan nama file:

```
sudo jq 'select(.syscheck.path? |
test("test-soc.conf"))' \
/var/ossec/logs/alerts/alerts.json
```

Simpan Evidence FIM

```
mkdir -p ~/soc-lab/evidence/fim
sudo jq 'select(.syscheck.path? | test("test-soc.conf"))' \
/var/ossec/logs/alerts/alerts.json \
| tee ~/soc-lab/evidence/fim/test-soc-fim-alert.json
```

Membuat Catatan FIM

```
nano ~/soc-lab/evidence/fim/catatan-fim.txt
```

Isi:

```
# Catatan Uji FIM

Tanggal:
Endpoint:
IP endpoint:
File diuji:
/etc/test-soc.conf

Aksi:
1. touch file
2. edit file
3. hapus file

Alert muncul:
[ ] Ya
[ ] Tidak

Rule:
Level:
Waktu alert:
Path:
Action:
Hash sebelum:
Hash sesudah:

Kesimpulan:
```

Uji File Penting dengan Aman

Untuk file penting, jangan langsung mengubah isi.
Gunakan pendekatan aman:

```
sudo cp -a /etc/ssh/sshd_config /tmp/sshd_config.test
```

Lalu pantau file dummy di lokasi lab terlebih dahulu.
Jika ingin menguji file asli, pastikan:

- backup ada;
- akses konsol tersedia;
- tidak dilakukan di produksi;
- perubahan bisa dikembalikan.

SOC yang baik tidak merusak sistem demi membuat alert.

4.9 Troubleshooting Agent

Troubleshooting agent harus sistematis.

Masalah umum:

- agent disconnected;
- salah IP manager;
- firewall blocking;
- waktu server tidak sinkron;
- service mati;
- DNS gagal.

Alur Troubleshooting Cepat

```
status service
→ cek log agent
→ cek IP manager
→ cek koneksi jaringan
→ cek port 1514/1515
→ cek waktu
→ cek firewall
→ cek status di server
```

1. Agent Disconnected

Di endpoint:

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

Jika mati:

```
sudo systemctl restart wazuh-agent
```

Cek log:

```
sudo tail -f /var/ossec/logs/ossec.log
```

Di server:

```
sudo /var/ossec/bin/agent_control -lc
```

2. Salah IP Manager

Di endpoint:

```
sudo grep -n "<address>" /var/ossec/etc/ossec.conf
```

Jika salah, edit:

```
sudo nano /var/ossec/etc/ossec.conf
```

Restart:

```
sudo systemctl restart wazuh-agent
```

3. Firewall Blocking

Tes koneksi:

```
ping <IP_WAZUH_SERVER>
```

Contoh:

```
ping 192.168.56.10
```

Tes port komunikasi:

```
nc -vz <IP_WAZUH_SERVER> 1514
```

Tes port enrollment:

```
nc -vz <IP_WAZUH_SERVER> 1515
```

Contoh:

```
nc -vz 192.168.56.10 1514
nc -vz 192.168.56.10 1515
```

Jika nc belum ada:

```
sudo apt install -y netcat-openbsd
```

Di Wazuh server, cek firewall:

```
sudo ufw status verbose
```

Jika perlu buka port untuk lab:

```
sudo ufw allow from 192.168.56.0/24 to any port 1514
sudo ufw allow from 192.168.56.0/24 to any port 1515
```

Batasi berdasarkan jaringan lab. Jangan buka ke semua jaringan tanpa alasan.

4. Waktu Tidak Sinkron

Cek endpoint:

```
timedatectl
```

Cek server:

```
timedatectl
```

Pastikan timezone sama:

```
Asia/Jakarta
```

Jika berbeda:

```
sudo timedatectl set-timezone Asia/Jakarta
```

Waktu salah membuat timeline incident kacau.

5. Service Mati

Restart agent:

```
sudo systemctl restart wazuh-agent
```

Cek log service:

```
sudo journalctl -u wazuh-agent -n 100 --no-pager
```

Cek log internal:

```
sudo tail -n 100 /var/ossec/logs/ossec.log
```

Cari error:

```
sudo grep -iE "error|warn|failed|critical|fatal"
/var/ossec/logs/ossec.log
```

6. DNS Gagal

Cek resolusi:

```
resolvectl status
```

Tes domain:

```
ping -c 4 ubuntu.com
```

Jika agent memakai IP manager, DNS tidak kritikal untuk komunikasi ke manager. Namun DNS tetap penting untuk update package dan repository.

7. Agent Tidak Muncul di Server

Di server:

```
sudo /var/ossec/bin/agent_control -l
```

Cek log manager:

```
sudo tail -f /var/ossec/logs/ossec.log
```

Cek port aktif:

```
sudo ss -tulpn | grep -E '1514|1515'
```

Jika port tidak aktif, cek service manager:

```
sudo systemctl status wazuh-manager
```

4.10 Hands-on Python: Audit Status Agent dari File JSON

Python dapat membantu membuat laporan sederhana dari output agent atau API. Misal kita punya file:

```
~/soc-lab/evidence/api/wazuh-agents.json
```

Buat script:

```
nano ~/soc-lab/scripts/ringkas_agent.py
```

Isi:

```
from pathlib import Path
import json

file_path = Path.home() / "soc-lab" / "evidence" / "api" /
"wazuh-agents.json"

if not file_path.exists():
    raise FileNotFoundError(f"File tidak ditemukan: {file_path}")

data = json.loads(file_path.read_text(encoding="utf-8"))

agents = data.get("data", {}).get("affected_items", [])

print("Ringkasan Agent Wazuh")
print("=" * 30)

for agent in agents:
    agent_id = agent.get("id", "-")
    name = agent.get("name", "-")
    ip = agent.get("ip", "-")
    status = agent.get("status", "-")
    last_keep_alive = agent.get("lastKeepAlive", "-")

    print(f"ID      : {agent_id}")
    print(f"Nama    : {name}")
    print(f"IP      : {ip}")
    print(f"Status  : {status}")
    print(f"KeepAlive: {last_keep_alive}")
    print("-" * 30)
```

Jalankan:

```
python3 ~/soc-lab/scripts/ringkas_agent.py
```

Penjelasan Singkat

- `Path` dipakai untuk membuat path file yang rapi.
- `json.loads()` membaca JSON menjadi struktur Python.
- `get()` dipakai agar script tidak langsung error jika field tidak ada.
- script mencetak agent ID, nama, IP, status, dan last keep alive.

Python dipakai untuk mempercepat dokumentasi, bukan menggantikan analisis.

4.11 Checklist Validasi Akhir Bab 4

Agent

- Wazuh Agent terpasang di endpoint.
- Service `wazuh-agent` aktif.
- Agent mengarah ke IP manager yang benar.
- Agent muncul di Wazuh server.
- Agent status `active`.
- Agent ID dicatat.
- Hostname dan IP dicatat.

Log Collection

- `/var/log/auth.log` terbaca.
- `/var/log/syslog` terbaca.
- `journalctl -xe` dapat digunakan.
- Log collection dikonfigurasi.
- Alert autentikasi dapat muncul.
- Evidence log disimpan.

Grouping

- Group agent dirancang.
- Minimal satu group dibuat.
- Agent masuk group yang benar.
- Konfigurasi group terdokumentasi.

FIM

- File penting ditentukan.
- FIM dikonfigurasi.
- File dummy diuji.
- Alert FIM muncul.
- Evidence FIM disimpan.

Troubleshooting

- Bisa cek status service.
- Bisa baca log agent.
- Bisa cek IP manager.
- Bisa tes port 1514.
- Bisa tes port 1515.
- Bisa cek waktu.
- Bisa restart agent.

- Bisa cek status agent dari server.

4.12 Kesalahan Umum Bab 4

1. Agent Terpasang tetapi IP Manager Salah

Gejala:

- agent aktif di endpoint;
- tetapi tidak muncul di server.

Solusi:

```
sudo grep -n "<address>" /var/ossec/etc/ossec.conf
```

Perbaiki IP manager.

2. Firewall Server Menolak Agent

Gejala:

- ping berhasil;
- port 1514/1515 gagal.

Cek:

```
nc -vz <IP_WAZUH_SERVER> 1514
nc -vz <IP_WAZUH_SERVER> 1515
```

3. Terlalu Banyak File Dipantau FIM

Gejala:

- alert banjir;
- CPU naik;
- analis lelah.

Solusi:

- pantau file kritikal saja;
- mulai dari `/etc/passwd`, `/etc/sudoers`, `/etc/ssh/sshd_config`;
- hindari memantau seluruh filesystem.

4. Semua Endpoint Masuk Group Default

Gejala:

- konfigurasi tidak spesifik;
- server web, database, dan endpoint admin diperlakukan sama.

Solusi:

- buat group sesuai role;
- gunakan konfigurasi terpusat;
- uji per group.

5. Tidak Menyimpan Evidence

Alert tanpa evidence sulit dipakai untuk laporan.

Minimal simpan:

- alert JSON;
- log mentah;
- waktu kejadian;
- hostname;
- IP;
- aksi yang dilakukan;
- hasil.

4.13 Rangkuman Bab

Bab ini menjadikan endpoint Linux sebagai sumber telemetry utama SOC.

Pembaca belajar bahwa agent bukan sekadar paket yang dipasang. Agent adalah komponen kritikal untuk melihat aktivitas endpoint, mengirim log, memantau perubahan file, dan memberi bukti teknis kepada analis.

Inti Bab 4:

- **agent harus aktif dan terhubung**;
- **IP manager harus benar**;
- **log collection harus terarah**;
- **FIM harus fokus pada file kritikal**;
- **grouping membuat konfigurasi lebih rapi**;
- **troubleshooting harus sistematis**;
- **evidence harus disimpan**.

SOC yang kuat tidak hanya punya server pusat.

SOC yang kuat punya endpoint yang termonitor dengan benar.

Output Bab

Setelah menyelesaikan Bab 4, pembaca memiliki:

- Wazuh Agent aktif di endpoint Ubuntu 26.04;
- endpoint terhubung ke Wazuh Manager;
- agent dapat diperiksa dari endpoint dan server;
- group agent awal terbentuk;
- log endpoint dikumpulkan;
- FIM berjalan untuk file penting;
- simulasi perubahan file menghasilkan alert;
- evidence alert tersimpan;
- troubleshooting dasar agent dikuasai;
- script Python sederhana untuk membaca ringkasan agent.

Kalimat kunci Bab 4:

> **SOC yang matang bukan hanya memasang agent, tetapi memastikan setiap endpoint mengirim log yang benar, file kritikal terpantau, status agent jelas, dan setiap alert dapat dibuktikan dari CLI.**

Bab 5 — Sensor Jaringan: Security Onion, Suricata, Zeek

Bab 5 yang berfokus pada Security Onion, Suricata, Zeek, konsep sensor jaringan, simulasi port scan di lab tertutup, pembacaan alert, dan pengambilan PCAP sebagai evidence. Bab ini juga memakai arah dari dokumen **Open Source SOC Mastery** yang menekankan bahwa network monitoring harus menghubungkan alert Suricata, metadata Zeek, PCAP, endpoint log, dan catatan investigasi; bukan berhenti pada satu alert.

Security Onion resmi diposisikan sebagai platform terbuka untuk *network visibility*, *host visibility*, *intrusion detection*, *log management*, dan *case management*. Untuk visibilitas jaringan, Security Onion memakai deteksi berbasis signature melalui Suricata, metadata protokol melalui Zeek atau Suricata, serta full packet capture melalui komponen PCAP. (Dokumentasi Security Onion) Suricata menghasilkan output EVE JSON yang dapat memuat alert, anomaly, metadata, file info, dan record protokol. (Suricata User Guide) Zeek menghasilkan log penting seperti `conn.log`, `dns.log`, `http.log`, `ssl.log`, `files.log`, dan `notice.log` untuk membaca konteks trafik jaringan. (Dokumentasi Zeek)

5.1 Mengapa Agent Saja Tidak Cukup

Agent melihat endpoint.

Sensor melihat jaringan.

Keduanya saling melengkapi.

Agent Wazuh sangat penting untuk membaca:

- login gagal;
- perubahan file;
- aktivitas user;
- konfigurasi sistem;
- paket rentan;
- event endpoint;
- FIM;
- service mati.

Namun agent tidak selalu melihat seluruh percakapan jaringan.

Contoh yang lebih mudah terlihat dari sensor:

- port scanning;
- koneksi antar host;
- DNS query aneh;
- trafik HTTP mencurigakan;
- percobaan eksploitasi jaringan;
- pola koneksi berulang;
- komunikasi ke IP luar;
- trafik lateral antar mesin lab.

Endpoint log menjawab: apa yang terjadi di mesin.
Network sensor menjawab: bagaimana aktivitas bergerak di jaringan.

Contoh Perbedaan Pandangan

Kejadian	Agent Endpoint	Sensor Jaringan
SSH gagal	Melihat log autentikasi	Melihat koneksi ke port 22
Port scan	Bisa tidak terlihat jelas	Sangat terlihat
DNS mencurigakan	Tergantung log endpoint	Terlihat di dns.log
Web request aneh	Perlu log web	Terlihat di HTTP metadata
Lateral movement	Terlihat sebagian	Terlihat antar host
File berubah	Sangat terlihat dengan FIM	Tidak terlihat isi file
Malware komunikasi keluar	Bisa terlihat dari proses	Terlihat dari koneksi

SOC yang matang tidak memilih salah satu.

SOC yang matang memakai keduanya.

Prinsip Korelasi

Alert endpoint dan alert jaringan harus dikorelasikan.
Contoh:

Wazuh:
endpoint-01 mengalami login gagal berulang

Suricata:
IP yang sama melakukan scan ke port 22

Zeek:
conn.log menunjukkan banyak koneksi pendek ke port 22

Kesimpulan:
indikasi brute force / reconnaissance terhadap SSH

Satu log memberi sinyal. Banyak log memberi cerita.

5.2 Konsep Network Sensor

Network sensor adalah mesin atau perangkat yang ditempatkan untuk membaca trafik jaringan.

Sensor tidak harus menjadi gateway.

Dalam banyak desain SOC, sensor hanya “mendengar” trafik melalui mirror port, TAP, atau konfigurasi virtual lab.

Komponen utama:

- **management interface**
- **monitoring interface**
- **TAP / SPAN / mirror port**
- **packet capture**
- **alert**
- **flow**
- **metadata**

Management Interface

Management interface dipakai untuk mengelola sensor.

Fungsinya:

- akses SSH admin;
- update sistem;
- akses dashboard;
- sinkronisasi waktu;
- komunikasi ke server manajemen;
- pengiriman log.

Interface ini biasanya punya IP address.

Contoh:

sensor-01 management interface:
IP 192.168.56.40

Monitoring Interface

Monitoring interface dipakai untuk mendengar trafik.

Interface ini sering tidak diberi IP.

Ia hanya menerima salinan paket.

Dokumen Open Source SOC Mastery menegaskan konsep penting: management interface digunakan untuk mengelola sensor, sedangkan monitor interface digunakan untuk membaca trafik; kesalahan umum adalah sensor terpasang tetapi monitor interface tidak melihat trafik.

Contoh:

sensor-01 monitor interface:
no IP
promiscuous mode
membaca trafik mirror

TAP / SPAN / Mirror Port

Sensor harus mendapat salinan trafik.

Cara umum:

- TAP fisik;
- SPAN port;
- mirror port switch;
- virtual bridge;
- virtual switch;
- gateway sensor;
- lab internal network.

Dalam lab VM, gunakan pendekatan sederhana:

```
endpoint-01 → jaringan lab → target-lab
                        ↓
                sensor-01 membaca trafik
```

Packet Capture

Packet capture adalah proses merekam paket jaringan.

Tool utama:

`tcpdump`

Output biasanya disimpan sebagai file .pcap.
PCAP berguna sebagai evidence karena berisi paket mentah.
Namun PCAP juga berat dan sensitif.
Jangan menyimpan PCAP sembarangan jika berisi data pribadi, kredensial, atau trafik produksi.

Alert

Alert adalah sinyal deteksi.

Dalam bab ini, alert utama berasal dari Suricata.

Contoh field alert:

- timestamp;
- source IP;
- destination IP;
- source port;
- destination port;
- protocol;
- signature;
- category;
- severity;
- flow ID.

Suricata EVE JSON memiliki `flow_id` yang berguna untuk mengkorelasikan alert dengan data flow, protokol, file info, dan anomali dalam satu sesi atau flow. (Suricata User Guide)

Flow

Flow adalah ringkasan koneksi.

Biasanya dibaca dari Zeek `conn.log`.

Pertanyaan flow:

- koneksi dari IP mana ke IP mana?
- port tujuan apa?
- protokol apa?
- berapa lama?
- berapa byte?
- koneksi berhasil atau gagal?
- berulang atau tunggal?

Zeek `conn.log` adalah salah satu log terpenting dan mencatat koneksi untuk TCP maupun UDP. (Dokumentasi Zeek)

Metadata

Metadata adalah ringkasan teknis tanpa harus membaca seluruh paket.

Contoh metadata:

- DNS query;
- HTTP host;
- URI;
- TLS SNI;
- file hash;
- connection duration;
- user agent;
- status code.

Metadata membuat investigasi lebih cepat.

Alert memberi alarm. Flow memberi pola. Metadata memberi konteks. PCAP memberi bukti detail.

5.3 Security Onion sebagai Platform Sensor

Security Onion adalah platform SOC open source yang sangat cocok untuk network security monitoring.

Materi utama:

- peran Security Onion;
- Suricata;
- Zeek;
- Hunt;
- Alerts;
- Dashboards;
- PCAP;
- sensor deployment.

Peran Security Onion

Security Onion menyatukan banyak fungsi:

- sensor jaringan;
- alert IDS;
- metadata jaringan;
- pencarian event;
- dashboard;
- hunting;
- case management;
- PCAP;
- log management.

Untuk lab dengan resource cukup, Security Onion menjadi jalan praktis karena Suricata, Zeek, dashboard, dan PCAP sudah berada dalam satu ekosistem.

Namun jika resource terbatas, dokumen Open Source SOC Mastery menyarankan menjalankan Suricata/Zeek tanpa Security Onion; lebih baik sedikit tools tetapi stabil daripada banyak tools tetapi lambat dan tidak bisa dianalisis.

Suricata di Security Onion

Suricata berperan sebagai mesin deteksi berbasis rule.

Fokus:

- signature;
- alert;
- category;
- severity;
- EVE JSON;
- deteksi port scan;
- deteksi pola serangan.

Suricata cocok untuk menjawab:

> “Apakah trafik ini cocok dengan pola serangan tertentu?”

Zeek di Security Onion

Zeek berperan sebagai penghasil metadata.

Fokus:

- koneksi;
- DNS;
- HTTP;
- TLS;
- file;
- notice;
- protokol.

Zeek cocok untuk menjawab:

> “Apa yang sebenarnya terjadi di percakapan jaringan ini?”

Hunt

Hunt dipakai analis untuk mencari event secara aktif.

Contoh pertanyaan hunt:

- IP mana yang paling sering melakukan koneksi?
- Host mana yang melakukan DNS query aneh?
- Apakah ada koneksi ke port tidak biasa?
- Apakah ada banyak koneksi pendek ke banyak port?
- Apakah ada HTTP request mencurigakan?

Alerts

Alerts menampilkan sinyal deteksi.

Namun alert tidak boleh langsung dianggap incident.

Pertanyaan triage:

- signature apa?
- severity berapa?
- IP sumber siapa?
- IP tujuan siapa?
- port apa?
- apakah host milik kita?
- ada log Zeek terkait?
- ada alert endpoint terkait?
- ada PCAP?

Dashboards

Dashboard membantu membaca tren.

Panel berguna:

- top source IP;
- top destination IP;
- top alert;
- alert by severity;
- alert timeline;
- top DNS query;
- HTTP host;
- traffic volume.

Dashboard bukan pengganti CLI.

CLI tetap penting untuk pembuktian.

PCAP

PCAP dipakai saat metadata belum cukup.

Gunakan PCAP jika:

- alert penting;
- perlu bukti teknis;
- perlu melihat payload;
- perlu rekonstruksi komunikasi;
- perlu pelaporan detail.

Hati-hati:

- PCAP besar;
- PCAP sensitif;
- PCAP bisa berisi data rahasia;
- simpan dengan kontrol akses.

Sensor Deployment

Desain minimum:

```
sensor-01
├── management interface: punya IP
└── monitoring interface: membaca trafik, tanpa IP
```

Cek interface:

```
ip a
```

Cek apakah interface melihat trafik:

```
sudo tcpdump -i <monitor_interface> -nn
```

Contoh:

```
sudo tcpdump -i ens19 -nn
```

Jika tidak ada trafik, masalahnya mungkin bukan Suricata atau Zeek.
Masalahnya bisa arsitektur jaringan.
Sensor yang tidak melihat trafik tidak akan menghasilkan alert bermakna.

5.4 Suricata untuk Deteksi Rule-Based

Suricata adalah mesin deteksi jaringan berbasis rule.

File penting:

- rule;
- `eve.json`;
- `fast.log`;
- `suricata.yaml`.

File Penting Suricata

File	Fungsi
rule	Pola deteksi
eve.json	Output JSON utama
fast.log	Ringkasan alert
suricata.yaml	Konfigurasi Suricata

Suricata EVE output menghasilkan alert, anomaly, metadata, file info, dan protocol-specific records dalam JSON. (Suricata User Guide)

Lokasi Umum File

Pada instalasi mandiri, lokasi umum:

```
/etc/suricata/suricata.yaml
/var/log/suricata/eve.json
/var/log/suricata/fast.log
/var/lib/suricata/rules/
```

Pada Security Onion, lokasi bisa berbeda karena platform mengelola service dan pipeline sendiri.

Gunakan dokumentasi Security Onion untuk lokasi spesifik instalasi.

Cek Service Suricata

Jika Suricata standalone:

```
sudo systemctl status suricata
```

Cek interface:

```
ip a
```

Cek konfigurasi:

```
sudo suricata -T -c /etc/suricata/suricata.yaml -v
```

Penjelasan:

- -T test konfigurasi;
- -c menunjuk file konfigurasi;
- -v menampilkan output lebih detail.

Membaca EVE JSON

```
sudo tail -f /var/log/suricata/eve.json
```

Baca lebih rapi:

```
sudo jq . /var/log/suricata/eve.json | less
```

Filter hanya alert:

```
sudo jq 'select(.event_type=="alert")'
/var/log/suricata/eve.json
```

Ambil field penting:

```
sudo jq 'select(.event_type=="alert") | {
  time: .timestamp,
  src: .src_ip,
  sport: .src_port,
  dst: .dest_ip,
  dport: .dest_port,
  proto: .proto,
  signature: .alert.signature,
  category: .alert.category,
  severity: .alert.severity,
  flow_id: .flow_id
}' /var/log/suricata/eve.json
```

Membaca fast.log

```
sudo tail -f /var/log/suricata/fast.log
```

fast.log lebih ringkas.
Cocok untuk melihat alert cepat.
Namun untuk otomasi, gunakan eve.json.

Membaca suricata.yaml

```
sudo less /etc/suricata/suricata.yaml
```

Cari interface:

```
sudo grep -n "interface" /etc/suricata/suricata.yaml
```

Cari EVE:

```
sudo grep -n "eve-log" /etc/suricata/suricata.yaml
```

Catatan Penting Rule

Rule yang terlalu banyak tanpa tuning akan menghasilkan noise.

Mulai dari:

- rule bawaan;
- rule yang relevan dengan lab;
- alert port scan;
- alert SSH;
- alert HTTP;
- alert DNS.

Rule yang tajam lebih penting daripada rule yang banyak.

5.5 Zeek untuk Metadata Jaringan

Zeek bukan sekadar IDS berbasis signature.

Zeek menghasilkan metadata jaringan yang sangat berguna untuk investigasi.

Log penting:

- `conn.log`
- `dns.log`
- `http.log`
- `ssl.log`
- `files.log`
- `notice.log`

Daftar log Zeek resmi mencakup `conn.log`, `dns.log`, `http.log`, `files.log`, `ssl.log`, `ssh.log`, dan banyak log protokol lain. (Dokumentasi Zeek)

Lokasi Log Zeek

Pada instalasi umum:

```
ls -lah /opt/zeek/logs/current/
```

Baca koneksi:

```
cat /opt/zeek/logs/current/conn.log | less
```

Baca DNS:

```
cat /opt/zeek/logs/current/dns.log | less
```

Pada distribusi atau platform tertentu, lokasi log bisa berbeda.
Cek dengan:

```
sudo find / -name "conn.log" 2>/dev/null | head
```

conn.log

`conn.log` menjawab:

- IP sumber;
- IP tujuan;
- port tujuan;
- protokol;
- service;
- durasi;
- byte;
- status koneksi.

Baca field penting:

```
cat /opt/zeek/logs/current/conn.log | zeek-cut id.orig_h
id.resp_h id.resp_p proto service duration orig_bytes
resp_bytes
```

Jika `zeek-cut` tidak tersedia, baca dengan `less`.

dns.log

`dns.log` sangat penting untuk melihat nama domain yang diakses. Dokumentasi Zeek menyebut `dns.log` sebagai salah satu sumber data paling penting untuk admin keamanan dan jaringan. (Dokumentasi Zeek)

Baca:

```
cat /opt/zeek/logs/current/dns.log | zeek-cut id.orig_h
query answers
```

Pertanyaan analisis:

- host mana melakukan query?
- domain apa yang ditanya?
- jawaban IP apa?
- query berulang?
- domain tampak acak?
- ada koneksi setelah DNS query?

http.log

Baca:

```
cat /opt/zeek/logs/current/http.log | zeek-cut id.orig_h
id.resp_h host uri method status_code user_agent
```

Pertanyaan analisis:

- host apa yang diakses?
- URI apa?
- metode apa?
- status code apa?
- user agent apa?
- ada pola scanner?

ssl.log

Baca:

```
cat /opt/zeek/logs/current/ssl.log | zeek-cut id.orig_h
id.resp_h server_name version cipher
```

Pertanyaan analisis:

- server name apa?
- TLS version apa?
- koneksi ke IP mana?
- domain cocok dengan DNS?
- ada domain mencurigakan?

files.log

Baca:

```
cat /opt/zeek/logs/current/files.log | zeek-cut fuid
tx_hosts rx_hosts mime_type filename
```

Pertanyaan analisis:

- ada file lewat jaringan?
- tipe file apa?
- siapa pengirim?
- siapa penerima?
- apakah perlu hash?

notice.log

Baca:

```
cat /opt/zeek/logs/current/notice.log | zeek-cut ts uid
id.orig_h id.resp_h note msg
```

`notice.log` berisi event yang dianggap penting oleh Zeek.

Prinsip Membaca Zeek

Jangan hanya melihat satu log.

Lakukan pivot:

```
conn.log
→ dns.log
→ http.log
→ ssl.log
→ files.log
→ notice.log
→ PCAP
```

Zeek membuat trafik jaringan bisa dibaca seperti catatan percakapan.

5.6 Simulasi Port Scan di Lab Tertutup

Simulasi hanya dilakukan di lab milik sendiri.

Jangan memindai sistem publik.

Jangan memindai jaringan yang tidak punya izin.

Tujuan praktik:

- menghasilkan trafik terkontrol;
- melihat alert Suricata;
- melihat koneksi Zeek;
- membuat catatan investigasi;
- menyimpan evidence.

Topologi Lab

```
scanner-lab / endpoint-01
        |
        | trafik scan
        v
target-lab
        ^
        |
sensor-01 membaca trafik mirror
```

Contoh IP:

Mesin	IP
endpoint-01	192.168.56.20
target-lab	192.168.56.30
sensor-01	192.168.56.40

Instal Nmap di Mesin Lab

Di mesin scanner lab:

```
sudo apt update
sudo apt install -y nmap
```

Jalankan Scan Ringan

```
nmap -sS <IP_TARGET_LAB>
```

Contoh:

```
nmap -sS 192.168.56.30
```

Catatan:

- -sS adalah TCP SYN scan;
- lakukan hanya ke target lab milik sendiri;
- jangan memakai opsi agresif pada sistem nyata;
- cukup satu target untuk praktik.

Cek Suricata

Di sensor:

```
sudo tail -f /var/log/suricata/eve.json | jq .
```

Filter alert:

```
sudo jq 'select(.event_type=="alert")'
/var/log/suricata/eve.json
```

Cari IP target:

```
sudo jq 'select(.src_ip=="192.168.56.20" or
.dest_ip=="192.168.56.30")' \
/var/log/suricata/eve.json
```

Cek Zeek

```
cat /opt/zeek/logs/current/conn.log | zeek-cut id.orig_h
id.resp_h id.resp_p proto service
```

Cari IP scanner:

```
grep "192.168.56.20" /opt/zeek/logs/current/conn.log |
less
```

Cari IP target:

```
grep "192.168.56.30" /opt/zeek/logs/current/conn.log |
less
```

Catat Hasil

Buat folder evidence:

```
mkdir -p ~/soc-lab/evidence/network/portscan
```

Simpan alert Suricata:

```
sudo jq 'select(.event_type=="alert")'
/var/log/suricata/eve.json \
| tee ~/soc-lab/evidence/network/portscan/suricata-alerts.json
```

Simpan koneksi Zeek:

```
grep "192.168.56.30" /opt/zeek/logs/current/conn.log \
| tee ~/soc-lab/evidence/network/portscan/zeek-conn-target.log
```

Buat catatan:

```
nano
~/soc-lab/evidence/network/portscan/catatan-portscan.txt
```

Isi:

```
# Catatan Simulasi Port Scan

Tanggal:
Analis:
Scanner IP:
Target IP:
Sensor:
Interface monitor:

Command:
nmap -sS <IP_TARGET_LAB>

Bukti Suricata:
- File:
- Signature:
- Severity:
- Source IP:
- Destination IP:
- Destination port:

Bukti Zeek:
- conn.log:
- jumlah koneksi:
- port tujuan:
- service:

Kesimpulan:
Port scan terdeteksi di lab tertutup.
```

5.7 Membaca Alert Network

Alert network harus dibaca disiplin.

Pertanyaan analis:

- source IP siapa?
- destination IP siapa?
- port apa?
- signature apa?
- severity berapa?
- ada bukti flow?
- ada bukti DNS/HTTP?

Field Penting Suricata

Dokumen Open Source SOC Mastery menekankan field penting Suricata: timestamp, event_type, src_ip, src_port, dest_ip, dest_port, proto, alert.signature, alert.category, alert.severity, dan flow_id.

Gunakan:

```
sudo jq 'select(.event_type=="alert") | {
  timestamp: .timestamp,
  src_ip: .src_ip,
  src_port: .src_port,
  dest_ip: .dest_ip,
  dest_port: .dest_port,
  proto: .proto,
  signature: .alert.signature,
  category: .alert.category,
  severity: .alert.severity,
  flow_id: .flow_id
}' /var/log/suricata/eve.json
```

Interpretasi Cepat

Field	Pertanyaan
`timestamp`	Kapan terjadi?
`src_ip`	Siapa sumber?
`dest_ip`	Siapa target?
`dest_port`	Layanan apa?
`proto`	Protokol apa?
`signature`	Deteksi apa?
`category`	Kategori ancaman apa?
`severity`	Prioritas berapa?
`flow_id`	Bisa dikorelasikan ke flow apa?

Jangan Langsung Menyimpulkan

Alert port scan belum tentu incident.

Bisa saja:

- admin melakukan audit;
- vulnerability scanner internal;
- lab test;
- monitoring sah;
- false positive.

Triage wajib bertanya:

- apakah source IP dikenal?
- apakah aktivitas diizinkan?
- apakah terjadi di jam wajar?
- apakah target aset kritikal?
- apakah ada login sukses setelah scan?
- apakah ada alert endpoint?
- apakah ada koneksi lanjutan?

Pivot ke Zeek

Dari alert Suricata, ambil:

```
src_ip
dest_ip
dest_port
timestamp
```

Lalu cari di Zeek:

```
grep "<SRC_IP>" /opt/zeek/logs/current/conn.log | less
```

Contoh:

```
grep "192.168.56.20" /opt/zeek/logs/current/conn.log |
less
```

Jika ada DNS:

```
grep "192.168.56.20" /opt/zeek/logs/current/dns.log |
less
```

Jika ada HTTP:

```
grep "192.168.56.20" /opt/zeek/logs/current/http.log |
less
```

Pivot ke Endpoint

Di Wazuh server:

```
sudo jq 'select(.agent.ip=="192.168.56.30")'
/var/ossec/logs/alerts/alerts.json
```

Atau cari IP:

```
sudo grep "192.168.56.20"
/var/ossec/logs/alerts/alerts.json
```

Tujuan:

- cek apakah target juga punya alert endpoint;
- cek apakah login gagal muncul;
- cek apakah file berubah;
- cek apakah service mati.

Alert jaringan harus dikaitkan dengan bukti endpoint.

5.8 Mengambil PCAP untuk Evidence

PCAP adalah bukti paket mentah.

Gunakan saat diperlukan, bukan untuk semua trafik.

`tcpdump` mendukung opsi `-w` untuk menulis raw packet ke file dan opsi `-r` untuk membaca kembali file tersebut. (Linux Die)

Capture Trafik Target

```
sudo tcpdump -i <interface> -w evidence-portscan.pcap
host <IP_TARGET>
```

Contoh:

```
sudo tcpdump -i ens19 -w evidence-portscan.pcap host
192.168.56.30
```

Penjelasan:

- sudo diperlukan untuk akses interface;
- tcpdump menangkap paket;
- -i ens19 memilih interface monitor;
- -w evidence-portscan.pcap menyimpan paket ke file;
- host 192.168.56.30 memfilter trafik yang melibatkan target.

Baca PCAP

```
tcpdump -r evidence-portscan.pcap
```

Agar tidak melakukan DNS lookup:

```
tcpdump -nn -r evidence-portscan.pcap
```

Agar ringkas:

```
tcpdump -nn -r evidence-portscan.pcap | less
```

Capture dengan Batas Jumlah Paket

```
sudo tcpdump -i ens19 -c 200 -w evidence-portscan.pcap
host 192.168.56.30
```

Penjelasan:

- -c 200 berhenti setelah 200 paket;
- ini mencegah file terlalu besar.

Simpan ke Folder Evidence

```
mkdir -p ~/soc-lab/evidence/network/pcap
mv evidence-portscan.pcap ~/soc-lab/evidence/network/pcap/
```

Buat hash:

```
sha256sum ~/soc-lab/evidence/network/pcap/evidence-portscan.pcap
\
| tee ~/soc-lab/evidence/network/pcap/evidence-portscan.sha256
```

Catatan Chain of Custody Sederhana

```
nano ~/soc-lab/evidence/network/pcap/catatan-pcap.txt
```

Isi:

```
# Catatan Evidence PCAP

Nama file:
Hash SHA256:
Tanggal capture:
Analis:
Interface:
Filter:
Target IP:
Tujuan capture:
Durasi / jumlah paket:
Lokasi penyimpanan:
Catatan sensitivitas:
```

Prinsip PCAP

- capture hanya trafik yang diperlukan;
- gunakan filter;
- batasi durasi;
- simpan hash;
- jangan edit file asli;
- jangan membagikan PCAP berisi data sensitif;
- dokumentasikan konteks.

PCAP adalah bukti kuat, tetapi juga data sensitif. Perlakukan seperti evidence.

5.9 Hands-on Python: Ringkas Alert Suricata EVE JSON

Python membantu membuat ringkasan alert.

Buat script:

```
nano ~/soc-lab/scripts/ringkas_suricata_alert.py
```

Isi:

```
from pathlib import Path
import json
from collections import Counter

eve_file = Path("/var/log/suricata/eve.json")

if not eve_file.exists():
    raise FileNotFoundError(f"File tidak ditemukan: {eve_file}")

src_counter = Counter()
signature_counter = Counter()
severity_counter = Counter()

with eve_file.open("r", encoding="utf-8", errors="ignore") as f:
    for line in f:
        try:
            event = json.loads(line)
        except json.JSONDecodeError:
            continue

        if event.get("event_type") != "alert":
            continue

        src_ip = event.get("src_ip", "-")
        signature = event.get("alert", {}).get("signature", "-")
        severity = event.get("alert", {}).get("severity", "-")

        src_counter[src_ip] += 1
        signature_counter[signature] += 1
        severity_counter[str(severity)] += 1

print("Top Source IP")
print("=" * 30)
for ip, count in src_counter.most_common(10):
    print(f"{ip}: {count}")

print("\nTop Signature")
print("=" * 30)
for sig, count in signature_counter.most_common(10):
    print(f"{sig}: {count}")

print("\nSeverity")
print("=" * 30)
for sev, count in severity_counter.most_common():
    print(f"{sev}: {count}")
```

Jalankan:

```
sudo python3 ~/soc-lab/scripts/ringkas_suricata_alert.py
```

Penjelasan Singkat

- `Path` menunjuk file EVE JSON.
- `json.loads()` membaca setiap baris JSON.
- `Counter` menghitung kemunculan.
- script hanya memproses event dengan `event_type=="alert"`.
- output menampilkan top source IP, top signature, dan severity.

Simpan Output

```
mkdir -p ~/soc-lab/evidence/network/report
sudo python3 ~/soc-lab/scripts/ringkas_suricata_alert.py \
| tee ~/soc-lab/evidence/network/report/ringkasan-suricata.txt
```

Python mempercepat ringkasan. Analisis tetap dilakukan manusia.

5.10 Template Catatan Investigasi Network Alert

Buat template:

```
nano ~/soc-lab/docs/template-investigasi-network-alert.txt
```

Isi:

```
# Template Investigasi Network Alert

## 1. Identitas Alert
Tanggal:
Analis:
Sumber alert:
Sensor:
Interface:

## 2. Data Alert Suricata
Timestamp:
Source IP:
Source port:
Destination IP:
Destination port:
Protocol:
Signature:
Category:
Severity:
Flow ID:

## 3. Data Zeek
conn.log:
dns.log:
http.log:
ssl.log:
files.log:
notice.log:

## 4. Data Endpoint
Wazuh alert:
Hostname:
Agent ID:
User:
Log terkait:

## 5. PCAP
Nama file:
Hash SHA256:
Filter capture:
Catatan:
```

```
## 6. Analisis
Apakah source IP dikenal?
Apakah target aset lab?
Apakah aktivitas diizinkan?
Apakah ada koneksi berulang?
Apakah ada DNS/HTTP terkait?
Apakah ada alert endpoint?
Apakah ini false positive?

## 7. Kesimpulan
[ ] False positive
[ ] Aktivitas lab sah
[ ] Perlu monitoring lanjut
[ ] Incident

## 8. Rekomendasi
Tindakan:
Approval:
Catatan risiko:
Lesson learned:
```

Template ini akan dipakai kembali di Bab 9 untuk incident handling.

5.11 Checklist Validasi Akhir Bab 5

Sensor

- Sensor memiliki management interface.
- Sensor memiliki monitoring interface.
- Monitoring interface melihat trafik.
- IP management dicatat.
- Interface monitor dicatat.
- Waktu sensor sinkron.

Security Onion / Sensor Platform

- Sensor aktif.
- Suricata berjalan.
- Zeek berjalan.
- Dashboard dapat diakses dari jaringan management.
- PCAP tersedia jika diaktifkan.
- Log mulai masuk.

Suricata

- `eve.json` tersedia.
- `fast.log` tersedia.
- Alert dapat difilter dengan `jq`.
- Field penting dipahami.
- Alert port scan muncul di lab.

Zeek

- `conn.log` tersedia.
- `dns.log` tersedia.
- `http.log` dicek jika ada trafik web.
- `ssl.log` dicek jika ada TLS.
- Pivot dari alert ke Zeek berhasil.

PCAP

- Capture bisa dilakukan.
- File `.pcap` tersimpan.
- File bisa dibaca dengan `tcpdump -r`.
- Hash SHA256 dibuat.
- Catatan evidence dibuat.

Dokumentasi

- Catatan port scan dibuat.
- Template investigasi network alert tersedia.
- Evidence Suricata disimpan.
- Evidence Zeek disimpan.
- Evidence PCAP disimpan jika perlu.
- Ringkasan Python dibuat.

5.12 Kesalahan Umum Bab 5

1. Sensor Tidak Melihat Trafik

Gejala:

- dashboard kosong;
- `eve.json` sepi;
- Zeek log tidak bertambah;
- tidak ada alert saat scan.

Penyebab:

- salah interface;
- mirror port belum aktif;
- monitor interface diberi IP dan salah jalur;
- trafik tidak lewat sensor;
- mode VM salah.

Cek:

```
sudo tcpdump -i <monitor_interface> -nn
```

2. Hanya Membaca Suricata, Tidak Membaca Zeek

Suricata memberi alert.
Zeek memberi konteks.
Tanpa Zeek, analis mudah berhenti pada signature.

Alert tanpa konteks sering menyesatkan.

3. Tidak Menyimpan PCAP Penting

PCAP tidak perlu selalu disimpan.
Namun untuk alert penting, PCAP bisa menjadi evidence kuat.
Gunakan filter dan hash.

4. Tidak Menyamakan Waktu

Jika waktu sensor, endpoint, dan server berbeda, timeline rusak.
Cek:

```
timedatectl
```

Pastikan zona waktu konsisten.

5. Terlalu Banyak Rule Tanpa Tuning

Rule yang terlalu banyak dapat membuat alert banjir.
Mulai dari rule relevan.
Tuning berdasarkan aset dan lab.

6. Melakukan Scan ke Sistem Publik

Ini pelanggaran serius.
Semua simulasi harus dilakukan di lab sendiri.

Lab SOC bukan alasan untuk menguji sistem orang lain.

5.13 Rangkuman Bab

Bab ini memperluas SOC dari endpoint ke jaringan.
Wazuh Agent tetap penting untuk melihat aktivitas host. Tetapi Security Onion, Suricata, dan Zeek memberi visibilitas terhadap trafik. Suricata membantu mendeteksi pola serangan berbasis rule. Zeek membantu membaca metadata koneksi, DNS, HTTP, TLS, file, dan notice. PCAP memberi bukti paket mentah saat diperlukan.

Inti Bab 5 adalah korelasi.

Alur investigasi yang sehat:

```
Alert Suricata
→ field penting
→ Zeek conn.log
→ dns.log / http.log / ssl.log
→ PCAP jika perlu
→ Wazuh endpoint alert
→ catatan investigasi
→ kesimpulan
```

SOC yang matang tidak hanya bertanya:

> “Ada alert apa?”

SOC yang matang bertanya:

> “Apa buktinya, konteksnya apa, dan apakah endpoint juga melihat hal yang sama?”

Output Bab

Setelah menyelesaikan Bab 5, pembaca memiliki:

- pemahaman mengapa agent saja tidak cukup;
- pemahaman management interface dan monitoring interface;
- sensor jaringan aktif;
- Suricata menghasilkan `eve.json` dan `fast.log`;
- Zeek menghasilkan `conn.log`, `dns.log`, dan log metadata lain;
- simulasi port scan berjalan di lab tertutup;
- alert Suricata dapat dibaca dari CLI;
- log Zeek dapat dipakai untuk pivot;
- PCAP dapat diambil sebagai evidence;
- hash evidence dibuat;
- template investigasi network alert tersedia;
- script Python sederhana untuk ringkasan alert Suricata.

Kalimat kunci Bab 5:

> **Serangan tidak hanya meninggalkan jejak di endpoint; ia juga meninggalkan percakapan di jaringan. SOC yang matang harus mampu membaca keduanya.**

Bab 6 — Akses Log via CLI: journalctl, syslog, auditd, Wazuh, Suricata, Zeek

Bab 6 yang menekankan pembacaan log langsung dari CLI memakai `journalctl`, syslog, auditd, Wazuh, Suricata, Zeek, dan korelasi manual. Bab ini juga meneruskan fondasi Bab 3–5: Wazuh untuk alert endpoint, Suricata untuk alert jaringan, dan Zeek untuk metadata trafik.

Secara teknis, `journalctl` adalah alat untuk menampilkan log yang disimpan oleh `systemd-journald`. (Man7.org) `auditd` adalah komponen userspace dari Linux Auditing System yang menulis audit record ke disk, sedangkan pembacaan dilakukan dengan `ausearch` dan `aureport`, serta konfigurasi rule dilakukan dengan `auditctl`. (Man7.org) Wazuh menyimpan alert secara default di `/var/ossec/logs/alerts/alerts.log` dan `/var/ossec/logs/alerts/alerts.json`. (Dokumentasi Wazuh) Suricata EVE JSON menghasilkan alert, anomaly, metadata, file info, dan record protokol dalam format JSON. (Suricata User Guide) Zeek menyediakan log seperti `conn.log`, `dns.log`, dan `http.log` untuk membaca koneksi, DNS, dan HTTP. (Dokumentasi Zeek)

6.1 Prinsip Membaca Log

Dashboard mempercepat visualisasi.

CLI mempercepat pembuktian.

Dashboard membantu melihat tren. CLI membantu melihat bukti mentah.

> **SOC yang kuat tidak bergantung penuh pada dashboard.**

Seorang analis harus bisa menjawab:

- log mentahnya di mana?
- field pentingnya apa?
- kapan kejadian terjadi?
- host mana yang terlibat?
- IP sumber siapa?
- user siapa?
- service apa?
- rule apa?
- severity berapa?
- apakah ada bukti dari sumber lain?

Alur Berpikir Membaca Log

```
pertanyaan
→ sumber log
→ filter waktu
→ filter host / IP / user
→ baca event mentah
→ cocokkan sumber lain
→ buat timeline
→ simpan evidence
→ tulis kesimpulan
```

Prinsip 1 — Mulai dari Waktu

Setiap investigasi harus dimulai dari waktu.
Contoh:

```
Kejadian diduga: 2026-06-02 10:15 WIB
Rentang analisis: 10:00–10:30 WIB
```

Tanpa waktu, pencarian log akan melebar dan melelahkan.

Prinsip 2 — Jangan Hanya Cari Alert

Alert adalah hasil interpretasi.
Log mentah adalah bukti awal.

Cari keduanya:

- alert;
- log autentikasi;
- log service;
- log kernel;
- log audit;
- log jaringan;
- metadata Zeek;
- PCAP bila perlu.

Prinsip 3 — Gunakan Filter Sederhana

Tool utama:

Tool	Fungsi
journalctl	Membaca log systemd
tail	Mengikuti log real-time
grep	Mencari string
awk	Memilih kolom
jq	Membaca JSON
ausearch	Mencari audit event
aureport	Ringkasan audit
zeek-cut	Memilih field log Zeek
tcpdump	Membaca paket jaringan

Prinsip 4 — Simpan Evidence

Jangan hanya membaca log di layar.
Simpan output penting:

```
mkdir -p ~/soc-lab/evidence/bab6
```

Contoh:

```
sudo grep "Failed password" /var/log/auth.log \
| tee ~/soc-lab/evidence/bab6/failed-password.log
```

Jika tidak disimpan, sulit dipakai untuk laporan.

6.2 Log Sistem Ubuntu

`journalctl` adalah pintu utama membaca log sistem pada distribusi Linux modern yang memakai `systemd`.

Membaca Semua Journal

```
journalctl
```

Makna:

- membaca log dari journal;
- output biasanya sangat panjang;
- gunakan untuk eksplorasi awal.

Keluar dari tampilan:

```
q
```

Membaca Error Terbaru

```
journalctl -xe
```

Makna:

- -x menambahkan penjelasan jika tersedia;
- -e langsung ke bagian akhir log.

Gunakan saat:

- service gagal;
- konfigurasi error;
- server baru restart;
- agent tidak aktif.

Membaca Log SSH

```
journalctl -u ssh
```

Makna:

- -u berarti unit systemd;
- ssh adalah service SSH.

Untuk mengikuti real-time:
journalctl -u ssh -f

Membaca Log Wazuh Agent

Di endpoint:

```
journalctl -u wazuh-agent
```

Real-time:

```
journalctl -u wazuh-agent -f
```

Gunakan saat:

- agent disconnected;
- manager IP salah;
- service restart;
- koneksi gagal.

Membaca Log 1 Jam Terakhir

```
journalctl --since "1 hour ago"
```

Contoh untuk SSH:

```
journalctl -u ssh --since "1 hour ago"
```

Membaca Log Hari Ini

```
journalctl --since today
```

Contoh untuk Wazuh Agent:

```
journalctl -u wazuh-agent --since today
```

Kombinasi Waktu

```
journalctl --since "2026-06-02 10:00:00" --until
"2026-06-02 10:30:00"
```

Gunakan format waktu konkret saat investigasi.

SOC tidak boleh kabur soal waktu.

Menyimpan Log Systemd

```
journalctl -u ssh --since "1 hour ago" \
| tee ~/soc-lab/evidence/bab6/ssh-last-hour.log
```

Tips Praktis journalctl

Kebutuhan	Perintah
Log service	journalctl -u nama-service
Log real-time	journalctl -f
Log kernel	journalctl -k
Log sejak waktu tertentu	journalctl --since "1 hour ago"
Log hari ini	journalctl --since today
Log service real-time	journalctl -u ssh -f

6.3 Log Autentikasi

Log autentikasi adalah salah satu sumber paling penting untuk SOC.

Di Ubuntu, log autentikasi umum berada di:

```
/var/log/auth.log
```

Log ini berguna untuk membaca:

- login SSH gagal;
- login SSH berhasil;
- penggunaan sudo;
- user tidak valid;
- percobaan privilege;
- aktivitas PAM;
- perubahan session.

Membaca Real-Time

```
sudo tail -f /var/log/auth.log
```

Gunakan saat melakukan simulasi login gagal di lab.

Mencari Login Gagal

```
sudo grep "Failed password" /var/log/auth.log
```

Makna:

- mencari string Failed password;
- biasanya terkait percobaan SSH gagal.

Simpan evidence:

```
sudo grep "Failed password" /var/log/auth.log \
| tee ~/soc-lab/evidence/bab6/auth-failed-password.log
```

Mencari Login Berhasil

```
sudo grep "Accepted password" /var/log/auth.log
```

Login berhasil setelah banyak login gagal adalah indikator penting.
Cari kombinasi:

```
sudo grep -E "Failed password|Accepted password"
/var/log/auth.log
```

Mencari Aktivitas sudo

```
sudo grep "sudo" /var/log/auth.log
```

Aktivitas sudo penting karena menunjukkan upaya menjalankan perintah dengan hak lebih tinggi.
Ambil ringkasan:

```
sudo grep "sudo" /var/log/auth.log | tail -n 20
```

Mencari User Tidak Valid

```
sudo grep "Invalid user" /var/log/auth.log
```

Contoh user yang sering dicoba:

- admin;
- test;
- oracle;
- postgres;
- ubuntu;
- root.

Menghitung IP Login Gagal

```
sudo grep "Failed password" /var/log/auth.log \
| awk '{print $(NF-3)}' \
| sort \
| uniq -c \
| sort -nr
```

Catatan:

- format log dapat berbeda;
- posisi kolom IP bisa berubah;
- validasi manual tetap wajib.

Versi Lebih Aman dengan grep IP

Jika sudah tahu IP mencurigakan:

```
sudo grep "<IP_MENCURIGAKAN>" /var/log/auth.log
```

Contoh:

```
sudo grep "192.168.56.20" /var/log/auth.log
```

Autentikasi adalah gerbang. Log autentikasi adalah kamera di gerbang.

6.4 Log Kernel dan Network

Log kernel membantu membaca kejadian level rendah.

Network command membantu memahami kondisi jaringan saat investigasi.

Membaca Kernel Ring Buffer

```
sudo dmesg -T
```

Makna:

- dmesg membaca pesan kernel;
- -T menampilkan waktu yang mudah dibaca manusia.

Gunakan untuk:

- error driver;
- interface jaringan naik/turun;
- disk error;
- firewall kernel log;
- pesan hardware.

Membaca Log Kernel dari Journal

```
sudo journalctl -k
```

Untuk 1 jam terakhir:

```
sudo journalctl -k --since "1 hour ago"
```

Cek IP Address

```
ip a
```

Pertanyaan:

- interface aktif?
- IP benar?
- ada IP ganda?
- interface down?
- interface sensor punya IP atau tidak?

Cek Routing

```
ip route
```

Pertanyaan:

- default gateway benar?
- rute ke server SOC ada?
- rute ke endpoint ada?
- trafik lewat jalur yang benar?

Cek Port Aktif

```
ss -tulpn
```

Makna umum:

- -t TCP;
- -u UDP;
- -l listening;
- -p proses;
- -n angka, tidak resolve nama.

Cari SSH:

```
sudo ss -tulpn | grep ssh
```

Cari Wazuh:

```
sudo ss -tulpn | grep -E '1514|1515|55000'
```

Capture Trafik Sementara

```
sudo tcpdump -i any
```

Lebih aman batasi:
sudo tcpdump -i any -nn -c 50

Makna:

- -i any membaca semua interface;
- -nn tidak resolve nama host/port;
- -c 50 berhenti setelah 50 paket.

Filter IP:

```
sudo tcpdump -i any -nn host <IP_MENCURIGAKAN>
```

Filter port SSH:

```
sudo tcpdump -i any -nn port 22
```

Simpan PCAP Kecil

```
sudo tcpdump -i any -nn -c 200 -w
~/soc-lab/evidence/bab6/network-sample.pcap
```

Baca ulang:

```
tcpdump -nn -r
~/soc-lab/evidence/bab6/network-sample.pcap
```

Network troubleshooting dan security investigation sering bertemu di perintah yang sama.

6.5 auditd untuk Monitoring Aktivitas Sistem

`auditd` dipakai untuk audit aktivitas sistem Linux.

Ia berguna untuk memantau kejadian penting seperti:

- login;
- autentikasi;
- perubahan file;
- akses file sensitif;
- perubahan konfigurasi;
- event keamanan kernel;
- aktivitas user penting.

Instalasi auditd

```
sudo apt install -y auditd audispd-plugins
```

Aktifkan:

```
sudo systemctl enable --now auditd
```

Cek status:

```
sudo systemctl status auditd
```

Cek status audit:

```
sudo auditctl -s
```

Makna umum output:

- enabled menunjukkan audit aktif;
- backlog menunjukkan antrean event;
- lost menunjukkan event hilang;
- failure menunjukkan mode kegagalan.

Mencari Event Login

```
sudo ausearch -m USER_LOGIN
```

Mencari Event Autentikasi

```
sudo ausearch -m USER_AUTH
```

Membaca Audit Log Langsung

Lokasi umum:

```
/var/log/audit/audit.log
```

Baca:

```
sudo tail -f /var/log/audit/audit.log
```

Ringkasan Audit

```
sudo aureport
```

Ringkasan autentikasi:

```
sudo aureport -au
```

Ringkasan file:

```
sudo aureport -f
```

Prinsip auditd

auditd bukan pengganti Wazuh.
auditd adalah sumber bukti tambahan.
Wazuh dapat membaca dan membuat alert.
auditd memberi jejak audit yang lebih detail untuk sistem Linux.
Untuk investigasi serius, auditd memperkuat bukti.

6.6 Membuat Rule auditd Sederhana

Tujuan praktek ini adalah memantau perubahan `/etc/passwd`.

File ini penting karena berisi daftar user sistem.

Membuat Rule Sementara

```
sudo auditctl -w /etc/passwd -p wa -k passwd_change
```

Penjelasan:

- auditctl mengatur rule audit;
- -w /etc/passwd memantau file /etc/passwd;
- -p wa memantau write dan attribute change;
- -k passwd_change memberi key agar mudah dicari.

Cek Rule

```
sudo auditctl -l
```

Uji dengan Perubahan Aman

Jangan sembarang mengubah `/etc/passwd`.
Gunakan perintah yang aman dan terkontrol.
Contoh melihat metadata tidak memicu write. Untuk uji perubahan sebenarnya sebaiknya dilakukan di VM lab.

Alternatif aman: pantau file dummy.

```
sudo touch /etc/test-audit.conf
sudo auditctl -w /etc/test-audit.conf -p wa -k
test_audit_change
echo "uji audit" | sudo tee -a /etc/test-audit.conf
```

Cari event:

```
sudo ausearch -k test_audit_change
```

Hapus file dummy:

```
sudo rm /etc/test-audit.conf
```

Cari lagi:

```
sudo ausearch -k test_audit_change
```

Mencari Rule /etc/passwd

```
sudo ausearch -k passwd_change
```

Membuat Report File

```
sudo aureport -f
```

Simpan Evidence

```
sudo ausearch -k passwd_change \
| tee ~/soc-lab/evidence/bab6/audit-passwd-change.log
```

Untuk dummy:

```
sudo ausearch -k test_audit_change \
| tee ~/soc-lab/evidence/bab6/audit-test-file.log
```

Menghapus Rule Sementara

```
sudo auditctl -W /etc/passwd -p wa -k passwd_change
```

Untuk dummy:

```
sudo auditctl -W /etc/test-audit.conf -p wa -k
test_audit_change
```

Catatan Persisten

Rule `auditctl` yang dibuat dari CLI bersifat sementara dan dapat hilang setelah reboot.

Untuk rule permanen, gunakan file rule audit sesuai distribusi dan kebijakan sistem, lalu load melalui mekanisme audit yang benar.

Untuk lab, rule sementara cukup. Untuk produksi, rule harus terdokumentasi dan persisten.

6.7 Membaca Wazuh Alert JSON

Wazuh menyimpan alert dalam format teks dan JSON. Format JSON sangat penting untuk CLI, otomasi, dan Python.

Lokasi default:

```
/var/ossec/logs/alerts/alerts.json
```

Membaca Real-Time

```
sudo tail -f /var/ossec/logs/alerts/alerts.json
```

Agar rapi:

```
sudo tail -f /var/ossec/logs/alerts/alerts.json | jq .
```

Filter Alert Level Tinggi

```
sudo jq 'select(.rule.level >= 10)'
/var/ossec/logs/alerts/alerts.json
```

Makna:

- pilih alert dengan level 10 ke atas;
- berguna untuk prioritas triage.

Ambil Field Penting

```
sudo jq '.agent.name, .rule.description, .rule.level'
/var/ossec/logs/alerts/alerts.json
```

Lebih rapi:

```
sudo jq '{
  time: .timestamp,
  agent: .agent.name,
  agent_ip: .agent.ip,
  rule: .rule.id,
  level: .rule.level,
  description: .rule.description
}' /var/ossec/logs/alerts/alerts.json
```

Cari Alert Agent Tertentu

```
sudo jq 'select(.agent.name=="endpoint-01")'
/var/ossec/logs/alerts/alerts.json
```

Cari IP Mencurigakan

```
sudo grep "<IP_MENCURIGAKAN>"
/var/ossec/logs/alerts/alerts.json
```

Atau dengan jq jika field tersedia:

```
sudo jq 'select(.data.srcip=="<IP_MENCURIGAKAN>")'
/var/ossec/logs/alerts/alerts.json
```

Simpan Alert Level Tinggi

```
sudo jq 'select(.rule.level >= 10)'
/var/ossec/logs/alerts/alerts.json \
| tee ~/soc-lab/evidence/bab6/wazuh-high-alerts.json
```

Ringkas Wazuh dengan Python

Buat script:

```
nano ~/soc-lab/scripts/ringkas_wazuh_alert.py
```

Isi:

```
from pathlib import Path
import json
from collections import Counter

file_path = Path("/var/ossec/logs/alerts/alerts.json")

agent_counter = Counter()
rule_counter = Counter()
level_counter = Counter()

with file_path.open("r", encoding="utf-8", errors="ignore") as
f:
    for line in f:
        try:
            event = json.loads(line)
        except json.JSONDecodeError:
            continue

        agent = event.get("agent", {}).get("name", "-")
        rule = event.get("rule", {}).get("description", "-")
        level = str(event.get("rule", {}).get("level", "-"))

        agent_counter[agent] += 1
        rule_counter[rule] += 1
        level_counter[level] += 1

print("Top Agent")
print("=" * 30)
for item, count in agent_counter.most_common(10):
    print(f"{item}: {count}")
```

```
print("\nTop Rule")
print("=" * 30)
for item, count in rule_counter.most_common(10):
    print(f"{item}: {count}")

print("\nLevel")
print("=" * 30)
for item, count in level_counter.most_common():
    print(f"{item}: {count}")

Jalankan:
sudo python3 ~/soc-lab/scripts/ringkas_wazuh_alert.py
```

6.8 Membaca Suricata Eve JSON

Suricata `eve.json` adalah sumber utama untuk alert dan metadata jaringan.

Lokasi umum:

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

Filter Alert

```
sudo jq 'select(.event_type=="alert")'
/var/log/suricata/eve.json
```

Ambil Field Penting

```
sudo jq '.src_ip, .dest_ip, .alert.signature'
/var/log/suricata/eve.json
```

Lebih rapi:

```
sudo jq 'select(.event_type=="alert") | {
  time: .timestamp,
  src_ip: .src_ip,
  src_port: .src_port,
  dest_ip: .dest_ip,
  dest_port: .dest_port,
  proto: .proto,
  signature: .alert.signature,
  category: .alert.category,
  severity: .alert.severity,
  flow_id: .flow_id
}' /var/log/suricata/eve.json
```

Cari IP Mencurigakan

```
sudo jq 'select(.src_ip=="<IP_MENCURIGAKAN>" or
.dest_ip=="<IP_MENCURIGAKAN>")' \
/var/log/suricata/eve.json
```

Atau:

```
sudo grep "<IP_MENCURIGAKAN>" /var/log/suricata/eve.json
```

Cari Alert Severity Tertentu

```
sudo jq 'select(.event_type=="alert" and .alert.severity
<= 2)' \
/var/log/suricata/eve.json
```

Catatan:

- beberapa rule memakai angka severity lebih kecil untuk prioritas lebih tinggi;
- pahami skema rule yang digunakan.

Simpan Evidence Suricata

```
sudo jq 'select(.event_type=="alert")'
/var/log/suricata/eve.json \
| tee ~/soc-lab/evidence/bab6/suricata-alerts.json
```

Ringkasan Source IP

```
sudo jq -r 'select(.event_type=="alert") | .src_ip'
/var/log/suricata/eve.json \
| sort \
| uniq -c \
| sort -nr
```

Ringkasan Signature

```
sudo jq -r 'select(.event_type=="alert") | .alert.signature'
/var/log/suricata/eve.json \
| sort \
| uniq -c \
| sort -nr
```

Suricata memberi sinyal jaringan. Tugas analis adalah mencari konteksnya.

6.9 Membaca Zeek Log

Zeek log biasanya berbentuk kolom. Tool zeek-cut memudahkan memilih field tertentu.

Lokasi umum:

```
/opt/zeek/logs/current/
```

Masuk ke folder:

```
cd /opt/zeek/logs/current/
```

Cek file:

```
ls -lah
```

Membaca conn.log

```
cat conn.log | zeek-cut id.orig_h id.resp_h id.resp_p
proto service
```

Makna field:

Field	Makna
id.orig_h	IP sumber
id.resp_h	IP tujuan
id.resp_p	port tujuan
proto	protokol
service	layanan yang dikenali

Ringkasan koneksi per sumber:

```
cat conn.log | zeek-cut id.orig_h \
| sort \
| uniq -c \
| sort -nr
```

Cari IP:

```
grep "<IP_MENCURIGAKAN>" conn.log | less
```

Membaca dns.log

```
cat dns.log | zeek-cut query answers
```

Lebih lengkap:

```
cat dns.log | zeek-cut id.orig_h query qtype_name
answers
```

Pertanyaan:

- host mana melakukan query?
- domain apa?
- jawaban apa?
- query berulang?
- domain terlihat acak?

Membaca http.log

```
cat http.log | zeek-cut host uri method status_code
```

Lebih lengkap:

```
cat http.log | zeek-cut id.orig_h id.resp_h host uri
method status_code user_agent
```

Pertanyaan:

- host apa yang diakses?
- URI apa?
- method apa?
- status code apa?
- user agent apa?

Membaca ssl.log

Jika tersedia:

```
cat ssl.log | zeek-cut id.orig_h id.resp_h server_name
version cipher
```

Membaca notice.log

```
cat notice.log | zeek-cut ts uid id.orig_h id.resp_h note msg
```

Simpan Evidence Zeek

```
grep "<IP_MENCURIGAKAN>" conn.log \
| tee ~/soc-lab/evidence/bab6/zeek-ip-conn.log
```

Untuk DNS:

```
grep "<IP_MENCURIGAKAN>" dns.log \
| tee ~/soc-lab/evidence/bab6/zeek-ip-dns.log
```

Jika zeek-cut Tidak Ada

Gunakan:

```
head conn.log
less conn.log
grep "<IP_MENCURIGAKAN>" conn.log
```

Namun zeek-cut sangat disarankan karena membuat field lebih mudah dibaca.

Zeek adalah alat membaca percakapan jaringan tanpa harus membuka semua paket.

6.10 Korelasi Manual dengan CLI

Korelasi manual adalah inti Bab 6.

Contoh investigasi:

- cari IP di Wazuh;
- cari IP di Suricata;
- cari IP di Zeek;
- cocokkan waktu;
- buat timeline.

Skenario

Misal IP mencurigakan:

```
192.168.56.20
```

Target:

```
192.168.56.30
```

Rentang waktu:

```
2026-06-02 10:00–10:30 WIB
```

1. Cari IP di Log Autentikasi

```
grep "<IP_MENCURIGAKAN>" /var/log/auth.log
```

Contoh:

```
grep "192.168.56.20" /var/log/auth.log
```

Simpan:

```
grep "192.168.56.20" /var/log/auth.log \
| tee ~/soc-lab/evidence/bab6/correlation-auth.log
```

2. Cari IP di Wazuh

```
grep "<IP_MENCURIGAKAN>"
/var/ossec/logs/alerts/alerts.json
```

Contoh:

```
sudo grep "192.168.56.20"
/var/ossec/logs/alerts/alerts.json \
| tee ~/soc-lab/evidence/bab6/correlation-wazuh.json
```

Jika ingin format rapi:

```
sudo grep "192.168.56.20"
/var/ossec/logs/alerts/alerts.json | jq .
```

3. Cari IP di Suricata

```
grep "<IP_MENCURIGAKAN>" /var/log/suricata/eve.json
```

Contoh:

```
sudo grep "192.168.56.20" /var/log/suricata/eve.json \
| tee ~/soc-lab/evidence/bab6/correlation-suricata.json
```

Dengan jq:

```
sudo jq 'select(.src_ip=="192.168.56.20" or
.dest_ip=="192.168.56.20")' \
/var/log/suricata/eve.json
```

4. Cari IP di Zeek

Masuk ke folder Zeek:

```
cd /opt/zeek/logs/current/
```

Cari di conn.log:

```
grep "192.168.56.20" conn.log \
| tee ~/soc-lab/evidence/bab6/correlation-zeek-conn.log
```

Cari di DNS:

```
grep "192.168.56.20" dns.log \
| tee ~/soc-lab/evidence/bab6/correlation-zeek-dns.log
```

Cari di HTTP:

```
grep "192.168.56.20" http.log \
| tee ~/soc-lab/evidence/bab6/correlation-zeek-http.log
```

5. Buat Timeline

Buat file:

```
nano ~/soc-lab/evidence/bab6/timeline-ip-192.168.56.20.txt
```

Isi:

```
# Timeline Investigasi IP 192.168.56.20

Tanggal:
Analis:
IP mencurigakan:
Target:
Rentang waktu:
```

Waktu	Sumber Log	Event	Bukti	Catatan
10:01	Zeek conn.log	Koneksi ke target	correlation-zeek-conn.log	Banyak port
10:02	Suricata eve.json	Alert scan	correlation-suricata.json	Signature port scan
10:03	auth.log	SSH gagal	correlation-auth.log	User tidak valid
10:04	Wazuh alerts.json	Alert auth	correlation-wazuh.json	Level perlu dicek

```
Kesimpulan:
IP 192.168.56.20 melakukan aktivitas mencurigakan di lab.
Perlu validasi apakah ini aktivitas admin, scanner internal,
atau simulasi.
```

6. Buat Ringkasan dengan Python

Buat script:

```
nano ~/soc-lab/scripts/buat_timeline_sederhana.py
```

Isi:

```
from pathlib import Path
from datetime import datetime

base = Path.home() / "soc-lab" / "evidence" / "bab6"
base.mkdir(parents=True, exist_ok=True)

items = [
    ("auth.log", "correlation-auth.log"),
    ("wazuh", "correlation-wazuh.json"),
    ("suricata", "correlation-suricata.json"),
    ("zeek_conn", "correlation-zeek-conn.log"),
    ("zeek_dns", "correlation-zeek-dns.log"),
    ("zeek_http", "correlation-zeek-http.log"),
]

output = base / "ringkasan-korelasi.txt"

lines = []
lines.append("# Ringkasan Korelasi Manual\n")
lines.append(f"Dibuat: {datetime.now().isoformat()}\n")

for label, filename in items:
    path = base / filename
    if path.exists():
        count = len(path.read_text(encoding="utf-8",
errors="ignore").splitlines())
        lines.append(f"- {label}: {count} baris
evidence")
    else:
        lines.append(f"- {label}: file belum ada")

output.write_text("\n".join(lines), encoding="utf-8")
print(f"Ringkasan dibuat: {output}")
```

Jalankan:

```
python3 ~/soc-lab/scripts/buat_timeline_sederhana.py
cat ~/soc-lab/evidence/bab6/ringkasan-korelasi.txt
```

Prinsip Korelasi Manual

Korelasi bukan sekadar mencari IP.

Korelasi harus menjawab:

- apakah waktu cocok?
- apakah host cocok?
- apakah pola cocok?
- apakah alert jaringan dan endpoint saling mendukung?
- apakah ada bukti DNS/HTTP?
- apakah ini aktivitas sah?
- apakah perlu dibuat case?

Korelasi mengubah potongan log menjadi cerita teknis.

6.11 Latihan Mini: Investigasi Login Gagal dan Port Scan

Tujuan latihan:

- membaca `auth.log`;
- membaca Wazuh alert;
- membaca Suricata alert;
- membaca Zeek connection;
- membuat timeline.

Langkah 1 — Siapkan Folder

```
mkdir -p ~/soc-lab/evidence/bab6/latihan-mini
```

Langkah 2 — Cari Login Gagal

```
sudo grep "Failed password" /var/log/auth.log \
| tee
~/soc-lab/evidence/bab6/latihan-mini/failed-password.log
```

Langkah 3 — Cari Alert Wazuh

```
sudo jq 'select(.rule.description |
test("authentication|password|login"; "i"))' \
/var/ossec/logs/alerts/alerts.json \
| tee
~/soc-lab/evidence/bab6/latihan-mini/wazuh-auth-alert.json
```

Langkah 4 — Cari Alert Suricata

```
sudo jq 'select(.event_type=="alert")'
/var/log/suricata/eve.json \
| tee
~/soc-lab/evidence/bab6/latihan-mini/suricata-alert.json
```

Langkah 5 — Cari Koneksi Zeek

```
cd /opt/zeek/logs/current/
cat conn.log | zeek-cut id.orig_h id.resp_h id.resp_p proto
service \
| tee
~/soc-lab/evidence/bab6/latihan-mini/zeek-conn-summary.log
```

Langkah 6 — Buat Kesimpulan

```
nano ~/soc-lab/evidence/bab6/latihan-mini/kesimpulan.txt
```

Isi:

```
# Kesimpulan Latihan Mini Bab 6

Pertanyaan:
Apakah ada indikasi login gagal atau scanning?

Bukti:
1. auth.log:
2. Wazuh alerts.json:
3. Suricata eve.json:
4. Zeek conn.log:

Analisis:
- IP sumber:
- IP target:
- Waktu:
- Port:
- User:
- Signature:
- Severity:

Kesimpulan sementara:
[ ] Aktivitas normal
[ ] Simulasi lab
[ ] Perlu investigasi lanjut
[ ] Incident

Catatan:
```

6.12 Checklist Validasi Akhir Bab 6

Log Sistem

- Bisa memakai `journalctl`.
- Bisa memakai `journalctl -xe`.
- Bisa membaca log service dengan `journalctl -u`.
- Bisa filter waktu dengan `--since`.
- Bisa membaca log kernel dengan `journalctl -k`.

Log Autentikasi

- Bisa membaca `/var/log/auth.log`.
- Bisa mencari `Failed password`.
- Bisa mencari `Accepted password`.
- Bisa mencari aktivitas `sudo`.
- Bisa menyimpan evidence autentikasi.

Kernel dan Network

- Bisa memakai `dmesg -T`.
- Bisa membaca `ip a`.
- Bisa membaca `ip route`.
- Bisa membaca `ss -tulpn`.
- Bisa capture paket dengan `tcpdump`.

auditd

- `auditd` terpasang.
- Service `auditd` aktif.
- Bisa memakai `auditctl -s`.
- Bisa memakai `ausearch`.
- Bisa memakai `aureport`.
- Bisa membuat rule audit sederhana.

Wazuh

- Bisa membaca `alerts.json`.
- Bisa filter level dengan `jq`.
- Bisa mengambil agent, rule, dan level.
- Bisa menyimpan alert sebagai evidence.

Suricata

- Bisa membaca `eve.json`.
- Bisa filter `event_type=="alert"`.
- Bisa mengambil source IP, destination IP, dan signature.
- Bisa menyimpan alert jaringan.

Zeek

- Bisa membaca `conn.log`.
- Bisa membaca `dns.log`.
- Bisa membaca `http.log`.
- Bisa memakai `zeek-cut`.
- Bisa mencari IP mencurigakan.

Korelasi

- Bisa mencari IP di auth.log.
- Bisa mencari IP di Wazuh.
- Bisa mencari IP di Suricata.
- Bisa mencari IP di Zeek.
- Bisa membuat timeline.
- Bisa menulis kesimpulan sementara.

6.13 Kesalahan Umum Bab 6

1. Membaca Log Tanpa Rentang Waktu

Akibat:

- terlalu banyak data;
- analisis melebar;
- kesimpulan kabur.

Gunakan:

```
journalctl --since "1 hour ago"
```

atau waktu konkret.

2. Hanya Mengandalkan Dashboard

Dashboard bagus untuk ringkasan.
Namun bukti teknis tetap harus diambil dari log.
Dashboard adalah peta. Log adalah medan.

3. Grep Tanpa Memahami Format

`grep` cepat, tetapi bisa menyesatkan.

Selalu cek:

- format log;
- posisi field;
- timestamp;
- timezone;
- host;
- konteks baris sebelum dan sesudah.

4. Membaca JSON Tanpa jq

JSON mentah sulit dibaca.

Gunakan:

```
jq .
```

Untuk filter:

```
jq 'select(...)'
```

5. Tidak Menyimpan Evidence

Output terminal mudah hilang.
Simpan dengan:

```
tee nama-file.log
```

6. Tidak Mencocokkan Sumber Lain

Satu log jarang cukup.

Korelasikan minimal:

- endpoint log;
- Wazuh alert;
- Suricata alert;
- Zeek metadata;
- waktu kejadian.

7. Lupa Timezone

SOC sangat sensitif pada waktu.
Pastikan:

```
timedatectl
```

6.14 Rangkuman Bab

Bab ini adalah bab inti untuk membangun insting analis SOC.

Pembaca belajar membaca log langsung dari CLI:

- `journalctl` untuk log sistem;
- `/var/log/auth.log` untuk autentikasi;
- `dmesg` dan `journalctl -k` untuk kernel;
- `ip`, `ss`, dan `tcpdump` untuk jaringan;
- `auditd`, `auditctl`, `ausearch`, dan `aureport` untuk audit sistem;
- Wazuh `alerts.json` untuk alert endpoint;
- Suricata `eve.json` untuk alert jaringan;
- Zeek `conn.log`, `dns.log`, dan `http.log` untuk metadata trafik;
- `grep`, `jq`, dan `zeek-cut` untuk filter cepat;
- Python untuk membuat ringkasan sederhana.

Inti Bab 6:

> **CLI bukan alat lama. CLI adalah alat pembuktian.**

SOC yang kuat bukan hanya melihat dashboard.

SOC yang kuat mampu membaca bukti mentah, memfilter event, mencocokkan waktu, menghubungkan sumber log, dan membuat timeline yang dapat dipertanggungjawabkan.

Output Bab

Setelah menyelesaikan Bab 6, pembaca mampu:

- mengakses log sistem Ubuntu dengan `journalctl`;
- membaca log autentikasi;
- membaca log kernel dan network;
- memakai auditd untuk audit aktivitas sistem;
- membuat rule audit sederhana;
- membaca Wazuh `alerts.json`;
- membaca Suricata `eve.json`;
- membaca Zeek `conn.log`, `dns.log`, dan `http.log`;
- mencari IP mencurigakan di banyak sumber log;
- membuat timeline investigasi;
- menyimpan evidence;
- membuat ringkasan sederhana dengan Python.

Kalimat kunci Bab 6:

> **Analis SOC yang kuat bukan yang paling banyak membuka dashboard, tetapi yang paling cepat menemukan bukti dari log mentah dan menyusunnya menjadi timeline yang masuk akal.**

Bab 7 — OpenSearch dan Dashboard SOC: Query, Index, Alert

Bab 7 yang menekankan OpenSearch, dashboard, index, field, query, panel minimum SOC, dan alert berbasis query. Secara resmi, Wazuh Indexer memakai pola index seperti `wazuh-alerts-*` untuk alert dari Wazuh server, `wazuh-archives-*` untuk event yang dikirim ke Wazuh server, dan `wazuh-monitoring-*` untuk status agent. (Dokumentasi Wazuh)

OpenSearch menyediakan REST API sehingga admin dapat memeriksa cluster, index, statistik, dan query data langsung dari terminal memakai `curl` atau program lain. (OpenSearch Documentation) OpenSearch Dashboards menyediakan antarmuka untuk eksplorasi, pencarian, dan visualisasi data observability, sedangkan fitur Alerting memakai konsep monitor dan trigger untuk menjalankan query berkala dan membuat alert jika kondisi terpenuhi. (OpenSearch Documentation)

7.1 Mengapa Log Perlu Diindeks

Log mentah sulit dianalisis jika tidak dapat dicari cepat.
File seperti ini penting:

```
/var/ossec/logs/alerts/alerts.json
/var/log/suricata/eve.json
/opt/zeek/logs/current/conn.log
/var/log/auth.log
```

Namun jika ukurannya makin besar, pencarian manual menjadi lambat.
Contoh:

```
grep "192.168.56.20" /var/ossec/logs/alerts/alerts.json
```

Perintah ini berguna untuk file kecil.
Tetapi untuk jutaan event, admin butuh index.

Index membuat log bisa dicari cepat, di filter, divisualisasikan, dan dijadikan alert.

Log Mentah vs Log Terindeks

Aspek	Log Mentah	Log Terindeks
Lokasi	File lokal	Index OpenSearch
Pencarian	grep, jq, awk	REST API, query, dashboard
Kecepatan	Menurun saat file besar	Lebih cepat untuk data besar
Visualisasi	Manual	Dashboard
Alert	Script manual	Monitor / query alert
Korelasi	Manual	Lebih mudah lintas field
Retensi	Tergantung file	Dapat diatur dengan kebijakan index

Mengapa Penting untuk SOC

SOC harus menjawab cepat:

- alert level tinggi apa yang muncul hari ini?
- agent mana paling banyak menghasilkan alert?
- IP sumber mana paling sering muncul?
- ada berapa login gagal dalam 10 menit?
- apakah `/etc/passwd` berubah?
- apakah ada port scan?
- apakah ada koneksi ke IOC?
- apakah teknik MITRE tertentu muncul?

Tanpa index, semua ini berat.

Dengan index, admin bisa membuat query.

SOC tidak cukup hanya mengumpulkan log. SOC harus bisa mencari log dengan cepat.

7.2 Instalasi OpenSearch dan OpenSearch Dashboards

Topik utama:

- repository;
- service;
- password admin;
- port;
- TLS dasar;
- akses dashboard.

Untuk lab Wazuh, OpenSearch biasanya hadir sebagai **Wazuh Indexer**. Jika mengikuti instalasi Wazuh *all-in-one* pada Bab 3, komponen indexer dan dashboard sudah terpasang bersama Wazuh. Namun bab ini tetap menjelaskan konsep OpenSearch agar admin memahami apa yang terjadi di belakang dashboard.

OpenSearch menyediakan paket instalasi untuk host Linux, termasuk konfigurasi port dan pengaturan penting host. (OpenSearch Documentation) OpenSearch Dashboards juga dapat dipasang melalui repository APT pada sistem berbasis Debian/Ubuntu. (OpenSearch Documentation)

Dua Skenario Instalasi

Skenario	Dipakai Saat	Catatan
Wazuh all-in-one	Lab SOC buku ini	Lebih sederhana
OpenSearch mandiri	Integrasi khusus / riset	Perlu konfigurasi lebih detail

Untuk buku ini, pendekatan utama:

```
Wazuh Manager
→ Wazuh Indexer / OpenSearch
→ Wazuh Dashboard
→ Query
→ Dashboard
→ Alert
```

Komponen yang Harus Dipahami

Komponen	Fungsi
OpenSearch / Wazuh Indexer	Menyimpan dan mencari log
OpenSearch Dashboards / Wazuh Dashboard	Visualisasi dan eksplorasi
Index	Kumpulan dokumen log
Field	Kolom data dalam dokumen
Mapping	Definisi tipe field
Query	Pencarian data
Monitor	Query berkala untuk alert
Trigger	Syarat yang memicu alert

Port Penting

Port	Fungsi
9200	API OpenSearch / Wazuh Indexer
5601 atau port dashboard terkait	OpenSearch Dashboards
443 pada beberapa setup	Dashboard via HTTPS
55000	Wazuh API

Catatan:

- port dapat berbeda sesuai instalasi;
- jangan membuka port indexer ke jaringan publik;
- batasi akses hanya dari host admin atau jaringan management;
- gunakan TLS dan autentikasi.

Password Admin

Catat dengan aman:

- user admin;
- password;
- URL dashboard;
- URL API;
- lokasi sertifikat;
- tanggal perubahan password.

Buat catatan:

```
nano ~/soc-lab/docs/catatan-opensearch.txt
```

Isi:

```
# Catatan OpenSearch / Wazuh Indexer

Tanggal:
Hostname:
IP:
Port API:
Port dashboard:
User admin:
Lokasi password:
TLS:
Catatan firewall:
Catatan error:
```

Jangan menyimpan password di file yang bisa dibaca publik.

7.3 Cek Service OpenSearch

Setelah instalasi, admin wajib memeriksa service.
Jika memakai OpenSearch mandiri:

```
sudo systemctl status opensearch
```

Jika memakai OpenSearch Dashboards mandiri:

```
sudo systemctl status opensearch-dashboards
```

Jika memakai Wazuh all-in-one, cek:

```
sudo systemctl status wazuh-indexer
sudo systemctl status wazuh-dashboard
```

Cek API OpenSearch

Contoh dari outline:

```
curl -k https://localhost:9200
```

Jika butuh autentikasi:

```
curl -k -u admin:<password> https://localhost:9200
```

Penjelasan:

- curl mengakses REST API;
- -k menerima sertifikat TLS self-signed di lab;
- -u admin:<password> memakai autentikasi;
- https://localhost:9200 adalah endpoint API lokal.

Cek Port

```
sudo ss -tulpn | grep -E '9200|5601|443'
```

Makna:

- port 9200 untuk API indexer;
- port dashboard tergantung instalasi;
- port 443 bisa dipakai jika dashboard dipublikasikan via HTTPS lokal.

Cek Health Cluster

```
curl -k -u admin:<password>
https://localhost:9200/_cluster/health?pretty
```

API cluster health memberikan ringkasan status operasional cluster. OpenSearch memakai status warna untuk menggambarkan kondisi alokasi shard. (OpenSearch Documentation)
Contoh output:

```
{
  "cluster_name": "wazuh-cluster",
  "status": "green",
  "number_of_nodes": 1,
  "active_primary_shards": 12,
  "active_shards": 12
}
```

Makna umum:

Status	Makna
green	Semua shard utama dan replika sehat
yellow	Shard utama sehat, replika belum lengkap
red	Ada shard utama bermasalah

Untuk lab single-node, status `yellow` bisa terjadi karena replika tidak tersedia. Namun status `red` harus diperiksa serius.

Cek Log Service

```
sudo journalctl -u wazuh-indexer -n 100 --no-pager
sudo journalctl -u wazuh-dashboard -n 100 --no-pager
```

Jika memakai OpenSearch mandiri:

```
sudo journalctl -u opensearch -n 100 --no-pager
sudo journalctl -u opensearch-dashboards -n 100
--no-pager
```

Catatan Validasi

Buat file:

```
nano ~/soc-lab/docs/validasi-opensearch.txt
```

Isi:

```
# Validasi OpenSearch / Wazuh Indexer

Tanggal:
Analis:

Service:
[ ] indexer aktif
[ ] dashboard aktif
[ ] port 9200 aktif
[ ] dashboard dapat diakses
[ ] API bisa diakses
[ ] cluster health dicek
[ ] index Wazuh muncul
[ ] query dasar berhasil

Catatan:
```

7.4 Index, Field, Mapping

Sebelum membuat query, admin harus memahami tiga istilah:

- **index**
- **field**
- **mapping**

Index

Index adalah kumpulan dokumen yang saling terkait.
Dalam Wazuh, pola index penting:

```
wazuh-alerts-*
wazuh-archives-*
wazuh-monitoring-*
```

Wazuh memakai `wazuh-alerts-*` untuk alert yang dihasilkan oleh Wazuh server, `wazuh-archives-*` untuk semua event yang dikirim ke server, dan `wazuh-monitoring-*` untuk status agent. (Dokumentasi Wazuh)

Field

Field adalah bagian data dalam dokumen.
Field penting:

- `timestamp`
- `host`
- `agent`
- `source_ip`
- `destination_ip`
- `rule_id`
- `severity`
- `event_type`
- `mitre_tactic`
- `mitre_technique`

Pada Wazuh, nama field nyata bisa berbeda, misalnya:

```
timestamp
agent.name
agent.ip
rule.id
rule.level
rule.description
data.srcip
data.dstip
rule.mitre.tactic
rule.mitre.technique
```

Selalu cek data asli.

Mapping

Mapping menentukan tipe data field.

Contoh tipe:

Tipe	Contoh
tanggal	timestamp
angka	rule.level
teks	rule.description
keyword	agent.name
IP	data.srcip

Mapping penting karena mempengaruhi:

- pencarian;
- sorting;
- aggregation;
- dashboard;
- alert.

Cek Index

```
curl -k -u admin:<password>
https://localhost:9200/_cat/indices?v
```

CAT indices API menampilkan informasi index seperti ukuran disk, jumlah shard, dan status kesehatan index. (OpenSearch Documentation)

Contoh output:

```
health status index                       docs.count store.size
green  open   wazuh-alerts-4.x-2026.06.02 1200       8mb
```

Cek Mapping

```
curl -k -u admin:<password> \
"https://localhost:9200/wazuh-alerts-*/_mapping?pretty"
\
| less
```

Cari field tertentu:

```
curl -k -u admin:<password> \
"https://localhost:9200/wazuh-alerts-*/_mapping?pretty"
\
| grep -n "rule.level"
```

Cek Satu Dokumen Alert

```
curl -k -u admin:<password> -X GET \
"https://localhost:9200/wazuh-alerts-*/_search?size=1&pretty"
```

Tujuan:

- melihat struktur field;
- memastikan nama field benar;
- menghindari query salah;
- memahami isi dokumen.

Jangan membuat dashboard sebelum memahami field.

7.5 Query Dasar dengan curl

OpenSearch dapat diakses dengan REST API.

Ini penting karena admin SOC harus bisa query dari CLI.

Melihat Daftar Index

```
curl -k -u admin:<password>
https://localhost:9200/_cat/indices?v
```

Versi lebih rapi:

```
curl -k -u admin:<password>
"https://localhost:9200/_cat/indices?v&s=index"
```

Melihat Cluster Health

```
curl -k -u admin:<password>
https://localhost:9200/_cluster/health?pretty
```

Melihat Informasi Cluster

```
curl -k -u admin:<password> https://localhost:9200
```

Query Sederhana ke Wazuh Alerts

```
curl -k -u admin:<password> -X GET \
"https://localhost:9200/wazuh-alerts-*/_search?pretty"
```

Secara default, output bisa panjang. Batasi ukuran:

```
curl -k -u admin:<password> -X GET \
"https://localhost:9200/wazuh-alerts-*/_search?size=5&pr
etty"
```

Query dengan Body JSON

Buat file query:

```
nano ~/soc-lab/queries/query-alert-terbaru.json
```

Isi:

```
{
  "size": 5,
  "sort": [
    {
      "timestamp": {
        "order": "desc"
      }
    }
  ]
}
```

Jalankan:

```
curl -k -u admin:<password> -X GET \
"https://localhost:9200/wazuh-alerts-*/_search?pretty" \
-H "Content-Type: application/json" \
-d @~/soc-lab/queries/query-alert-terbaru.json
```

Ambil Field Penting dengan jq

```
curl -k -u admin:<password> -X GET \
"https://localhost:9200/wazuh-alerts-*/_search?size=5&pretty"
\
| jq '.hits.hits[]._source | {
  time: .timestamp,
  agent: .agent.name,
  level: .rule.level,
  rule: .rule.description
}'
```

Membuat Folder Query

```
mkdir -p ~/soc-lab/queries
```

Simpan semua query di folder ini.
Prinsip:

- satu query satu file;
- nama file jelas;
- query bisa diuji ulang;
- query bisa dipakai untuk laporan.

Contoh nama file:

```
query-alert-terbaru.json
query-level-tinggi.json
query-failed-login.json
query-fim-passwd.json
query-portscan.json
```

7.6 Query Alert Severity Tinggi

Target bagian ini adalah mencari alert prioritas tinggi.
Contoh dari outline:

```
curl -k -u admin:<password> -X GET
"https://localhost:9200/wazuh-alerts-*/_search?pretty"
```

Kita perlu membuatnya lebih tajam.

Query Rule Level >= 10

Buat file:

```
nano ~/soc-lab/queries/query-rule-level-tinggi.json
```

Isi:

```
{
  "size": 20,
  "query": {
    "range": {
      "rule.level": {
        "gte": 10
      }
    }
  },
  "sort": [
    {
      "timestamp": {
        "order": "desc"
      }
    }
  ]
}
```

Jalankan:

```
curl -k -u admin:<password> -X GET \
"https://localhost:9200/wazuh-alerts-*/_search?pretty" \
-H "Content-Type: application/json" \
-d @~/soc-lab/queries/query-rule-level-tinggi.json
```

Ambil ringkasan:

```
curl -k -u admin:<password> -X GET \
"https://localhost:9200/wazuh-alerts-*/_search?pretty" \
-H "Content-Type: application/json" \
-d @~/soc-lab/queries/query-rule-level-tinggi.json \
| jq '.hits.hits[]._source | {
  time: .timestamp,
  agent: .agent.name,
  ip: .agent.ip,
  level: .rule.level,
  rule_id: .rule.id,
  description: .rule.description
}'
```

Query Alert 1 Jam Terakhir

```
nano ~/soc-lab/queries/query-alert-1jam.json
```

Isi:

```
{
  "size": 50,
  "query": {
    "range": {
      "timestamp": {
        "gte": "now-1h",
        "lte": "now"
      }
    }
  },
  "sort": [
    {
      "timestamp": {
        "order": "desc"
      }
    }
  ]
}
```

Jalankan:

```
curl -k -u admin:<password> -X GET \
"https://localhost:9200/wazuh-alerts-*/_search?pretty" \
-H "Content-Type: application/json" \
-d @~/soc-lab/queries/query-alert-1jam.json
```

Query Level Tinggi dalam 1 Jam

```
nano ~/soc-lab/queries/query-level-tinggi-1jam.json
```

Isi:

```
{
  "size": 50,
  "query": {
    "bool": {
      "must": [
        {
          "range": {
            "rule.level": {
              "gte": 10
            }
          }
        },
        {
          "range": {
            "timestamp": {
              "gte": "now-1h",
              "lte": "now"
            }
          }
        }
      ]
    }
  },
  "sort": [
    {
      "timestamp": {
        "order": "desc"
      }
    }
  ]
}
```

Export Evidence

```
mkdir -p ~/soc-lab/evidence/opensearch
curl -k -u admin:<password> -X GET \
"https://localhost:9200/wazuh-alerts-*/_search?pretty" \
-H "Content-Type: application/json" \
-d @~/soc-lab/queries/query-level-tinggi-1jam.json \
| tee ~/soc-lab/evidence/opensearch/high-alerts-1h.json
```

Ringkas dengan Python

Buat script:

```
nano ~/soc-lab/scripts/ringkas_opensearch_alert.py
```

Isi:

```
from pathlib import Path
import json
from collections import Counter

file_path = Path.home() / "soc-lab" / "evidence" / "opensearch"
/ "high-alerts-1h.json"

if not file_path.exists():
    raise FileNotFoundError(f"File tidak ditemukan:
{file_path}")

data = json.loads(file_path.read_text(encoding="utf-8"))

hits = data.get("hits", {}).get("hits", [])

agent_counter = Counter()
rule_counter = Counter()
level_counter = Counter()

for hit in hits:
    source = hit.get("_source", {})
    agent = source.get("agent", {}).get("name", "-")
    rule = source.get("rule", {}).get("description", "-")
    level = str(source.get("rule", {}).get("level", "-"))

    agent_counter[agent] += 1
    rule_counter[rule] += 1
    level_counter[level] += 1

print("Ringkasan Alert Level Tinggi")
print("=" * 40)

print("\nTop Agent")
for agent, count in agent_counter.most_common(10):
    print(f"{agent}: {count}")

print("\nTop Rule")
for rule, count in rule_counter.most_common(10):
    print(f"{rule}: {count}")

print("\nLevel")
for level, count in level_counter.most_common():
    print(f"{level}: {count}")
```

Jalankan:

```
python3 ~/soc-lab/scripts/ringkas_opensearch_alert.py
```

Query yang baik harus bisa diulang, diuji, dan dijadikan evidence.

7.7 Dashboard Minimum SOC

Dashboard SOC bukan hiasan.

Dashboard harus menjawab pertanyaan operasional.

Panel minimum:

- alert per severity;
- top agent;
- top source IP;
- failed login;
- FIM alert;
- Suricata alert;
- timeline incident.

OpenSearch Dashboards dipakai untuk mengeksplorasi, menemukan, dan memvisualisasikan data observability. (OpenSearch Documentation) Wazuh juga menyediakan dashboard untuk OpenSearch dan memakai index pattern `wazuh-alerts-*` agar alert Wazuh dapat divisualisasikan. (Dokumentasi Wazuh)

Prinsip Dashboard Minimum

Dashboard harus:

- sederhana;
- langsung menjawab pertanyaan SOC;
- tidak terlalu ramai;
- memakai field yang jelas;
- punya filter waktu;
- punya filter agent;
- punya filter severity;
- bisa membantu triage.

Dashboard buruk adalah dashboard yang indah tetapi tidak membantu keputusan.

Panel 1 — Alert per Severity

Tujuan:

- melihat tingkat risiko;
- melihat apakah alert tinggi meningkat;
- melihat prioritas triage.

Field:

```
rule.level
```

Visualisasi:

- bar chart;
- table;
- time series.

Pertanyaan analis:

- level tertinggi berapa?
- apakah level tinggi meningkat?
- rule apa yang dominan?

Panel 2 — Top Agent

Tujuan:

- melihat endpoint paling banyak menghasilkan alert.

Field:

```
agent.name
```

Pertanyaan:

- agent mana paling bising?
- apakah endpoint tertentu diserang?
- apakah konfigurasi agent terlalu sensitif?

Panel 3 — Top Source IP

Field bisa berbeda.

Contoh:

```
data.srcip
src_ip
source.ip
```

Pertanyaan:

- IP mana paling sering muncul?
- apakah IP internal atau eksternal?
- apakah IP scanner lab?
- apakah IP masuk whitelist?

Panel 4 — Failed Login

Query contoh:

```
rule.description contains "Failed password"
```

Atau field sesuai data:

```
rule.groups: authentication_failed
```

Pertanyaan:

- login gagal naik?
- user apa yang dicoba?
- sumber dari mana?
- ada login sukses setelahnya?

Panel 5 — FIM Alert

Query contoh:

```
rule.groups: syscheck
```

Atau path:

```
syscheck.path: /etc/passwd
```

Pertanyaan:

- file apa yang berubah?
- siapa endpointnya?
- kapan?
- apakah perubahan sah?

Panel 6 — Suricata Alert

Jika Suricata dikirim ke OpenSearch, panel berisi:

- signature;
- source IP;
- destination IP;
- severity;
- event_type.

Field contoh:

```
event_type
alert.signature
src_ip
dest_ip
```

Pertanyaan:

- signature apa yang dominan?
- IP mana yang melakukan scanning?
- port apa yang disasar?

Panel 7 — Timeline Incident

Timeline harus menggabungkan:

- waktu alert Wazuh;
- waktu alert Suricata;
- waktu metadata Zeek;
- waktu respons.

Panel ini membantu menjawab:

> “Apa urutan kejadian?”

Catatan Desain Dashboard

Buat file:

```
nano ~/soc-lab/docs/desain-dashboard-minimum-soc.txt
```

Isi:

```
# Desain Dashboard Minimum SOC

## Filter Global
- Time range
- Agent
- Severity
- Source IP
- Destination IP

## Panel
1. Alert per severity
2. Top agent
3. Top source IP
4. Failed login
5. FIM alert
6. Suricata alert
7. Timeline incident

## Tujuan
Dashboard dipakai untuk triage awal.
Validasi tetap dilakukan melalui CLI dan log mentah.
```

Dashboard adalah peta situasi. Bukan pengganti investigasi.

7.8 Membuat Alert Berbasis Query

Alert berbasis query membuat SOC lebih proaktif.

Contoh:

- login gagal > 10 kali;
- rule level >= 10;
- perubahan `/etc/passwd`;
- port scan terdeteksi;
- koneksi ke IOC.

OpenSearch Alerting memakai monitor untuk menjalankan query secara berkala, trigger untuk menentukan kondisi alert, dan action untuk tindakan setelah alert terpicu. (OpenSearch Documentation) Alerting API juga mendukung query-level monitor yang menjalankan query dan menentukan apakah hasilnya harus memicu alert. (OpenSearch Documentation)

Prinsip Alert Berbasis Query

Alert harus memenuhi syarat:

- jelas;
- bisa diuji;
- tidak terlalu bising;
- punya threshold;
- punya rentang waktu;
- punya severity;
- punya tindakan lanjut;
- bisa dijelaskan dalam laporan.

Alert buruk:

```
semua login gagal = alert kritikal
```

Alert lebih baik:

```
login gagal lebih dari 10 kali dari IP yang sama dalam 5
menit
```

Contoh 1 — Rule Level >= 10

Query:

```
{
  "query": {
    "range": {
      "rule.level": {
        "gte": 10
      }
    }
  }
}
```

Gunakan untuk alert prioritas tinggi.
Tindakan:

- kirim notifikasi;
- buat catatan triage;
- cek agent;
- simpan evidence.

Contoh 2 — Login Gagal > 10 Kali

Konsep:

```
index: wazuh-alerts-*
filter: rule.description mengandung
failed/password/authentication
range: 5 menit terakhir
aggregation: source IP / agent
threshold: count > 10
```

Pertanyaan:

- apakah IP sama?
- user apa yang dicoba?
- target mana?
- apakah ada login sukses setelahnya?

Contoh 3 — Perubahan /etc/passwd

Query sederhana:

```
{
  "query": {
    "match_phrase": {
      "syscheck.path": "/etc/passwd"
    }
  }
}
```

Tindakan:

- cek apakah perubahan sah;
- cari user baru;
- cek /var/log/auth.log;
- simpan FIM evidence;
- eskalasi jika tidak sah.

Contoh 4 — Port Scan Terdeteksi

Jika alert Suricata masuk index:

```
{
  "query": {
    "match_phrase": {
      "alert.signature": "scan"
    }
  }
}
```

Atau gunakan field kategori sesuai rule.
Tindakan:

- cek Suricata alert;
- pivot ke Zeek conn.log;
- cek endpoint target;
- ambil PCAP jika perlu.

Contoh 5 — Koneksi ke IOC

Konsep:

```
source/destination IP cocok dengan daftar IOC
```

Tahapan:

- siapkan daftar IOC;
- query field IP;
- jika cocok, buat alert;
- enrichment dengan MISP;
- buat case.

Template Catatan Alert Query

```
nano ~/soc-lab/docs/template-alert-query.txt
```

Isi:

```
# Template Alert Berbasis Query

Nama alert:
Tujuan:
Index:
Field utama:
Query:
Threshold:
Time window:
Severity:
False positive umum:
Tindakan awal:
Evidence yang disimpan:
Approval diperlukan:
Catatan tuning:
```

Jangan Terlalu Agresif

Alert berbasis query harus diuji.
Siklus tuning:

```
buat query
→ uji di data lama
→ cek jumlah hasil
→ cek false positive
→ atur threshold
→ aktifkan monitor
→ review mingguan
```

Alert yang tidak di tuning akan berubah menjadi noise.

7.9 Hands-on Python: Query OpenSearch dari Script

Python membantu membuat query berulang.
Script ini memakai library standar Python agar sederhana.

Buat file:

```
nano ~/soc-lab/scripts/query_opensearch_high_alert.py
```

Isi:

```
import base64
import json
import ssl
import urllib.request
from pathlib import Path

OPENSEARCH_URL = "https://localhost:9200"
USERNAME = "admin"
PASSWORD = "GANTI_PASSWORD"
INDEX = "wazuh-alerts-*"

query = {
    "size": 20,
    "query": {
        "range": {
            "rule.level": {
                "gte": 10
            }
        }
    },
    "sort": [
        {
            "timestamp": {
                "order": "desc"
            }
        }
    ]
}

auth_text = f"{USERNAME}:{PASSWORD}"
auth_header = base64.b64encode(auth_text.encode()).decode()

request = urllib.request.Request(
    f"{OPENSEARCH_URL}/{INDEX}/_search?pretty",
    data=json.dumps(query).encode(),
    method="GET",
    headers={
        "Content-Type": "application/json",
        "Authorization": f"Basic {auth_header}",
    },
)

ctx = ssl._create_unverified_context()
```

```
with urllib.request.urlopen(request, context=ctx) as response:
    result = json.loads(response.read().decode())

output_dir = Path.home() / "soc-lab" / "evidence" / "opensearch"
output_dir.mkdir(parents=True, exist_ok=True)

output_file = output_dir / "python-high-alerts.json"
output_file.write_text(json.dumps(result, indent=2),
encoding="utf-8")

hits = result.get("hits", {}).get("hits", [])

print(f"Jumlah alert ditemukan: {len(hits)}")

for hit in hits:
    source = hit.get("_source", {})
    print(
        source.get("timestamp", "-"),
        source.get("agent", {}).get("name", "-"),
        source.get("rule", {}).get("level", "-"),
        source.get("rule", {}).get("description", "-"),
    )

print(f"Output disimpan ke: {output_file}")
```

Jalankan:

```
python3 ~/soc-lab/scripts/query_opensearch_high_alert.py
```

Catatan Keamanan Script

Untuk lab, password boleh ditulis sementara agar mudah belajar.

Untuk penggunaan serius:

- jangan hardcode password;
- gunakan file konfigurasi dengan permission ketat;
- gunakan environment variable;
- batasi akses script;
- validasi sertifikat TLS.

Contoh permission:

```
chmod 700 ~/soc-lab/scripts
chmod 600
~/soc-lab/scripts/query_opensearch_high_alert.py
```

Script SOC harus aman, karena script sering memegang kredensial.

7.10 Troubleshooting OpenSearch dan Dashboard

Masalah 1 — Service Tidak Aktif

Cek:

```
sudo systemctl status wazuh-indexer
sudo journalctl -u wazuh-indexer -n 100 --no-pager
```

Jika OpenSearch mandiri:

```
sudo systemctl status opensearch
sudo journalctl -u opensearch -n 100 --no-pager
```

Masalah 2 — Port 9200 Tidak Terbuka

Cek:

```
sudo ss -tulpn | grep 9200
```

Jika kosong:

- service belum aktif;
- bind address salah;
- konfigurasi gagal;
- resource tidak cukup.

Masalah 3 — Autentikasi Gagal

Cek:

- username;
- password;
- karakter khusus pada password;
- apakah perlu kutip;
- apakah akun terkunci;
- apakah memakai sertifikat atau mode keamanan tertentu.

Contoh:

```
curl -k -u 'admin:<password>' https://localhost:9200
```

Masalah 4 — Index Tidak Muncul

Cek:

```
curl -k -u admin:<password>
https://localhost:9200/_cat/indices?v
```

Jika wazuh-alerts-* belum muncul:

- Wazuh belum menghasilkan alert;
- pipeline belum mengirim data;
- indexer belum sehat;
- waktu salah;
- agent belum aktif.

Cek Wazuh:

```
sudo tail -f /var/ossec/logs/alerts/alerts.json
```

Masalah 5 — Dashboard Kosong

Penyebab umum:

- time range salah;
- index pattern belum benar;
- belum ada data;
- field mapping tidak cocok;
- dashboard belum import;
- user tidak punya akses.

Cek dengan API dulu:

```
curl -k -u admin:<password> \
"https://localhost:9200/wazuh-alerts-*/_search?size=1&pr
etty"
```

Jika API ada data, masalah mungkin di dashboard.
Jika API kosong, masalah ada di pipeline data.

Masalah 6 — Query Lambat

Penyebab:

- storage lambat;
- RAM kurang;
- index terlalu besar;
- query terlalu lebar;
- tidak membatasi waktu;
- terlalu banyak wildcard;
- shard bermasalah.

Solusi awal:

- batasi rentang waktu;
- batasi `size`;
- gunakan filter field spesifik;
- cek cluster health;
- cek disk;
- cek heap/memori;
- cek index lifecycle.

Wazuh mendokumentasikan Index State Management untuk membantu mengontrol siklus hidup index seperti rollover dan deletion agar performa indexer lebih terjaga. (Dokumentasi Wazuh)

7.11 Checklist Validasi Akhir Bab 7

Service

- Indexer aktif.
- Dashboard aktif.
- Port `9200` aktif.
- Port dashboard aktif.
- Cluster health dapat dibaca.
- Log service dicek.

Index

- `_cat/indices` berhasil.
- `wazuh-alerts-*` muncul.
- Mapping dapat dibaca.
- Satu dokumen alert dapat ditampilkan.
- Field penting dikenali.

Query

- Query alert terbaru berhasil.
- Query level tinggi berhasil.
- Query 1 jam terakhir berhasil.
- Output query dapat difilter dengan `jq`.
- Evidence query tersimpan.

Dashboard

- Index pattern benar.
- Filter waktu benar.
- Panel alert per severity dibuat.
- Panel top agent dibuat.
- Panel top source IP dibuat.
- Panel failed login dibuat.
- Panel FIM dibuat.
- Panel Suricata dibuat jika data tersedia.
- Panel timeline incident dibuat.

Alert Berbasis Query

- Rule level >= 10 dirancang.
- Login gagal > 10 kali dirancang.
- Perubahan `/etc/passwd` dirancang.
- Port scan dirancang.
- Koneksi ke IOC dirancang.
- Template alert query dibuat.
- False positive dicatat.

Dokumentasi

- Catatan OpenSearch dibuat.
- Query disimpan di `~/soc-lab/queries`.
- Evidence disimpan di `~/soc-lab/evidence/opensearch`.
- Script Python query dibuat.
- Dashboard minimum terdokumentasi.

7.12 Kesalahan Umum Bab 7

1. Dashboard Dibuat Sebelum Field Dipahami

Akibat:

- panel kosong;
- query salah;
- visualisasi menyesatkan.

Solusi:

```
curl -k -u admin:<password> \
"https://localhost:9200/wazuh-alerts-*/_search?size=1&pr
etty"
```

Baca struktur field dulu.

2. Query Tidak Membatasi Waktu

Query terlalu lebar membuat sistem lambat.
Gunakan:

```
"range": {
  "timestamp": {
    "gte": "now-1h",
    "lte": "now"
  }
}
```

3. Membuka Port Indexer ke Jaringan Luas

Port 9200 sangat sensitif.
Batasi akses.
Gunakan firewall.
Jangan ekspos sembarangan.

4. Semua Alert Dianggap Penting

Jika semua alert penting, tidak ada yang benar-benar penting.
Gunakan severity, threshold, konteks, dan tuning.

5. Tidak Menyimpan Query

Query yang tidak disimpan sulit diuji ulang.
Simpan di:

```
~/soc-lab/queries/
```

6. Dashboard Terlalu Ramai

> Dashboard yang terlalu ramai membuat analis lambat.
> Mulai dari 7 panel minimum.
> Tambah panel jika ada kebutuhan nyata.

7.13 Rangkuman Bab

Bab ini mengubah log dari file mentah menjadi data yang bisa dicari, divisualisasikan, dan dijadikan alert.

Pembaca belajar:

- mengapa log perlu diindeks;
- cara memahami OpenSearch / Wazuh Indexer;
- cara memeriksa service;
- cara membaca index;
- cara memahami field dan mapping;
- cara query dengan `curl`;
- cara mencari alert severity tinggi;
- cara membuat dashboard minimum SOC;
- cara merancang alert berbasis query;
- cara membuat script Python untuk query OpenSearch.

Inti Bab 7:

> **Index mempercepat pencarian. Dashboard mempercepat visualisasi.**
> **Query mempercepat pembuktian. Alert mempercepat respons.**

Namun semuanya harus tetap dikontrol oleh analis.
Dashboard bukan pengganti log.
Alert bukan pengganti triage.
Query bukan pengganti konteks.

Output Bab

Setelah menyelesaikan Bab 7, pembaca memiliki:

- OpenSearch / Wazuh Indexer aktif;
- dashboard aktif;
- index `wazuh-alerts-*` dapat dibaca;
- cluster health dapat dicek;
- query dasar berjalan dengan `curl`;
- alert level tinggi dapat dicari;
- query tersimpan di folder kerja;
- evidence query tersimpan;
- dashboard minimum SOC dirancang;
- alert berbasis query siap diuji;
- script Python sederhana untuk mengambil alert dari OpenSearch.

Kalimat kunci Bab 7:

> **SOC yang matang bukan hanya menyimpan log, tetapi membuat log dapat dicari cepat, divisualisasikan tepat, dan dijadikan alert yang tajam.**

Bab 8 — Deteksi, Rule, Decoder, FIM, Vulnerability, dan Active Response

Bab 8 yang menempatkan deteksi, rule, decoder, FIM, vulnerability detection, YARA, active response, simulasi SSH brute force di lab, dan export alert sebagai fokus utama. Bab ini juga selaras dengan dokumen **Open Source SOC Mastery** yang menegaskan bahwa SOC modern bukan sekadar memasang tools, tetapi membangun disiplin membaca sinyal, membuktikan serangan, menyimpan evidence, dan merespons dengan etis.

Secara teknis, Wazuh mendukung **custom rules** untuk mendefinisikan kondisi deteksi sesuai lingkungan organisasi; rule lokal umum disimpan di `/var/ossec/etc/rules/local_rules.xml`. Wazuh juga menyediakan `wazuh-logtest` untuk menguji decoder dan rule terhadap contoh log sebelum rule dipakai pada sistem nyata. (Dokumentasi Wazuh) Untuk decoder, dokumentasi Wazuh menyarankan decoder lokal di `/var/ossec/etc/decoders/local_decoder.xml` atau file baru di direktori decoder lokal untuk skala lebih besar. (Dokumentasi Wazuh) FIM Wazuh memantau pembuatan, perubahan, dan penghapusan file/direktori, sedangkan vulnerability detection bekerja dengan Syscollector untuk mencocokkan inventaris software endpoint dengan informasi CVE. (Dokumentasi Wazuh)

8.1 Deteksi Adalah Inti SOC

SOC yang matang bukan hanya memasang tools.
SOC yang matang mampu membuat deteksi yang sesuai konteks organisasi.

Deteksi adalah jantung SOC.

Tanpa deteksi yang baik:

- log hanya menjadi arsip;
- dashboard hanya menjadi tampilan;
- alert menjadi noise;
- analis mudah lelah;
- insiden terlambat ditemukan.

Dari Log ke Deteksi

Alur deteksi:

```
log mentah
→ decoder
→ field
→ rule
→ alert
→ triage
→ evidence
→ response
```

Maknanya:

- **log mentah** adalah catatan asli;
- **decoder** mengekstrak field;
- **rule** memberi logika deteksi;
- **alert** adalah sinyal;
- **triage** memutuskan apakah alert penting;
- **evidence** menyimpan bukti;
- **response** dilakukan setelah bukti cukup.

Deteksi Bawaan vs Deteksi Lokal

Wazuh sudah membawa banyak rule bawaan.
Namun organisasi punya konteks masing-masing.
Contoh konteks lokal:

- server admin hanya boleh diakses dari subnet tertentu;
- perubahan `/etc/sudoers` harus dianggap kritikal;
- login gagal dari jaringan internal tertentu harus diperiksa;
- service tertentu tidak boleh mati;
- direktori web tidak boleh berubah di luar jam maintenance;
- koneksi ke IOC tertentu harus menjadi prioritas.

Rule bawaan memberi fondasi. Rule lokal memberi ketajaman.

Deteksi yang Baik

Deteksi yang baik harus:

- jelas tujuannya;
- berbasis log yang nyata;
- bisa diuji;
- tidak terlalu bising;
- punya level yang masuk akal;
- punya deskripsi yang mudah dipahami;
- menghasilkan evidence;
- bisa dituning;
- bisa dijelaskan dalam laporan.

Deteksi buruk:

```
semua login gagal = critical
```

Deteksi lebih baik:

```
login gagal berulang dalam timeframe pendek dari sumber
yang sama
```

Kesalahan Umum

- membuat rule tanpa memahami log;
- menaikkan level semua alert;
- tidak menguji dengan `wazuh-logtest`;
- tidak mencatat false positive;
- tidak punya rencana response;
- tidak menyimpan evidence.

Deteksi bukan soal membuat alert sebanyak mungkin. Deteksi adalah memilih sinyal yang benar.

8.2 Wazuh Rule dan Decoder

Lokasi utama:

```
/var/ossec/etc/rules/local_rules.xml
/var/ossec/etc/decoders/local_decoder.xml
```

Rule

Rule menentukan kapan event menjadi alert.

Rule bisa memakai:

- `id;`
- `level;`
- `decoded_as;`
- `if_sid;`
- `match;`
- `regex;`
- `field;`
- `description;`
- `group;`
- `frequency;`
- `timeframe.`

Contoh konsep rule:

```
<rule id="100100" level="10">
  <if_sid>5716</if_sid>
  <description>Login gagal berulang terdeteksi</description>
  <group>authentication_failed,ssh,</group>
</rule>
```

Decoder

Decoder membaca log dan mengambil field.
Contoh log aplikasi lokal:

```
SOCAPP: user=admin srcip=192.168.56.20 action=login_failed
```

Decoder membantu mengambil:

- user;
- srcip;
- action.

Contoh decoder sederhana:

```
<decoder name="socapp">
  <prematch>^SOCAPP:</prematch>
</decoder>

<decoder name="socapp-fields">
  <parent>socapp</parent>
  <regex>user=(\S+) srcip=(\S+) action=(\S+)</regex>
  <order>user,srcip,action</order>
</decoder>
```

Backup Sebelum Edit

Di Wazuh server:

```
sudo cp -a /var/ossec/etc/rules/local_rules.xml \
/var/ossec/etc/rules/local_rules.xml.backup.$(date
+%F-%H%M)

sudo cp -a /var/ossec/etc/decoders/local_decoder.xml \
/var/ossec/etc/decoders/local_decoder.xml.backup.$(date
+%F-%H%M)
```

Edit Rule Lokal

```
sudo nano /var/ossec/etc/rules/local_rules.xml
```

Edit Decoder Lokal

```
sudo nano /var/ossec/etc/decoders/local_decoder.xml
```

Cek Error Setelah Edit

Restart manager:

```
sudo systemctl restart wazuh-manager
```

Cek status:

```
sudo systemctl status wazuh-manager
```

Cek log:

```
sudo tail -n 100 /var/ossec/logs/ossec.log
```

Cari error:

```
sudo grep -iE "error|critical|fatal|invalid|xml"
/var/ossec/logs/ossec.log
```

Rule dan decoder salah bisa membuat engine analisis bermasalah. Backup dulu, edit sedikit, uji cepat.

8.3 Menguji Rule dengan Logtest

Jangan menguji rule langsung di produksi.

Gunakan:

```
sudo /var/ossec/bin/wazuh-logtest
```

`wazuh-logtest` memungkinkan pengujian decoder dan rule terhadap contoh log di Wazuh server, sehingga analis bisa melihat decoder apa yang match, field apa yang terbaca, dan rule apa yang memicu alert. (Dokumentasi Wazuh)

Menjalankan Logtest

```
sudo /var/ossec/bin/wazuh-logtest
```

Masukkan contoh log:

```
SOCAPP: user=admin srcip=192.168.56.20
action=login_failed
```

Output yang dicari:

- decoder match;
- field terbaca;
- rule match;
- level alert;
- deskripsi rule.

Contoh Rule untuk Log Aplikasi Lokal

Edit decoder:

```
sudo nano /var/ossec/etc/decoders/local_decoder.xml
```

Tambahkan:

```
<decoder name="socapp">
  <prematch>^SOCAPP:</prematch>
</decoder>

<decoder name="socapp-fields">
  <parent>socapp</parent>
  <regex>user=(\S+) srcip=(\S+) action=(\S+)</regex>
  <order>user,srcip,action</order>
</decoder>
```

Edit rule:

```
sudo nano /var/ossec/etc/rules/local_rules.xml
```

Tambahkan sebelum penutup </group> jika file memakai struktur group lokal:

```
<group name="socapp,">
  <rule id="100200" level="7">
    <decoded_as>socapp</decoded_as>
    <field name="action">login_failed</field>
    <description>SOCAPP login gagal dari $(srcip) untuk
user $(user)</description>
    <group>authentication_failed,</group>
  </rule>
</group>
```

Restart:

```
sudo systemctl restart wazuh-manager
```

Uji:

```
sudo /var/ossec/bin/wazuh-logtest
```

Input:

```
SOCAPP: user=admin srcip=192.168.56.20
action=login_failed
```

Catatan Penting ID Rule

Gunakan ID lokal yang tidak bertabrakan.
Praktik aman:

```
100000–120000 untuk rule lokal lab
```

Catat setiap rule:

```
nano ~/soc-lab/docs/daftar-rule-lokal.txt
```

Isi:

```
# Daftar Rule Lokal

+---------+--------------------+-------+-------------------+--------+
| Rule ID | Nama               | Level | Tujuan            | Status |
+---------+--------------------+-------+-------------------+--------+
| 100200  | SOCAPP login gagal | 7     | Uji decoder lokal | aktif  |
+---------+--------------------+-------+-------------------+--------+
```

Rule yang tidak dicatat akan menjadi utang teknis SOC.

8.4 Custom Rule Login Gagal

Materi:

- rule ID lokal;
- level;
- group;
- description;
- frequency;
- timeframe.

Tujuan praktik ini adalah membuat rule lokal yang mendeteksi login gagal berulang. Untuk keamanan dan etika, simulasi dilakukan hanya di endpoint milik sendiri atau dengan contoh log melalui `wazuh-logtest`.

Konsep Rule Login Gagal

Kita tidak ingin alert kritikal untuk satu kegagalan login.
Yang dicari adalah pola:
banyak login gagal
dalam waktu pendek
dari sumber yang sama atau target yang sama

Rule Dasar Berbasis Rule Bawaan SSH

Wazuh memiliki rule bawaan untuk SSH. Salah satu pendekatan aman adalah membuat rule turunan dari rule bawaan yang sudah mengenali SSH gagal.

Contoh konsep:

```
<group name="local,ssh,authentication_failed,">
  <rule id="100300" level="10" frequency="5"
timeframe="120">
    <if_matched_sid>5716</if_matched_sid>
    <description>SSH login gagal berulang dalam 120
detik</description>
    <group>authentication_failed,ssh,bruteforce,</group>
  </rule>
</group>
```

Catatan:

- ID `5716` adalah contoh rule bawaan yang sering terkait autentikasi SSH gagal, tetapi **validasi di instalasi masing-masing dengan logtest**;
- `frequency="5"` berarti 5 kejadian;
- `timeframe="120"` berarti 120 detik;
- level `10` membuatnya prioritas tinggi.

Edit Rule

```
sudo nano /var/ossec/etc/rules/local_rules.xml
```

Tambahkan:

```
<group name="local,ssh,authentication_failed,">
  <rule id="100300" level="10" frequency="5" timeframe="120">
    <if_matched_sid>5716</if_matched_sid>
    <description>SSH login gagal berulang dalam 120
detik</description>
    <group>authentication_failed,ssh,bruteforce,</group>
  </rule>
</group>
```

Restart manager:

```
sudo systemctl restart wazuh-manager
```

Cek error:

```
sudo tail -n 100 /var/ossec/logs/ossec.log
```

Uji dengan Logtest

```
sudo /var/ossec/bin/wazuh-logtest
```

Masukkan contoh log SSH gagal beberapa kali:

```
Jun  2 10:01:01 endpoint-01 sshd[1234]: Failed password
for invalid user admin from 192.168.56.20 port 53322
ssh2
```

Ulangi beberapa kali untuk melihat rule frequency bekerja.

Simpan Evidence

```
mkdir -p ~/soc-lab/evidence/rules
sudo cp -a /var/ossec/etc/rules/local_rules.xml \
~/soc-lab/evidence/rules/local_rules.xml.after-ssh-rule
```

Hash:

```
sha256sum
~/soc-lab/evidence/rules/local_rules.xml.after-ssh-rule
\
| tee ~/soc-lab/evidence/rules/local_rules.sha256
```

Catatan Tuning

Jika terlalu bising:

- naikkan `frequency`;
- pendekkan atau panjangkan `timeframe`;
- batasi source IP;
- turunkan level;
- tambahkan group khusus;
- buat whitelist untuk scanner internal.

Rule yang baik lahir dari tuning, bukan dari copy-paste.

8.5 FIM untuk File Penting

Pantau:

- `/etc/passwd`
- `/etc/shadow`
- `/etc/sudoers`
- `/etc/ssh/sshd_config`
- `/var/www/html`

Wazuh FIM memantau pembuatan, modifikasi, dan penghapusan file/direktori, lalu membandingkan checksum dan atribut terhadap baseline untuk memicu alert jika ada perubahan. (Dokumentasi Wazuh)

Mengapa File Ini Penting?

File / Direktori	Risiko
`/etc/passwd`	user baru atau perubahan akun
`/etc/shadow`	perubahan hash password
`/etc/sudoers`	privilege escalation
`/etc/ssh/sshd_config`	SSH dilemahkan
`/var/www/html`	perubahan halaman, web shell, defacement

Backup Konfigurasi Agent

Di endpoint:

```
sudo cp -a /var/ossec/etc/ossec.conf \
/var/ossec/etc/ossec.conf.before-fim.$(date +%F-%H%M)
```

Edit Konfigurasi FIM

```
sudo nano /var/ossec/etc/ossec.conf
```

Cari blok:

```
<syscheck>
```

Tambahkan:

```
<directories check_all="yes" realtime="yes">/etc/passwd</directories>
<directories check_all="yes" realtime="yes">/etc/shadow</directories>
<directories check_all="yes" realtime="yes">/etc/sudoers</directories>
<directories check_all="yes"
realtime="yes">/etc/ssh/sshd_config</directories>
<directories check_all="yes" realtime="yes">/var/www/html</directories>
```

Restart agent:

```
sudo systemctl restart wazuh-agent
```

Cek status:

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

Uji Aman dengan File Dummy

Jangan mengubah file kritikal tanpa rencana rollback.
Gunakan file uji:

```
sudo touch /etc/test-soc-fim.conf
sudo nano /etc/test-soc-fim.conf
sudo rm /etc/test-soc-fim.conf
```

Di Wazuh server:

```
sudo tail -f /var/ossec/logs/alerts/alerts.json | jq .
```

Cari alert FIM:

```
sudo jq 'select(.rule.groups[]? == "syscheck")' \
/var/ossec/logs/alerts/alerts.json
```

Cari file uji:

```
sudo jq 'select(.syscheck.path? | test("test-soc-fim.conf"))' \
/var/ossec/logs/alerts/alerts.json
```

Simpan Evidence FIM

```
mkdir -p ~/soc-lab/evidence/fim
sudo jq 'select(.syscheck.path? | test("test-soc-fim.conf"))'
\
/var/ossec/logs/alerts/alerts.json \
> ~/soc-lab/evidence/fim/test-soc-fim-alert.json
```

Catatan FIM

```
nano ~/soc-lab/evidence/fim/catatan-fim.txt
```

Isi:

```
# Catatan Uji FIM

Tanggal:
Endpoint:
IP endpoint:
File diuji:
/etc/test-soc-fim.conf

Aksi:
- dibuat
- diubah
- dihapus

Alert:
Rule ID:
Level:
Path:
Action:
Waktu:
Kesimpulan:
```

FIM yang baik harus fokus pada file penting. Jangan memantau semua file tanpa strategi.

8.6 Vulnerability Detection

Materi:

- inventaris paket;
- CVE;
- severity;
- prioritas patch;
- false positive.

Wazuh vulnerability detection bekerja dengan Syscollector: agent mengumpulkan inventaris software, lalu Wazuh mencocokkannya dengan vulnerability intelligence untuk menemukan CVE yang relevan. (Dokumentasi Wazuh) Modul ini menghasilkan alert untuk kerentanan pada sistem operasi dan aplikasi di endpoint yang dipantau. (Dokumentasi Wazuh)

Konsep Dasar

```
agent inventory
→ daftar paket
→ versi paket
→ database vulnerability
→ CVE match
→ severity
→ alert
→ prioritas patch
```

Inventaris Paket

Data penting:

- nama paket;
- versi;
- arsitektur;
- vendor/repository;
- endpoint;
- waktu scan.

Di endpoint Linux:

```
dpkg -l | less
```

Cari paket tertentu:

```
dpkg -l | grep openssh
```

CVE

CVE adalah identitas kerentanan.
Contoh format:

```
CVE-2026-XXXX
```

Namun jangan menganggap semua CVE sama kritikal.
Perhatikan:

- severity;
- exploitability;
- exposure;
- versi paket;
- apakah paket benar terpasang;
- apakah service aktif;
- apakah aset kritikal;
- apakah ada mitigasi.

Severity

Severity membantu prioritas.
Namun severity bukan satu-satunya faktor.
Gunakan formula sederhana:

```
Prioritas patch = severity + exposure + criticality aset +
exploitability
```

Contoh:

CVE	Severity	Service Aktif	Aset	Prioritas
CVE A	Critical	Ya	Internet-facing	Sangat tinggi
CVE B	High	Tidak	Lab internal	Sedang
CVE C	Medium	Ya	Endpoint admin	Tinggi
CVE D	Low	Tidak	Non-kritikal	Rendah

False Positive

Vulnerability alert bisa salah konteks.

Contoh:

- paket terdeteksi rentan, tetapi service tidak aktif;
- versi paket sudah dipatch oleh distro dengan backport;
- CVE tidak berlaku pada konfigurasi lokal;
- aplikasi tidak terekspos jaringan;
- data inventory belum update.

Jangan panik karena CVE. Validasi konteks.

Query Vulnerability dari Alert

Di Wazuh server:

```
sudo jq 'select(.rule.groups[]? | test("vulnerability"; "i"))' \
/var/ossec/logs/alerts/alerts.json
```

Simpan:

```
mkdir -p ~/soc-lab/evidence/vulnerability
sudo jq 'select(.rule.groups[]? | test("vulnerability"; "i"))' \
/var/ossec/logs/alerts/alerts.json \
> ~/soc-lab/evidence/vulnerability/vulnerability-alerts.json
```

Template Prioritas Patch

```
nano ~/soc-lab/docs/template-prioritas-patch.txt
```

Isi:

```
# Template Prioritas Patch

Tanggal:
Analis:
Endpoint:
IP:
Paket:
Versi:
CVE:
Severity:
Service aktif:
Aset kritikal:
Terekspos jaringan:
Exploit tersedia:
False positive:
Prioritas:
Rekomendasi:
Status:
```

Vulnerability detection bukan akhir pekerjaan. Ia adalah awal proses validasi dan patch management.

8.7 Deteksi Malware Sederhana dengan YARA

Aman hanya memakai dummy file, bukan malware asli.

YARA adalah tool untuk membantu peneliti malware mengidentifikasi dan mengklasifikasikan sampel berdasarkan pola tekstual atau biner yang ditulis sebagai rule. (YARA) Wazuh juga mendokumentasikan integrasi FIM dan YARA: FIM mendeteksi file baru/berubah, lalu pemindaian YARA dapat dipakai untuk memeriksa file tersebut. (Dokumentasi Wazuh)

Prinsip Aman

Jangan memakai malware asli.
Gunakan dummy string:

```
SOC_TEST_MALWARE_STRING
```

Tujuan:

- belajar rule YARA;
- belajar scanning file;
- belajar evidence;
- tidak membahayakan sistem.

Instal YARA

Di endpoint lab:

```
sudo apt update
sudo apt install -y yara
```

Cek:

```
yara --version
```

Buat Rule YARA Dummy

```
mkdir -p ~/soc-lab/yara
nano ~/soc-lab/yara/soc_dummy_rule.yar
```

Isi:

```
rule SOC_Dummy_Test_File
{
    meta:
        description = "Dummy rule untuk latihan SOC,
bukan malware asli"
        author = "SOC Lab"
        severity = "low"

    strings:
        $test_string = "SOC_TEST_MALWARE_STRING"

    condition:
        $test_string
}
```

Buat File Dummy

```
mkdir -p ~/soc-lab/malware-test
echo "ini file dummy SOC_TEST_MALWARE_STRING untuk latihan" \
> ~/soc-lab/malware-test/dummy.txt
```

Jalankan Scan YARA

```
yara ~/soc-lab/yara/soc_dummy_rule.yar
~/soc-lab/malware-test/dummy.txt
```

Output yang diharapkan:

```
SOC_Dummy_Test_File
/home/user/soc-lab/malware-test/dummy.txt
```

Simpan Evidence

```
mkdir -p ~/soc-lab/evidence/yara
yara ~/soc-lab/yara/soc_dummy_rule.yar
~/soc-lab/malware-test/dummy.txt \
| tee ~/soc-lab/evidence/yara/yara-dummy-result.txt
```

Hash file:

```
sha256sum ~/soc-lab/malware-test/dummy.txt \
| tee ~/soc-lab/evidence/yara/dummy-file.sha256
```

Catatan Penting

YARA bukan antivirus lengkap.

YARA adalah rule-based matching.

Ia berguna untuk:

- hunting file tertentu;
- mendeteksi pola string;
- klasifikasi sampel;
- validasi IOC file;
- latihan malware detection secara aman.

Dalam buku ini, YARA dipakai untuk belajar deteksi, bukan bermain malware.

8.8 Active Response

Prinsip:

- jangan aktifkan respons otomatis agresif di produksi;
- uji di lab;
- catat semua tindakan;
- respons harus bisa diaudit.

Wazuh Active Response memungkinkan tindakan otomatis berdasarkan trigger seperti rule ID, level alert, atau group rule tertentu. Script active response dapat dibuat dalam berbagai bahasa, tetapi harus dipanggil melalui command dan konfigurasi active response yang jelas. (Dokumentasi Wazuh)

Mengapa Harus Hati-Hati?

Active response bisa melakukan tindakan seperti:

- memblokir IP;
- menjalankan script;
- menulis log;
- mengisolasi proses;
- menghapus file;
- mengubah konfigurasi.

Jika salah trigger, dampaknya serius.

Contoh risiko:

- IP admin terblokir;
- service penting terganggu;
- file evidence terhapus;
- false positive berubah menjadi outage;
- tindakan tidak bisa diaudit.

Active response harus dikendalikan, bukan dilepas bebas.

Desain Aman untuk Lab

Gunakan respons pasif dulu:

```
alert masuk
→ active response menulis log
→ tidak memblokir
→ tidak menghapus
→ tidak mematikan service
→ evidence tersimpan
```

Contoh Script Active Response Aman

Di agent atau server lab:

```
sudo nano /var/ossec/active-response/bin/soc_log_only.sh
```

Isi:

```
#!/bin/bash

LOG_FILE="/tmp/soc-active-response.log"

echo "$(date -Is) Active response dipanggil: $0 $*" >>
"$LOG_FILE"

exit 0
```

Permission:

```
sudo chmod 750
/var/ossec/active-response/bin/soc_log_only.sh
sudo chown root:wazuh
/var/ossec/active-response/bin/soc_log_only.sh
```

Catatan:

- script hanya menulis log;
- tidak memblokir;
- tidak menghapus;
- aman untuk latihan awal.

Cek Log Active Response

```
cat /tmp/soc-active-response.log
```

Prinsip Konfigurasi

Active response harus punya:

- command;
- trigger;
- lokasi eksekusi;
- timeout jika relevan;
- log tindakan;
- rollback;
- approval untuk tindakan agresif.

Dokumentasi Wazuh menjelaskan bahwa konfigurasi `<active-response>` mengikat command dengan rule/rule type dan kriteria kapan command dijalankan. (Dokumentasi Wazuh)

Catatan Audit

Buat template:

```
nano ~/soc-lab/docs/template-active-response-audit.txt
```

Isi:

```
# Audit Active Response

Tanggal:
Rule ID:
Alert:
Endpoint:
Command:
Tindakan:
Mode:
[ ] log only
[ ] block
[ ] isolate
[ ] remove
Approval:
Evidence:
Rollback:
Hasil:
Catatan risiko:
```

Active response yang tidak tercatat adalah risiko baru.

8.9 Simulasi SSH Brute Force di Lab

Hanya terhadap endpoint sendiri.

Bab ini tidak memakai tool serangan otomatis. Tujuan latihan adalah menghasilkan beberapa log login gagal secara terkendali agar rule dan alert bisa diuji.

Prasyarat

- endpoint milik sendiri;
- SSH aktif di endpoint;
- Wazuh Agent aktif;
- rule login gagal sudah diuji;
- lab terisolasi;
- tidak memakai akun orang lain;
- tidak memakai target publik.

Cara Aman: Login Gagal Manual

Dari mesin lab:

```
ssh user_salah@<IP_ENDPOINT>
```

Contoh:

```
ssh user_salah@192.168.56.20
```

Masukkan password salah.
Ulangi beberapa kali sesuai threshold rule.

Cek Log Endpoint

Di endpoint:

```
sudo tail -f /var/log/auth.log
```

Cari:

```
sudo grep "Failed password" /var/log/auth.log
```

Cek Alert Wazuh

Di Wazuh server:

```
sudo tail -f /var/ossec/logs/alerts/alerts.json | jq .
```

Filter:

```
sudo jq 'select(.rule.description |
test("SSH|password|authentication|login"; "i"))' \
/var/ossec/logs/alerts/alerts.json
```

Simpan Evidence

```
mkdir -p ~/soc-lab/evidence/ssh-bruteforce
sudo jq 'select(.rule.description |
test("SSH|password|authentication|login"; "i"))' \
/var/ossec/logs/alerts/alerts.json \
>
~/soc-lab/evidence/ssh-bruteforce/wazuh-ssh-alerts.json
```

Simpan auth log:

```
sudo grep "Failed password" /var/log/auth.log \
>
~/soc-lab/evidence/ssh-bruteforce/auth-failed-password.l
og
```

Buat Catatan Simulasi

```
nano
~/soc-lab/evidence/ssh-bruteforce/catatan-simulasi.txt
```

Isi:

```
# Simulasi SSH Login Gagal di Lab

Tanggal:
Analis:
Endpoint:
IP endpoint:
IP sumber:
Jumlah percobaan:
Rule ID:
Level:
Alert muncul:
[ ] Ya
[ ] Tidak

Bukti:
- auth-failed-password.log
- wazuh-ssh-alerts.json
```

```
Kesimpulan:
```

Etika

Jangan menjalankan simulasi ini ke:

- server publik;
- jaringan kampus/organisasi tanpa izin;
- server rekan;
- alamat IP acak;
- aset produksi tanpa approval.

Latihan SOC harus aman, legal, dan terkendali.

8.10 Export Alert untuk Laporan

Alert harus bisa diekspor menjadi evidence.
Contoh dari outline:

```
sudo jq 'select(.rule.level >= 10)'
/var/ossec/logs/alerts/alerts.json >
~/soc-lab/evidence/high-alerts.json
```

Siapkan Folder Evidence

```
mkdir -p ~/soc-lab/evidence/reports
```

Export Alert Level Tinggi

```
sudo jq 'select(.rule.level >= 10)' \
/var/ossec/logs/alerts/alerts.json \
> ~/soc-lab/evidence/reports/high-alerts.json
```

Export Alert FIM

```
sudo jq 'select(.rule.groups[]? == "syscheck")' \
/var/ossec/logs/alerts/alerts.json \
> ~/soc-lab/evidence/reports/fim-alerts.json
```

Export Alert SSH

```
sudo jq 'select(.rule.description |
test("SSH|password|authentication|login"; "i"))' \
/var/ossec/logs/alerts/alerts.json \
> ~/soc-lab/evidence/reports/ssh-alerts.json
```

Ringkas Alert dengan Python

Buat script:

```
nano ~/soc-lab/scripts/ringkas_alert_laporan.py
```

Isi:

```
from pathlib import Path
import json
from collections import Counter

file_path = Path.home() / "soc-lab" / "evidence" / "reports" /
"high-alerts.json"

if not file_path.exists():
    raise FileNotFoundError(f"File tidak ditemukan:
{file_path}")

agent_counter = Counter()
rule_counter = Counter()
level_counter = Counter()
```

```
with file_path.open("r", encoding="utf-8", errors="ignore") as
f:
    for line in f:
        try:
            event = json.loads(line)
        except json.JSONDecodeError:
            continue

        agent = event.get("agent", {}).get("name", "-")
        rule = event.get("rule", {}).get("description", "-")
        level = str(event.get("rule", {}).get("level", "-"))

        agent_counter[agent] += 1
        rule_counter[rule] += 1
        level_counter[level] += 1

print("# Ringkasan Alert Level Tinggi")
print()

print("## Top Agent")
for agent, count in agent_counter.most_common(10):
    print(f"- {agent}: {count}")

print()
print("## Top Rule")
for rule, count in rule_counter.most_common(10):
    print(f"- {rule}: {count}")

print()
print("## Level")
for level, count in level_counter.most_common():
    print(f"- Level {level}: {count}")
```

Jalankan:

```
python3 ~/soc-lab/scripts/ringkas_alert_laporan.py \
| tee ~/soc-lab/evidence/reports/ringkasan-high-alerts.md
```

Buat Hash Evidence

```
sha256sum ~/soc-lab/evidence/reports/*.json \
> ~/soc-lab/evidence/reports/SHA256SUMS.txt
```

Template Laporan Singkat

```
nano ~/soc-lab/evidence/reports/laporan-deteksi-bab8.md
```

Isi:

```
# Laporan Deteksi Bab 8

## Ringkasan
Tuliskan ringkasan alert yang diuji.

## Lingkungan
- SOC server:
- Endpoint:
- IP:
- Waktu:

## Deteksi yang Diuji
- Custom rule login gagal
- FIM
- Vulnerability detection
- YARA dummy rule
- Active response log-only

## Evidence
- high-alerts.json
- fim-alerts.json
- ssh-alerts.json
- yara-dummy-result.txt
- SHA256SUMS.txt

## Analisis
Tuliskan apakah alert valid, false positive, atau perlu
tuning.

## Rekomendasi
Tuliskan perbaikan rule, FIM, patch, atau response.

## Lesson Learned
Tuliskan pelajaran teknis.
```

Alert yang tidak bisa diekspor sulit dipertanggungjawabkan.

8.11 Checklist Validasi Akhir Bab 8

Rule dan Decoder

- Backup `local_rules.xml` dibuat.
- Backup `local_decoder.xml` dibuat.
- Rule lokal memiliki ID unik.
- Rule memiliki level jelas.
- Rule memiliki deskripsi jelas.
- Decoder lokal diuji.
- `wazuh-logtest` berhasil.
- Wazuh Manager restart tanpa error.

Custom Login Gagal

- Rule login gagal dibuat.
- `frequency` dipahami.
- `timeframe` dipahami.
- Simulasi dilakukan di lab sendiri.
- Alert muncul.
- Evidence disimpan.

FIM

- File kritikal ditentukan.
- FIM aktif.
- File dummy diuji.
- Alert FIM muncul.
- Evidence FIM disimpan.

Vulnerability Detection

- Inventaris paket dipahami.
- CVE dipahami.
- Severity tidak ditelan mentah-mentah.
- Prioritas patch dibuat.
- False positive dicatat.

YARA

- YARA terpasang.
- Rule dummy dibuat.
- File dummy dibuat.
- Scan berhasil.
- Tidak memakai malware asli.
- Evidence dan hash disimpan.

Active Response

- Prinsip risiko dipahami.
- Mode log-only diuji.
- Tidak memakai respons agresif di produksi.
- Tindakan tercatat.
- Template audit dibuat.

Export Laporan

- High alert diekspor.
- FIM alert diekspor.
- SSH alert diekspor.
- Ringkasan Python dibuat.
- Hash evidence dibuat.
- Laporan singkat dibuat.

8.12 Kesalahan Umum Bab 8

1. Membuat Rule Tanpa Logtest

Akibat:

- rule tidak match;
- decoder salah;
- alert tidak muncul;
- false positive meningkat.

Solusi:

```
sudo /var/ossec/bin/wazuh-logtest
```

2. Mengubah Rule Bawaan

Jangan ubah rule bawaan sembarangan.
Gunakan rule lokal.
Lokasi aman:

```
/var/ossec/etc/rules/local_rules.xml
```

3. Level Alert Terlalu Tinggi

Tidak semua alert harus critical.
Gunakan level sesuai risiko.

4. FIM Terlalu Luas

Memantau seluruh filesystem membuat noise.
Pantau file kritikal.

5. Vulnerability Alert Dianggap Pasti Eksploitable

CVE harus divalidasi.

Cek:

- paket;
- versi;
- exposure;
- service aktif;
- aset;
- mitigasi.

6. Memakai Malware Asli untuk Latihan

Tidak perlu.
Gunakan dummy file dan dummy string.

7. Active Response Terlalu Agresif

Jangan langsung blokir, hapus, atau matikan service.
Mulai dari log-only.

8. Evidence Tidak Disimpan

Jika alert hilang dari layar, laporan lemah.
Gunakan jq, tee, hash, dan catatan.

8.13 Rangkuman Bab

Bab ini mengubah pembaca dari pengguna dashboard menjadi pembuat deteksi.

Pembaca belajar:

- membuat rule lokal;
- memahami decoder;
- menguji rule dengan `wazuh-logtest`;
- membuat custom rule login gagal;
- mengaktifkan FIM untuk file penting;
- memahami vulnerability detection;
- memakai YARA secara aman dengan dummy file;
- memahami active response secara hati-hati;
- mensimulasikan login gagal di lab;
- mengekspor alert untuk laporan;
- membuat ringkasan alert dengan Python.

Inti Bab 8:

> **Deteksi yang baik bukan yang paling ramai, tetapi yang paling relevan, teruji, terdokumentasi, dan bisa dibuktikan.**

SOC yang matang tidak berhenti pada alert bawaan.

SOC yang matang membangun rule sesuai aset, risiko, dan konteks organisasi.

Output Bab

Setelah menyelesaikan Bab 8, pembaca mampu:

- membuat rule sederhana;
- menguji decoder dan rule;
- membaca alert hasil rule;
- menguji FIM;
- memahami vulnerability detection;
- membuat deteksi YARA dummy secara aman;
- memahami risiko active response;
- menjalankan active response log-only di lab;
- melakukan simulasi login gagal di endpoint sendiri;
- mengekspor alert level tinggi untuk laporan;
- membuat ringkasan evidence dengan Python.

Kalimat kunci Bab 8:

> **SOC yang kuat bukan hanya melihat alert; SOC yang kuat mampu membuat deteksi yang tajam, menguji rule sebelum dipakai, menyimpan evidence, dan merespons dengan disiplin.**

Bab 9 — Incident Handling, Case Management, MISP, dan SOAR

Bab 9 yang menekankan bahwa **alert bukan incident**, lalu dilanjutkan dengan triage, case management, MISP, mapping MITRE ATT&CK, SOAR, dan playbook “SSH Brute Force Detected”. Bab ini juga melanjutkan prinsip besar buku: SOC bukan sekadar dashboard, tetapi kemampuan membaca log, menghubungkan bukti, melakukan triage, dan merespons secara etis.

Secara teknis, DFIR-IRIS adalah platform kolaboratif untuk analis incident response agar dapat berbagi investigasi teknis, sedangkan TheHive diposisikan sebagai *collaborative case management platform* untuk tim SOC, CERT, dan CSIRT. (DFIR-IRIS Docs) MISP menyediakan model data dan API untuk mengelola **event**, **object**, dan **attribute**, serta mendukung format JSON untuk pertukaran event dan attribute. (MISP Threat Intelligence Platform) Shuffle menyediakan workflow automation berbasis trigger, node/action, webhook, dan integrasi; Wazuh juga memiliki integrasi ke Shuffle melalui webhook untuk meneruskan alert berdasarkan rule ID, group, atau level alert. (Shuffle)

9.1 Alert Bukan Incident

Alert baru menjadi incident jika ada:

- **bukti**;
- **konteks**;
- **dampak**;
- **keputusan analis**.

Alert adalah sinyal.
Incident adalah kejadian keamanan yang sudah dinilai, dibuktikan, dan memerlukan respons.

> **Jangan menyamakan alert dengan incident.**

Contoh Alert yang Belum Tentu Incident

Alert	Bisa Jadi	Belum Tentu Incident Jika
Login gagal SSH	brute force	admin salah password
Port scan	reconnaissance	vulnerability scanner internal
File berubah	compromise	maintenance terjadwal
Koneksi ke IP mencurigakan	C2 / malware	false positive threat intel
CVE terdeteksi	aset rentan	service tidak aktif / sudah mitigasi

Kapan Alert Naik Menjadi Incident?

Alert naik menjadi incident jika:

- ada bukti dari lebih dari satu sumber;
- ada indikasi dampak;
- ada aset penting terlibat;
- ada perilaku berulang;
- ada pelanggaran kebijakan;
- ada risiko terhadap layanan, data, atau akun;
- analis memutuskan perlu penanganan formal.

Contoh:

> Alert:
> SSH login gagal berulang.
>
> Bukti tambahan:
> - auth.log menunjukkan banyak gagal login.
> - Wazuh alert level tinggi muncul.
> - Suricata melihat banyak koneksi ke port 22.
> - Zeek conn.log menunjukkan koneksi berulang.
> - Source IP bukan scanner internal.
>
> Keputusan:
> Naik menjadi incident: suspected SSH brute force.

Prinsip Kritis

SOC yang buruk:

> alert muncul → panik → blokir

SOC yang baik:

> alert muncul
> → triage
> → cari bukti
> → nilai dampak
> → buat case
> → minta approval jika respons berisiko
> → respons
> → dokumentasi
> → lesson learned

Kecepatan penting, tetapi bukti lebih penting.

9.2 Triage Alert

Triage adalah proses memilah alert.

Tujuannya:

- menentukan prioritas;
- membuang noise;
- mencari bukti tambahan;
- menentukan apakah alert menjadi incident;
- menetapkan tindakan awal.

Checklist Triage

Pertanyaan utama:

- alert apa?
- host mana?
- user siapa?
- source IP?
- waktu kejadian?
- bukti tambahan?
- dampak?
- tindakan awal?

Template Triage Cepat

Alert:
Rule:
Severity:
Waktu:
Host:
User:
Source IP:
Destination IP:
Bukti Wazuh:
Bukti Suricata:
Bukti Zeek:
Bukti auth.log:
Dampak awal:
Keputusan:
Tindakan awal:

Langkah Praktis CLI

Misal IP mencurigakan:

```
192.168.56.20
```

Buat folder:

```
mkdir -p ~/soc-lab/evidence/incident/ssh-bruteforce-001
```

Cari di Wazuh:

```
sudo grep "192.168.56.20" /var/ossec/logs/alerts/alerts.json \
| tee ~/soc-lab/evidence/incident/ssh-bruteforce-001/wazuh-alerts.json
```

Cari di auth.log:

```
sudo grep "192.168.56.20" /var/log/auth.log \
| tee ~/soc-lab/evidence/incident/ssh-bruteforce-001/auth.log
```

Cari di Suricata:

```
sudo grep "192.168.56.20" /var/log/suricata/eve.json \
| tee ~/soc-lab/evidence/incident/ssh-bruteforce-001/suricata.json
```

Cari di Zeek:

```
grep "192.168.56.20" /opt/zeek/logs/current/conn.log \
| tee ~/soc-lab/evidence/incident/ssh-bruteforce-001/zeek-conn.log
```

Hitung Percobaan Login Gagal

```
sudo grep "Failed password" /var/log/auth.log \
| grep "192.168.56.20" \
| wc -l
```

Cari Login Sukses Setelah Gagal

```
sudo grep "Accepted password" /var/log/auth.log \
| grep "192.168.56.20"
```

Jika ada login sukses setelah banyak gagal, prioritas naik.

Keputusan Triage

Gunakan kategori:

Status	Makna
False positive	Tidak ada ancaman
Benign true positive	Aktivitas sah tetapi memicu alert
Suspicious	Perlu monitoring lanjut
Incident	Butuh penanganan formal
Critical incident	Butuh eskalasi cepat

Triage adalah titik di mana analis mengubah alert menjadi keputusan.

9.3 Case Management dengan DFIR-IRIS atau TheHive

Setelah alert dinilai cukup penting, buat case.

Case management mencegah investigasi tercecer.

DFIR-IRIS dirancang sebagai platform kolaboratif untuk berbagi detail teknis investigasi, sedangkan TheHive berfokus pada case management kolaboratif untuk tim keamanan. (DFIR-IRIS Docs)

Isi case:

- judul;
- severity;
- deskripsi;
- timeline;
- observables;
- evidence;
- task;
- analyst note;
- recommendation;
- closure.

Struktur Case Minimum

Judul:
Suspected SSH Brute Force terhadap endpoint-01

Severity:
Medium / High

Deskripsi:
Terdeteksi login gagal berulang dari IP tertentu ke endpoint lab.

Timeline:
10:01 koneksi awal terlihat di Zeek
10:02 Wazuh alert muncul
10:03 auth.log menunjukkan Failed password
10:05 Suricata mencatat koneksi berulang

Observables:
source_ip = 192.168.56.20
destination_ip = 192.168.56.30
destination_port = 22
user = user_salah

Evidence:
wazuh-alerts.json
auth.log
suricata.json
zeek-conn.log

Task:
- validasi user target
- cek login sukses
- cek reputasi IP
- cek endpoint target
- buat rekomendasi response

Analyst note:
Aktivitas berasal dari lab sendiri / belum diketahui.

Recommendation:
Blokir di lab jika bukan aktivitas sah.

Closure:
Closed as lab simulation / confirmed incident / false positive.

Membuat Case Manual di File Teks

Untuk lab kecil, sebelum memakai platform case management, biasakan menulis case manual.

```
nano ~/soc-lab/evidence/incident/ssh-bruteforce-001/case.md
```

Isi:

```
# Case: Suspected SSH Brute Force

## Severity
High

## Deskripsi
Terdeteksi login gagal berulang ke endpoint Linux.

## Timeline
+-------+----------+----------------------+-------------------+
| Waktu | Sumber   | Event                | Bukti             |
+-------+----------+----------------------+-------------------+
| 10:01 | Zeek     | Koneksi ke port 22   | zeek-conn.log     |
| 10:02 | auth.log | Failed password      | auth.log          |
| 10:03 | Wazuh    | Alert authentication | wazuh-alerts.json |
| 10:04 | Suricata | Alert jaringan       | suricata.json     |
+-------+----------+----------------------+-------------------+

## Observables
- Source IP:
- Destination IP:
- Destination port:
- Username:
- Hostname:

## Evidence
- wazuh-alerts.json
- auth.log
- suricata.json
- zeek-conn.log

## Task
- [ ] Cek jumlah percobaan
- [ ] Cek login sukses
- [ ] Cek reputasi IP
- [ ] Cek endpoint
- [ ] Tentukan respons

## Analyst Note
Tuliskan analisis singkat.

## Recommendation
Tuliskan rekomendasi.

## Closure
Status akhir:
```

DFIR-IRIS atau TheHive?

Kriteria	DFIR-IRIS	TheHive
Fokus	Incident response dan investigasi	Case management SOC
Cocok untuk	DFIR, timeline, evidence	SOC/CERT/CSIRT workflow
Kekuatan	Kolaborasi teknis IR	Case, task, observables
Penggunaan lab	Sangat cocok	Sangat cocok

Pilihan praktis:

- gunakan **DFIR-IRIS** jika fokus pada evidence dan investigasi teknis;
- gunakan **TheHive** jika fokus pada case workflow tim SOC.

Untuk buku ini, keduanya dapat dipilih. Minimal, pembaca harus memahami struktur case.

9.4 Threat Intelligence dengan MISP

Threat intelligence memberi konteks pada IOC.

Materi:

- event;
- attribute;
- IOC;
- tag;
- galaxy;
- feed;
- export JSON.

MISP API memungkinkan query, pembuatan, dan modifikasi data model seperti **Events**, **Objects**, dan **Attributes**. MISP core format juga memakai JSON untuk pertukaran event dan attribute. (MISP Threat Intelligence Platform) PyMISP adalah library Python untuk mengakses MISP melalui REST API, termasuk mengambil event, menambah event, dan memperbarui attribute. (CIRCL)

Istilah Penting MISP

Istilah	Makna
Event	Kumpulan informasi tentang kejadian / kampanye / intelijen
Attribute	IOC spesifik seperti IP, domain, URL, hash
IOC	Indicator of Compromise
Tag	Label konteks
Galaxy	Model konteks seperti aktor, malware, teknik
Feed	Sumber intelijen eksternal
Export JSON	Pertukaran data terstruktur

Contoh IOC

Jenis	Contoh
IP	192.0.2.10
Domain	example-malicious.test
URL	http://example.test/login
Hash	SHA256 file
Email	alamat pengirim phishing

Mengapa IOC Perlu Konteks?

IOC tanpa konteks bisa menyesatkan.

Pertanyaan:

- IOC berasal dari mana?
- kapan dibuat?
- masih aktif?
- confidence berapa?
- terkait malware apa?
- terkait teknik apa?
- apakah muncul di log kita?
- apakah ada koneksi nyata?
- apakah false positive?

Template IOC Lokal

```
mkdir -p ~/soc-lab/ioc
nano ~/soc-lab/ioc/ioc-list.json
```

Isi:

```
[
  {
    "type": "ip-src",
    "value": "192.168.56.20",
    "source": "lab",
    "tag": "ssh-bruteforce-lab",
    "confidence": "low",
    "note": "IOC simulasi lab, bukan ancaman nyata"
  }
]
```

Python: Cek IOC terhadap Evidence

Buat script:

```
nano ~/soc-lab/scripts/cek_ioc_lokal.py
```

Isi:

```
from pathlib import Path
import json

ioc_file = Path.home() / "soc-lab" / "ioc" / "ioc-list.json"
evidence_dir = Path.home() / "soc-lab" / "evidence" / "incident" /
"ssh-bruteforce-001"

iocs = json.loads(ioc_file.read_text(encoding="utf-8"))

print("Cek IOC Lokal")
print("=" * 30)

for ioc in iocs:
    value = ioc["value"]
    found = []

    for file_path in evidence_dir.glob("*"):
        if file_path.is_file():
            text = file_path.read_text(encoding="utf-8",
errors="ignore")
            if value in text:
                found.append(file_path.name)

    print(f"IOC: {value}")
    print(f"Tipe: {ioc['type']}")
    print(f"Tag: {ioc['tag']}")
    print(f"Ditemukan di: {', '.join(found) if found else 'tidak
ditemukan'}")
    print("-" * 30)
```

Jalankan:

```
python3 ~/soc-lab/scripts/cek_ioc_lokal.py
```

Export JSON untuk Case

```
cp ~/soc-lab/ioc/ioc-list.json \
~/soc-lab/evidence/incident/ssh-bruteforce-001/ioc-list.json
```

Threat intelligence bukan vonis. Ia adalah konteks yang harus divalidasi dengan log lokal.

9.5 Mapping MITRE ATT&CK

MITRE ATT&CK membantu menghubungkan alert dengan taktik dan teknik serangan.

Contoh:

- brute force;
- credential access;
- discovery;
- lateral movement;
- persistence.

Mengapa Mapping Penting?

Mapping membantu:

- menjelaskan serangan ke manajemen;
- menyusun deteksi;
- membuat playbook;
- menilai gap SOC;
- menyusun laporan incident;
- menghubungkan alert dengan perilaku attacker.

Contoh Mapping Sederhana

Aktivitas	Taktik	Teknik
SSH login gagal berulang	Credential Access	Brute Force
Port scan	Discovery	Network Service Discovery
User baru dibuat	Persistence	Create Account
Koneksi antar host	Lateral Movement	Remote Services
Modifikasi service	Persistence	Create or Modify System Process

Mapping untuk SSH Brute Force

Alert:

```
SSH login gagal berulang
```

Taktik:

```
Credential Access
```

Teknik:

```
Brute Force
```

Bukti:

```
auth.log
Wazuh alerts.json
Suricata eve.json
Zeek conn.log
```

Template Mapping

```
nano ~/soc-lab/evidence/incident/ssh-bruteforce-001/mitre-mapping.txt
```

Isi:

```
# MITRE ATT&CK Mapping

Incident:
Suspected SSH Brute Force

Tactic:
Credential Access

Technique:
Brute Force

Evidence:
- auth.log
- wazuh-alerts.json
- zeek-conn.log
- suricata.json

Catatan:
Mapping ini berdasarkan pola login gagal berulang.
Perlu validasi apakah ada login sukses setelah percobaan
gagal.
```

Prinsip Mapping

Jangan memaksakan mapping.
Jika bukti hanya menunjukkan scan, jangan langsung menyimpulkan lateral movement.
Jika bukti hanya menunjukkan login gagal, jangan menyimpulkan compromise.

Mapping harus mengikuti bukti, bukan keinginan analis.

9.6 SOAR dengan Shuffle

SOAR membantu otomasi respons.

Namun otomasi harus dikontrol.

Workflow minimum:

- terima alert JSON;
- parsing field;
- enrich IOC;
- buat case;
- kirim notifikasi;
- minta approval;
- catat audit trail.

Shuffle memakai trigger untuk menjalankan workflow otomatis. Trigger bisa berupa webhook, schedule, atau eksekusi manual. (Shuffle) Wazuh dapat meneruskan alert ke Shuffle melalui webhook berdasarkan rule ID, group, atau level alert. (Wazuh)

Prinsip SOAR Aman

SOAR yang baik:

- tidak langsung memblokir tanpa validasi;
- tidak menghapus evidence;
- punya approval;
- menyimpan audit trail;
- bisa rollback;
- hanya menjalankan aksi sesuai playbook;
- membedakan lab dan produksi.

Workflow Minimum

```
Wazuh alert JSON
→ Shuffle webhook
→ parse field
→ ambil source IP
→ cek IOC
→ buat case / update case
→ kirim notifikasi
→ minta approval
→ jika approved, respons lab
→ catat audit trail
```

Field yang Diparsing

Dari alert JSON:

- timestamp;
- rule ID;
- rule level;
- rule description;
- agent name;
- agent IP;
- source IP;
- destination IP;
- user;
- full log;
- group.

Contoh Payload Alert Ringkas

```
{
  "timestamp": "2026-06-02T10:15:00+0700",
  "rule": {
    "id": "100300",
    "level": 10,
    "description": "SSH login gagal berulang dalam 120 detik"
  },
  "agent": {
    "name": "endpoint-01",
    "ip": "192.168.56.30"
  },
  "data": {
    "srcip": "192.168.56.20",
    "srcuser": "user_salah"
  }
}
```

Audit Trail

Setiap workflow harus mencatat:

- alert diterima;
- field yang diparsing;
- IOC yang dicek;
- case yang dibuat;
- notifikasi terkirim;
- approval diberikan atau ditolak;
- aksi yang dilakukan;
- hasil aksi;
- waktu selesai.

Template Audit Trail

```
nano ~/soc-lab/docs/template-soar-audit-trail.txt
```

Isi:

```
# SOAR Audit Trail

Workflow:
Trigger:
Waktu alert:
Rule ID:
Source IP:
Agent:
Case ID:
IOC enrichment:
Approval:
Aksi:
Hasil:
Rollback:
Operator:
Catatan:
```

Batas Aman

Untuk lab, respons boleh berupa:

- tulis log;
- buat case;
- kirim notifikasi;
- catat IOC;
- minta approval;
- blokir IP hanya di jaringan lab jika disetujui.

Untuk produksi, respons harus mengikuti kebijakan organisasi.

SOAR bukan autopilot. SOAR adalah workflow dengan kontrol manusia.

9.7 Playbook “SSH Brute Force Detected”

Playbook adalah langkah operasional yang bisa diulang.

Langkah:

1. alert masuk;
2. ambil source IP;
3. cek jumlah percobaan;
4. cek user target;
5. cek reputasi IP;
6. buat case;
7. minta approval;
8. blokir di lab jika disetujui;
9. catat hasil.

Tujuan Playbook

Playbook ini menjawab:

- apakah benar brute force?
- siapa sumbernya?
- endpoint mana targetnya?
- user apa yang dicoba?
- apakah ada login sukses?
- apakah perlu blokir?
- apakah perlu eskalasi?
- apa evidence-nya?

Langkah 1 — Alert Masuk

Baca alert:

```
sudo jq 'select(.rule.description |
test("SSH|password|login|authentication"; "i"))' \
/var/ossec/logs/alerts/alerts.json
```

Simpan:

```
mkdir -p ~/soc-lab/evidence/playbook/ssh-bruteforce
sudo jq 'select(.rule.description |
test("SSH|password|login|authentication"; "i"))' \
/var/ossec/logs/alerts/alerts.json \
> ~/soc-lab/evidence/playbook/ssh-bruteforce/wazuh-ssh-alerts.json
```

Langkah 2 — Ambil Source IP

Jika field data.srcip tersedia:

```
sudo jq -r 'select(.data.srcip != null) | .data.srcip' \
/var/ossec/logs/alerts/alerts.json \
| sort \
| uniq -c \
| sort -nr
```

Langkah 3 — Cek Jumlah Percobaan

```
sudo grep "Failed password" /var/log/auth.log \
| awk '{print $(NF-3)}' \
| sort \
| uniq -c \
| sort -nr
```

Jika format field tidak cocok, gunakan grep IP langsung:

```
sudo grep "<SOURCE_IP>" /var/log/auth.log
```

Langkah 4 — Cek User Target

```
sudo grep "<SOURCE_IP>" /var/log/auth.log \
| grep "Failed password"
```

Cari user yang dicoba:

```
sudo grep "<SOURCE_IP>" /var/log/auth.log \
| grep "Failed password" \
| tee
~/soc-lab/evidence/playbook/ssh-bruteforce/failed-users.
log
```

Langkah 5 — Cek Login Sukses

```
sudo grep "<SOURCE_IP>" /var/log/auth.log \
| grep "Accepted"
```

Jika ada login sukses setelah gagal berulang, eskalasi.

Langkah 6 — Cek Bukti Network

Suricata:

```
sudo grep "<SOURCE_IP>" /var/log/suricata/eve.json \
| tee
~/soc-lab/evidence/playbook/ssh-bruteforce/suricata-source-ip.json
```

Zeek:

```
grep "<SOURCE_IP>" /opt/zeek/logs/current/conn.log \
| tee ~/soc-lab/evidence/playbook/ssh-bruteforce/zeek-source-ip.log
```

Langkah 7 — Buat Case

```
nano ~/soc-lab/evidence/playbook/ssh-bruteforce/case.md
```

Isi:

```
# Case: SSH Brute Force Detected

## Severity
High

## Alert
SSH login gagal berulang.

## Source
- Source IP:
- Target IP:
- Target host:
- User target:

## Evidence
- wazuh-ssh-alerts.json
- failed-users.log
- suricata-source-ip.json
- zeek-source-ip.log

## Timeline
| Waktu | Event | Sumber |
|------|-------|--------|
|      |       |        |

## Analisis
Tuliskan hasil analisis.

## Keputusan
[ ] False positive
[ ] Simulasi lab
[ ] Incident
[ ] Perlu eskalasi

## Approval
Nama:
Waktu:
Keputusan:

## Response
Tindakan:

## Closure
Status:
Lesson learned:
```

Langkah 8 — Minta Approval

Untuk lab, approval bisa dicatat manual:

```
Approval:
Disetujui oleh:
Jam:
Aksi:
Batas:
Rollback:
```

Langkah 9 — Blokir di Lab Jika Disetujui

Hanya di lab.
Contoh dengan UFW:

```
sudo ufw deny from <SOURCE_IP> to any
```

Cek:

```
sudo ufw status numbered
```

Rollback:

```
sudo ufw status numbered
sudo ufw delete <NOMOR_RULE>
```

Jangan melakukan blokir di produksi tanpa prosedur dan approval.

Langkah 10 — Catat Hasil

```
nano ~/soc-lab/evidence/playbook/ssh-bruteforce/response-log.txt
```

Isi:

```
# Response Log

Tanggal:
Source IP:
Aksi:
Approval:
Command:
Hasil:
Rollback:
Status:
Catatan:
```

9.8 Hands-on Python: Membuat Draft Case dari Alert JSON

Python membantu mempercepat dokumentasi.

Buat script:

```
nano ~/soc-lab/scripts/buat_case_ssh_bruteforce.py
```

Isi:

```
from pathlib import Path
import json
from datetime import datetime
from collections import Counter

base = Path.home() / "soc-lab" / "evidence" / "playbook" /
"ssh-bruteforce"
alert_file = base / "wazuh-ssh-alerts.json"
case_file = base / "case-auto.md"

if not alert_file.exists():
    raise FileNotFoundError(f"File tidak ditemukan:
{alert_file}")

events = []

with alert_file.open("r", encoding="utf-8", errors="ignore") as
f:
    for line in f:
        try:
            events.append(json.loads(line))
        except json.JSONDecodeError:
            continue

src_counter = Counter()
agent_counter = Counter()
rule_counter = Counter()

for event in events:
    src_ip = event.get("data", {}).get("srcip", "-")
    agent = event.get("agent", {}).get("name", "-")
    rule = event.get("rule", {}).get("description", "-")

    src_counter[src_ip] += 1
    agent_counter[agent] += 1
    rule_counter[rule] += 1

lines = []
lines.append("# Case Auto: SSH Brute Force Detected")
lines.append("")
lines.append(f"Dibuat: {datetime.now().isoformat()}")
lines.append("")
lines.append("## Ringkasan")
lines.append(f"Jumlah alert dibaca: {len(events)}")
lines.append("")
lines.append("## Top Source IP")
```

```
for ip, count in src_counter.most_common(10):
    lines.append(f"- {ip}: {count}")
lines.append("")
lines.append("## Top Agent")
for agent, count in agent_counter.most_common(10):
    lines.append(f"- {agent}: {count}")
lines.append("")
lines.append("## Top Rule")
for rule, count in rule_counter.most_common(10):
    lines.append(f"- {rule}: {count}")
lines.append("")
lines.append("## Triage")
lines.append("- [ ] Cek jumlah percobaan login")
lines.append("- [ ] Cek user target")
lines.append("- [ ] Cek login sukses")
lines.append("- [ ] Cek Suricata")
lines.append("- [ ] Cek Zeek")
lines.append("- [ ] Cek IOC")
lines.append("- [ ] Minta approval sebelum response")
lines.append("")
lines.append("## Kesimpulan")
lines.append("Belum final. Perlu validasi analis.")
lines.append("")

case_file.write_text("\n".join(lines), encoding="utf-8")
print(f"Case draft dibuat: {case_file}")
```

Jalankan:

```
python3 ~/soc-lab/scripts/buat_case_ssh_bruteforce.py
cat ~/soc-lab/evidence/playbook/ssh-bruteforce/case-auto.md
```

Makna Script

Script ini:

- membaca alert JSON;
- menghitung source IP;
- menghitung agent;
- menghitung rule;
- membuat draft case Markdown;
- menambahkan checklist triage.

Python membantu menulis draft. Keputusan tetap di tangan analis.

9.9 Checklist Validasi Akhir Bab 9

Triage

- Bisa membedakan alert dan incident.
- Bisa memakai checklist triage.
- Bisa mencari bukti Wazuh.
- Bisa mencari bukti auth.log.
- Bisa mencari bukti Suricata.
- Bisa mencari bukti Zeek.
- Bisa menentukan status alert.

Case Management

- Case memiliki judul.
- Case memiliki severity.
- Case memiliki deskripsi.
- Case memiliki timeline.
- Case memiliki observables.
- Case memiliki evidence.
- Case memiliki task.
- Case memiliki analyst note.
- Case memiliki recommendation.
- Case memiliki closure.

MISP / Threat Intelligence

- IOC dipahami.
- Event dipahami.
- Attribute dipahami.
- Tag dipahami.
- Galaxy dipahami.
- Feed dipahami.
- Export JSON dipahami.
- IOC divalidasi dengan log lokal.

MITRE ATT&CK

- Brute force dipetakan.
- Credential access dipahami.
- Discovery dipahami.
- Lateral movement tidak disimpulkan tanpa bukti.
- Persistence tidak disimpulkan tanpa bukti.
- Mapping masuk laporan.

SOAR

- Workflow minimum dirancang.
- Alert JSON diterima.
- Field diparsing.
- IOC enrichment dirancang.
- Case dibuat.
- Notifikasi dirancang.
- Approval disiapkan.
- Audit trail dicatat.

Playbook SSH Brute Force

- Alert masuk.
- Source IP diambil.
- Jumlah percobaan dicek.
- User target dicek.
- Reputasi IP / IOC dicek.
- Case dibuat.
- Approval dicatat.
- Respons lab dilakukan jika disetujui.
- Hasil dicatat.
- Lesson learned ditulis.

9.10 Kesalahan Umum Bab 9

1. Semua Alert Dianggap Incident

Akibat:

- analis lelah;
- case terlalu banyak;
- prioritas kabur;
- respons berlebihan.

Solusi:

- triage dulu;
- cari bukti;
- nilai dampak;
- baru buat incident.

2. Case Tanpa Timeline

Case tanpa timeline sulit dipercaya.

Timeline harus menjawab:

- kapan mulai?
- kapan alert muncul?
- kapan bukti tambahan ditemukan?
- kapan respons dilakukan?
- kapan ditutup?

3. IOC Ditelan Mentah-Mentah

Threat intelligence bisa salah konteks.

IOC harus divalidasi dengan log lokal.

4. MITRE Mapping Dipaksakan

Jangan memetakan lateral movement jika hanya ada port scan.

Jangan memetakan persistence jika hanya ada login gagal.

5. SOAR Terlalu Agresif

Otomasi tanpa approval dapat merusak layanan.

Mulai dari:

- parse;
- enrich;
- buat case;
- notifikasi;
- approval.

6. Response Tanpa Audit Trail

Jika tindakan tidak tercatat, respons sulit dipertanggungjawabkan.

Catat:

- siapa;
- kapan;
- command;
- alasan;
- hasil;
- rollback.

9.11 Rangkuman Bab

Bab ini mengubah alert menjadi proses incident response yang terukur.

Pembaca belajar:

- membedakan alert dan incident;
- melakukan triage alert;
- membuat case;
- menyimpan observables dan evidence;
- memakai MISP untuk konteks IOC;
- memetakan alert ke MITRE ATT&CK;
- merancang workflow SOAR;
- menjalankan playbook SSH brute force;
- meminta approval sebelum respons berisiko;
- mencatat audit trail;
- membuat draft case dengan Python.

Inti Bab 9:

> **Alert adalah sinyal. Case adalah investigasi. Response adalah keputusan. Semua harus berbasis bukti.**

SOC yang matang bukan hanya mendeteksi.
SOC yang matang mampu mengelola insiden dari awal sampai penutupan.

Output Bab

Setelah menyelesaikan Bab 9, pembaca mampu:

- mengubah alert menjadi case;
- menjalankan triage berbasis checklist;
- menyimpan observables dan evidence;
- membuat timeline;
- menambahkan IOC;
- memahami event, attribute, tag, galaxy, feed, dan export JSON di MISP;
- melakukan mapping MITRE ATT&CK sederhana;
- merancang workflow SOAR dengan kontrol manusia;
- menjalankan playbook “SSH Brute Force Detected”;
- mencatat approval dan audit trail;
- membuat draft case otomatis dengan Python.

Kalimat kunci Bab 9:

> **SOC yang kuat bukan hanya cepat melihat alert, tetapi disiplin mengubah alert menjadi case, memberi konteks IOC, mengambil keputusan berbasis bukti, dan mencatat setiap respons agar dapat diaudit.**

Bab 10 — Proyek Akhir: Mini SOC Ubuntu 26.04 Siap Operasi

Bab 10 yang mewajibkan satu SOC server, satu endpoint agent, satu target lab, Wazuh server, Wazuh agent, FIM aktif, minimal tiga rule/deteksi, akses log via CLI, dashboard minimum, satu case incident, dan satu laporan akhir. Dokumen yang sama juga menempatkan evidence, KPI, format laporan, rubrik penilaian, serta etika sebagai bagian wajib proyek akhir.

Secara teknis, proyek akhir ini mengikat semua bab sebelumnya: Wazuh untuk SIEM/XDR dan alert endpoint, FIM untuk memantau perubahan file, Security Onion/Suricata/Zeek untuk visibilitas jaringan, OpenSearch untuk query dan dashboard, serta DFIR-IRIS/MISP/Shuffle untuk case, IOC, dan otomasi terkontrol. Wazuh FIM resmi memantau pembuatan, perubahan, dan penghapusan file/direktori serta memicu alert saat ada perubahan; Security Onion memakai Suricata dan Zeek untuk visibilitas jaringan; OpenSearch Alerting memakai monitor, trigger, dan action untuk menjalankan query dan memicu alert; integrasi Wazuh dengan DFIR-IRIS dapat membantu korelasi event, pelacakan investigasi, dan koordinasi response. (Dokumentasi Wazuh)

10.1 Spesifikasi Mini SOC

Proyek akhir ini bukan sekadar latihan instalasi.

Proyek akhir ini harus menjadi **portofolio teknis**.

Artinya, hasil akhir harus bisa ditunjukkan sebagai bukti bahwa pembaca mampu:

- membangun server SOC;
- memasang agent;
- membaca log;
- membuat deteksi;
- menguji alert;
- menyimpan evidence;
- membuat case;
- menulis laporan;
- melakukan response dengan etika.

Spesifikasi Wajib

Komponen wajib:

- **1 Ubuntu 26.04 SOC server**
- **1 endpoint Ubuntu agent**
- **1 target lab**
- **Wazuh server**
- **Wazuh agent**
- **FIM aktif**
- **minimal 3 rule / deteksi**
- **log dapat dibaca dari CLI**
- **dashboard minimum**
- **1 case incident**
- **1 laporan akhir**

Spesifikasi Opsional

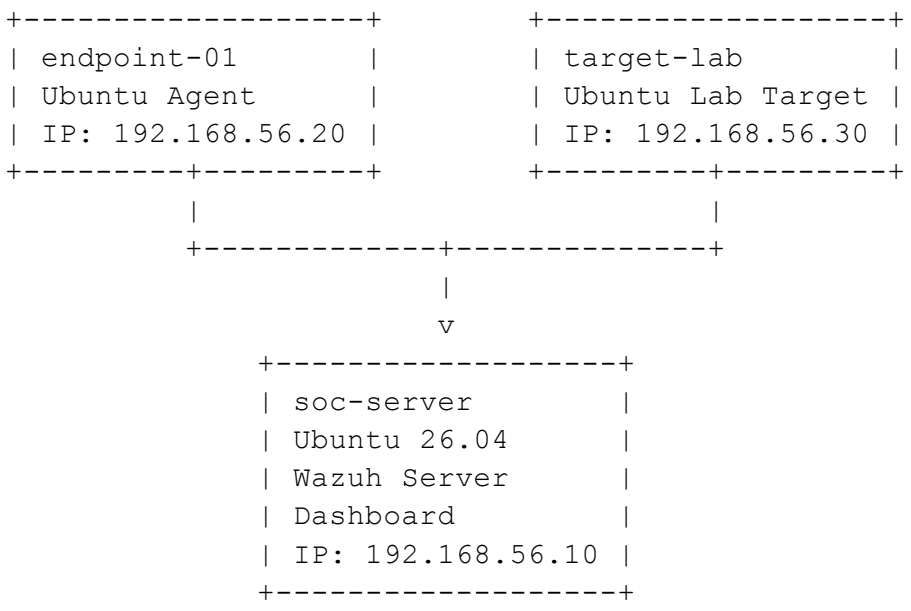
Komponen opsional:

- Security Onion;
- Suricata;
- Zeek;
- MISP;
- DFIR-IRIS;
- Shuffle.

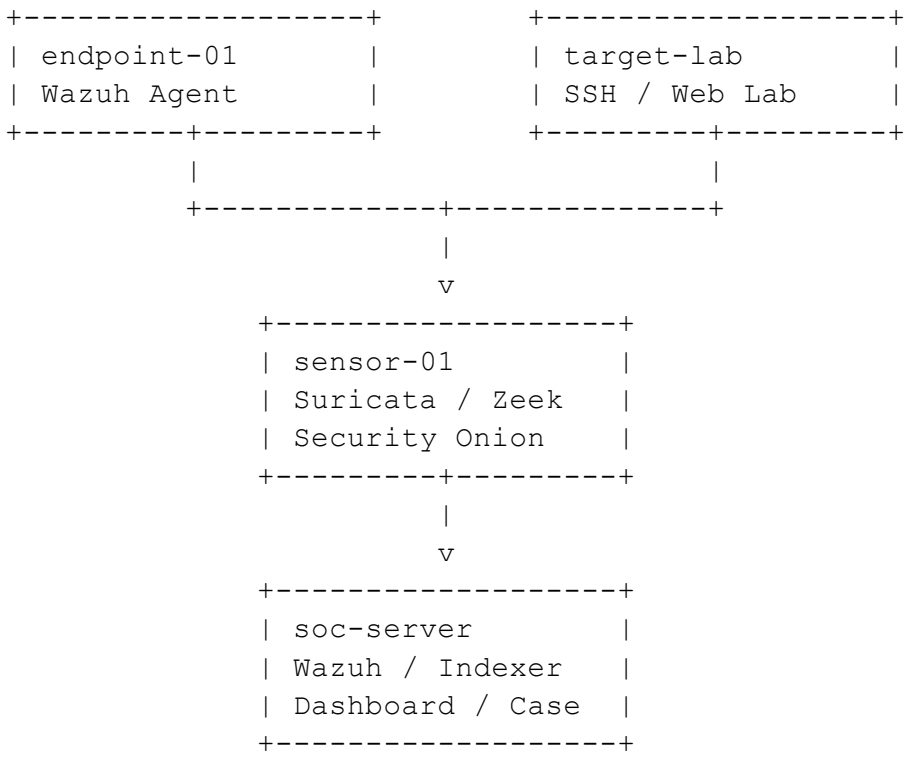
Komponen opsional bukan berarti tidak penting. Namun jika hardware terbatas, lebih baik membangun sistem kecil yang stabil daripada sistem besar yang tidak bisa dijelaskan.

Mini SOC yang rapi lebih kuat daripada SOC besar yang rapuh.

Topologi Minimum



Topologi Ideal



Tabel Aset Proyek

Buat file:

```
nano ~/soc-lab/docs/aset-mini-soc.txt
```

Isi:

```
# Daftar Aset Mini SOC

+-------------+---------------+-----------------+--------------+------------------+--------
--+
| Hostname    | IP            | Role            | OS           | Tool             | Status
|
+-------------+---------------+-----------------+--------------+------------------+--------
--+
| soc-server  | 192.168.56.10 | SOC server      | Ubuntu 26.04 | Wazuh, dashboard | aktif
|
| endpoint-01 | 192.168.56.20 | endpoint agent  | Ubuntu 26.04 | Wazuh agent      | aktif
|
| target-lab  | 192.168.56.30 | target uji      | Ubuntu 26.04 | SSH/web lab      | aktif
|
| sensor-01   | 192.168.56.40 | sensor opsional | Ubuntu 26.04 | Suricata/Zeek    |
opsional |
+-------------+---------------+-----------------+--------------+------------------+--------
--+
```

Struktur Folder Proyek Akhir

```
mkdir -p
~/soc-lab/final-project/{architecture,configs,evidence,querie
s,cases,reports,scripts,playbooks}
```

Cek:

```
tree ~/soc-lab/final-project
```

Struktur:

```
~/soc-lab/final-project/
├── architecture
├── cases
├── configs
├── evidence
├── playbooks
├── queries
├── reports
└── scripts
```

Makna:

Folder	Isi
architecture	topologi, IP, role mesin
configs	salinan konfigurasi penting
evidence	alert JSON, log CLI, PCAP, hash
queries	query OpenSearch / jq
cases	case incident
reports	laporan akhir
scripts	script Python
playbooks	langkah respons

10.2 Skenario Uji

Skenario uji minimal:

1. login gagal berulang;
2. perubahan file penting;
3. port scan di lab;
4. service mati;
5. akses web mencurigakan.

Semua skenario harus dilakukan **hanya di lab sendiri**.

Jangan menguji sistem publik.

Jangan menguji jaringan yang tidak punya izin.

Skenario 1 — Login Gagal Berulang

Tujuan:

- menguji log autentikasi;
- menguji Wazuh alert;
- menguji custom rule;
- menguji triage SSH brute force.

Di endpoint atau target lab:

```
sudo tail -f /var/log/auth.log
```

Dari mesin lab lain:

```
ssh user_salah@<IP_TARGET_LAB>
```

Ulangi beberapa kali secara manual.

Di SOC server:

```
sudo jq 'select(.rule.description |
test("SSH|password|authentication|login"; "i"))' \
/var/ossec/logs/alerts/alerts.json \
| tee ~/soc-lab/final-project/evidence/ssh-login-failed.json
```

Pertanyaan analisis:

- source IP siapa?
- user apa yang dicoba?
- berapa kali gagal?
- apakah ada login sukses?
- rule apa yang memicu?
- level berapa?

Skenario 2 — Perubahan File Penting

Tujuan:

- menguji FIM;
- menguji alert perubahan file;
- menguji evidence perubahan konfigurasi.

Gunakan file dummy dulu:

```
sudo touch /etc/test-final-soc.conf
echo "uji proyek akhir SOC" | sudo tee -a
/etc/test-final-soc.conf
sudo rm /etc/test-final-soc.conf
```

Di SOC server:

```
sudo jq 'select(.syscheck.path? |
test("test-final-soc.conf"))' \
/var/ossec/logs/alerts/alerts.json \
| tee
~/soc-lab/final-project/evidence/fim-test-final.json
```

Pertanyaan analisis:

- file apa berubah?
- dibuat, diubah, atau dihapus?
- endpoint mana?
- user/proses apa?
- apakah perubahan sah?

Skenario 3 — Port Scan di Lab

Tujuan:

- menguji sensor jaringan;
- menguji Suricata;
- menguji Zeek;
- menguji korelasi alert jaringan.

Di mesin lab:

```
nmap -sS <IP_TARGET_LAB>
```

Hanya ke target lab milik sendiri.

Di sensor:

```
sudo jq 'select(.event_type=="alert")'
/var/log/suricata/eve.json \
| tee
~/soc-lab/final-project/evidence/suricata-portscan.json
```

Zeek:

```
grep "<IP_TARGET_LAB>" /opt/zeek/logs/current/conn.log \
| tee
~/soc-lab/final-project/evidence/zeek-portscan-conn.log
```

Pertanyaan analisis:

- source IP siapa?
- target IP siapa?
- port apa saja?
- signature apa?
- ada flow?
- apakah endpoint target juga punya alert?

Skenario 4 — Service Mati

Tujuan:

- menguji monitoring service;
- menguji log systemd;
- menguji respons operasional.

Gunakan service lab, bukan service kritikal.

Contoh memakai SSH hanya jika ada akses konsol. Lebih aman gunakan service dummy atau web lab.

Cek service:

```
systemctl status ssh
```

Simulasi service mati di target lab:

```
sudo systemctl stop ssh
```

Cek log:

```
journalctl -u ssh --since "10 minutes ago" \
| tee
~/soc-lab/final-project/evidence/service-ssh-stop.log
```

Hidupkan kembali:

```
sudo systemctl start ssh
```

Pertanyaan analisis:

- service apa mati?
- kapan mati?
- siapa yang menghentikan?
- dampaknya apa?
- apakah ada alert?
- bagaimana rollback?

Skenario 5 — Akses Web Mencurigakan

Tujuan:

- menguji log web;
- menguji pola akses;
- menguji korelasi endpoint dan network.

Jika target lab memiliki web server:

```
curl "http://<IP_TARGET_LAB>/../../etc/passwd"
curl "http://<IP_TARGET_LAB>/admin"
curl "http://<IP_TARGET_LAB>/login"
```

Di target:

```
sudo tail -f /var/log/nginx/access.log
```

Simpan:

```
sudo grep "<IP_SUMBER>" /var/log/nginx/access.log \
| tee
~/soc-lab/final-project/evidence/web-suspicious-access.l
og
```

Jika Suricata aktif:

```
sudo grep "<IP_SUMBER>" /var/log/suricata/eve.json \
| tee ~/soc-lab/final-project/evidence/web-suricata.json
```

Pertanyaan analisis:

- URI apa yang diakses?
- status code apa?
- apakah terlihat scanning?
- apakah ada alert Suricata?
- apakah ada file berubah di web root?

10.3 Evidence yang Harus Dikumpulkan

Evidence wajib:

- screenshot dashboard;
- alert JSON;
- log CLI;
- timeline;
- case note;
- IOC;
- tindakan respons;
- lesson learned.

Prinsip Evidence

Evidence harus:

- jelas sumbernya;
- punya waktu;
- punya konteks;
- tidak diedit;
- diberi hash jika file penting;
- disimpan dalam folder rapi;
- bisa dipakai untuk laporan.

Tanpa evidence, incident report hanya opini.

Struktur Evidence

```
mkdir -p
~/soc-lab/final-project/evidence/{dashboard,json,cli,timeline
,case,ioc,response,lesson-learned,hash}
```

Cek:

```
tree ~/soc-lab/final-project/evidence
```

1. Screenshot Dashboard

Simpan ke:

```
~/soc-lab/final-project/evidence/dashboard/
```

Isi minimal:

- dashboard alert per severity;
- top agent;
- top source IP;
- failed login;
- FIM alert;
- timeline.

Catatan:

- hapus data sensitif jika laporan dibagikan;
- jangan tampilkan password/token;
- jangan tampilkan alamat publik sensitif.

2. Alert JSON

Wazuh high alert:

```
sudo jq 'select(.rule.level >= 10)'
/var/ossec/logs/alerts/alerts.json \
> ~/soc-lab/final-project/evidence/json/wazuh-high-alerts.json
```

FIM:

```
sudo jq 'select(.rule.groups[]? == "syscheck")'
/var/ossec/logs/alerts/alerts.json \
> ~/soc-lab/final-project/evidence/json/wazuh-fim-alerts.json
```

Suricata:

```
sudo jq 'select(.event_type=="alert")'
/var/log/suricata/eve.json \
> ~/soc-lab/final-project/evidence/json/suricata-alerts.json
```

3. Log CLI

Auth log:

```
sudo grep "Failed password" /var/log/auth.log \
> ~/soc-lab/final-project/evidence/cli/auth-failed-password.log
```

Systemd:

```
journalctl --since "1 hour ago" \
> ~/soc-lab/final-project/evidence/cli/journal-last-hour.log
```

Zeek:

```
grep "<IP_TARGET_LAB>" /opt/zeek/logs/current/conn.log \
> ~/soc-lab/final-project/evidence/cli/zeek-conn-target.log
```

4. Timeline

```
nano
~/soc-lab/final-project/evidence/timeline/timeline-incident-001.md
```

Isi:

```
# Timeline Incident 001

+-------+----------+--------------------+--------------------------+----------------+
| Waktu | Sumber   | Event              | Evidence                 | Catatan        |
+-------+----------+--------------------+--------------------------+----------------+
| 10:01 | Zeek     | Koneksi ke port 22 | zeek-conn-target.log     | Awal aktivitas |
| 10:02 | auth.log | Failed password    | auth-failed-password.log | User salah     |
| 10:03 | Wazuh    | Alert level tinggi | wazuh-high-alerts.json   | Rule match     |
| 10:05 | Case     | Incident dibuat    | case-incident-001.md     | Triage selesai |
+-------+----------+--------------------+--------------------------+----------------+
```

5. Case Note

```
nano
~/soc-lab/final-project/evidence/case/case-incident-001.md
```

Isi:

```
# Case Incident 001

## Judul
Suspected SSH Brute Force di Endpoint Lab

## Severity
High

## Deskripsi
Terdapat login gagal berulang ke endpoint lab.

## Observables
- Source IP:
- Destination IP:
- User:
- Port:

## Evidence
- auth-failed-password.log
- wazuh-high-alerts.json
- suricata-alerts.json
- zeek-conn-target.log

## Analisis
Tuliskan analisis.

## Response
Tuliskan tindakan.

## Closure
Tuliskan status akhir.
```

6. IOC

```
nano
~/soc-lab/final-project/evidence/ioc/ioc-incident-001.json
```

Isi:

```
[
  {
    "type": "ip-src",
    "value": "192.168.56.20",
    "context": "source IP simulasi SSH brute force lab",
    "confidence": "low",
    "tag": "lab-simulation"
  }
]
```

7. Tindakan Respons

```
nano
~/soc-lab/final-project/evidence/response/response-incident-0
01.txt
```

Isi:

```
Tanggal:
Incident:
Tindakan:
Approval:
Command:
Hasil:
Rollback:
Catatan:
```

8. Lesson Learned

```
nano
~/soc-lab/final-project/evidence/lesson-learned/lesson-learne
d-incident-001.md
```

Isi:

```
# Lesson Learned Incident 001

## Apa yang Berhasil
- Alert muncul.
- Log CLI dapat dibaca.
- Evidence tersimpan.
- Case dibuat.

## Apa yang Lemah
- Rule perlu tuning.
- Dashboard perlu panel tambahan.
- Dokumentasi masih perlu dirapikan.

## Perbaikan
- Tambah threshold.
- Tambah query OpenSearch.
- Tambah playbook response.
```

9. Hash Evidence

```
find ~/soc-lab/final-project/evidence -type f -exec sha256sum {} \; \
> ~/soc-lab/final-project/evidence/hash/SHA256SUMS.txt
```

Hash membuat evidence lebih kuat karena perubahan file dapat dideteksi.

10.4 KPI Mini SOC

KPI Mini SOC:

- MTTD;
- MTTR;
- jumlah alert;
- jumlah false positive;
- jumlah incident closed;
- jumlah rule aktif;
- jumlah endpoint monitored.

1. MTTD

MTTD berarti *Mean Time To Detect*.

Rumus sederhana:

MTTD = waktu deteksi - waktu kejadian awal

Contoh:

Kejadian awal: 10:01
Alert muncul: 10:03
MTTD: 2 menit

2. MTTR

MTTR berarti *Mean Time To Respond*.

Rumus sederhana:

MTTR = waktu response - waktu deteksi

Contoh:

Alert muncul: 10:03
Response dilakukan: 10:15
MTTR: 12 menit

3. Jumlah Alert

Hitung alert level tinggi:

```
sudo jq 'select(.rule.level >= 10)'
/var/ossec/logs/alerts/alerts.json \
| wc -l
```

4. Jumlah False Positive

False positive harus dicatat manual dari triage.

Buat file:

```
nano
~/soc-lab/final-project/reports/false-positive-log.csv
```

Isi:

```
tanggal,alert,source_ip,agent,alasan,status
2026-06-02,SSH failed
login,192.168.56.20,endpoint-01,simulasi lab,closed
```

5. Jumlah Incident Closed

Hitung case yang selesai:

```
find ~/soc-lab/final-project/cases -name "*.md" | wc -l
```

Atau catat manual:

```
incident_id,judul,severity,status
INC-001,SSH Brute Force Lab,High,closed
```

6. Jumlah Rule Aktif

Catat:

```
sudo grep -c "<rule id="
/var/ossec/etc/rules/local_rules.xml
```

Atau manual:

```
Rule aktif lokal:
- 100300 SSH login gagal berulang
- 100301 FIM file penting
- 100302 akses web mencurigakan
```

7. Jumlah Endpoint Monitored

Di Wazuh server:

```
sudo /var/ossec/bin/agent_control -lc
```

Atau lewat API sesuai Bab 3.

Template KPI

```
nano ~/soc-lab/final-project/reports/kpi-mini-soc.md
```

Isi:

```
# KPI Mini SOC

+--------------------+----------+-----------------------+-----------------------------+
| KPI                | Nilai    | Cara Hitung           | Catatan                     |
+--------------------+----------+-----------------------+-----------------------------+
| MTTD               | 2 menit  | alert - kejadian awal | SSH brute force lab         |
| MTTR               | 12 menit | response - alert      | blokir lab setelah approval |
| Jumlah alert       | 15       | jq / dashboard        | termasuk simulasi           |
| False positive     | 3        | triage manual         | aktivitas lab               |
| Incident closed    | 1        | case closed           | INC-001                     |
| Rule aktif         | 3        | local_rules.xml       | rule lokal                  |
| Endpoint monitored | 1        | agent_control         | endpoint-01                 |
+--------------------+----------+-----------------------+-----------------------------+
```

Python: Membuat Ringkasan KPI

Buat script:

```
nano ~/soc-lab/final-project/scripts/hitung_kpi_sederhana.py
```

Isi:

```
from pathlib import Path
import json

base = Path.home() / "soc-lab" / "final-project"
evidence = base / "evidence" / "json"
report = base / "reports"
report.mkdir(parents=True, exist_ok=True)

wazuh_file = evidence / "wazuh-high-alerts.json"

alert_count = 0
agents = set()
rules = set()

if wazuh_file.exists():
    with wazuh_file.open("r", encoding="utf-8",
errors="ignore") as f:
        for line in f:
            try:
                event = json.loads(line)
            except json.JSONDecodeError:
                continue

            alert_count += 1
```

```
                agent = event.get("agent", {}).get("name")
                rule = event.get("rule", {}).get("description")

                if agent:
                    agents.add(agent)
                if rule:
                    rules.add(rule)

output = report / "kpi-ringkas.md"

lines = []
lines.append("# KPI Ringkas Mini SOC")
lines.append("")
lines.append(f"- Jumlah high alert: {alert_count}")
lines.append(f"- Jumlah agent terlibat: {len(agents)}")
lines.append(f"- Jumlah rule terlibat: {len(rules)}")
lines.append("")
lines.append("## Agent")
for agent in sorted(agents):
    lines.append(f"- {agent}")
lines.append("")
lines.append("## Rule")
for rule in sorted(rules):
    lines.append(f"- {rule}")

output.write_text("\n".join(lines), encoding="utf-8")
print(f"KPI ringkas dibuat: {output}")
```

Jalankan:

```
python3
~/soc-lab/final-project/scripts/hitung_kpi_sederhana.py
cat ~/soc-lab/final-project/reports/kpi-ringkas.md
```

KPI bukan dekorasi laporan. KPI menunjukkan apakah SOC benar-benar bekerja.

10.5 Format Laporan Akhir

Struktur laporan akhir: **5–10 halaman**.

Isi:

1. ringkasan arsitektur;
2. daftar tools;
3. topologi jaringan;
4. konfigurasi utama;
5. skenario uji;
6. hasil alert;
7. evidence;
8. case incident;
9. response;
10. lesson learned.

Template Laporan Akhir

Buat file:

```
nano
~/soc-lab/final-project/reports/laporan-akhir-mini-soc.md
```

Isi:

```
# Laporan Akhir Mini SOC Ubuntu 26.04

## 1. Ringkasan Arsitektur
Mini SOC dibangun untuk memantau endpoint Linux, membaca log,
mendeteksi aktivitas mencurigakan, menyimpan evidence, dan
membuat
case incident.

## 2. Daftar Tools
- Ubuntu Server 26.04
- Wazuh Server
- Wazuh Agent
- OpenSearch / Wazuh Indexer
- Dashboard SOC
- auditd
- rsyslog
- jq, curl, grep, awk, journalctl, systemctl, ss, tcpdump
- Python
- Suricata / Zeek / Security Onion bila digunakan
- MISP / DFIR-IRIS / Shuffle bila digunakan

## 3. Topologi Jaringan
```

```
Tuliskan hostname, IP, role, dan alur log.

## 4. Konfigurasi Utama
Tuliskan:
- IP static
- SSH
- firewall
- Wazuh agent
- FIM
- rule lokal
- dashboard
- query penting

## 5. Skenario Uji
Tuliskan hasil:
1. login gagal berulang
2. perubahan file penting
3. port scan di lab
4. service mati
5. akses web mencurigakan

## 6. Hasil Alert
Ringkas alert:
- jumlah alert
- rule
- severity
- agent
- source IP
- waktu

## 7. Evidence
Daftar evidence:
- dashboard
- alert JSON
- log CLI
- timeline
- IOC
- response
- hash evidence

## 8. Case Incident
Tuliskan case utama:
- judul
- severity
- timeline
- observables
- analyst note
- recommendation
```

```
- closure

## 9. Response
Tuliskan:
- tindakan
- approval
- command
- hasil
- rollback bila ada

## 10. Lesson Learned
Tuliskan:
- apa yang berhasil
- apa yang gagal
- apa yang harus ditingkatkan
- rekomendasi operasional
```

Ringkasan Arsitektur

Contoh narasi:

```
Mini SOC ini memakai satu SOC server Ubuntu 26.04 sebagai pusat
pengumpulan dan analisis log. Endpoint Linux dipasang Wazuh Agent.
FIM aktif untuk memantau file penting. Dashboard digunakan untuk
visualisasi awal, sementara CLI digunakan untuk validasi evidence.
```

Daftar Tools

Tulis dalam tabel:

```
+---------------+-------------------------+--------+
| Tool          | Fungsi                  | Status |
+---------------+-------------------------+--------+
| Ubuntu 26.04  | OS server SOC           | aktif  |
| Wazuh         | SIEM/XDR                | aktif  |
| Wazuh Agent   | Endpoint telemetry      | aktif  |
| OpenSearch    | Index dan query log     | aktif  |
| auditd        | Audit aktivitas sistem | aktif  |
| Python        | Otomasi laporan         | aktif  |
+---------------+-------------------------+--------+
```

Topologi Jaringan

Gunakan teks sederhana:

```
endpoint-01
→ Wazuh Agent
→ soc-server
→ Wazuh Manager
→ Wazuh Indexer
→ Dashboard
→ Case
→ Report
```

Jika sensor aktif:

```
target-lab / endpoint-01
→ traffic mirror
→ sensor-01
→ Suricata / Zeek
→ evidence network
```

Konfigurasi Utama

Sertakan potongan konfigurasi yang penting, bukan seluruh file.

Contoh:

```
FIM:
- /etc/passwd
- /etc/shadow
- /etc/sudoers
- /etc/ssh/sshd_config
- /var/www/html
```

Hasil Alert

Gunakan tabel:

Waktu	Sumber	Rule	Level	Agent	Evidence
10:03	Wazuh	SSH failed login	10	endpoint-01	wazuh-high-alerts.json
10:10	Wazuh FIM	File changed	7	endpoint-01	fim-test-final.json
10:20	Suricata	Port scan	2	sensor-01	suricata-portscan.json

Lesson Learned

Lesson learned harus tajam.

Contoh:

- rule login gagal perlu threshold agar tidak bising;
- FIM harus fokus pada file kritikal;
- dashboard membantu triage tetapi CLI tetap wajib;
- evidence harus diberi hash;
- respons otomatis harus memakai approval;
- dokumentasi harus dibuat sejak awal, bukan di akhir.

Laporan akhir harus membuktikan kemampuan, bukan sekadar menceritakan instalasi.

10.6 Rubrik Penilaian

Rubrik:

- instalasi berhasil: 20%;
- agent dan sensor berjalan: 15%;
- akses log CLI: 15%;
- rule dan deteksi: 20%;
- incident report: 15%;
- dokumentasi dan etika: 15%.

Tabel Rubrik

Komponen	Bobot	Kriteria
Instalasi berhasil	20%	Server aktif, IP static, SSH, firewall, Wazuh berjalan
Agent dan sensor berjalan	15%	Agent aktif; sensor opsional membaca trafik
Akses log CLI	15%	Bisa membaca Wazuh, auth.log, journalctl, Suricata, Zeek
Rule dan deteksi	20%	Minimal 3 deteksi diuji dan menghasilkan alert
Incident report	15%	Case, timeline, evidence, response, closure
Dokumentasi dan etika	15%	Laporan rapi, evidence hash, simulasi legal

Detail Penilaian

1. Instalasi Berhasil — 20%

Nilai penuh jika:

- SOC server aktif;
- IP static benar;
- SSH aman;
- firewall aktif;
- Wazuh Manager aktif;
- indexer aktif;
- dashboard bisa diakses;
- konfigurasi dibackup.

CLI validasi:

```
hostnamectl
ip a
sudo systemctl status wazuh-manager
sudo systemctl status wazuh-indexer
sudo systemctl status wazuh-dashboard
sudo ufw status verbose
```

2. Agent dan Sensor Berjalan — 15%

Nilai penuh jika:

- Wazuh Agent aktif;
- agent terlihat di server;
- FIM berjalan;
- sensor jaringan opsional membaca trafik;
- Suricata/Zeek menghasilkan log jika digunakan.

CLI validasi:

```
sudo /var/ossec/bin/agent_control -lc
sudo systemctl status wazuh-agent
sudo tail -f /var/ossec/logs/alerts/alerts.json
```

Sensor:

```
sudo tail -f /var/log/suricata/eve.json
ls -lah /opt/zeek/logs/current/
```

3. Akses Log CLI — 15%

Nilai penuh jika pembaca bisa membaca:

- `journalctl`;
- `/var/log/auth.log`;
- Wazuh `alerts.json`;
- Suricata `eve.json`;
- Zeek `conn.log`;
- auditd bila digunakan.

CLI validasi:

```
journalctl --since "1 hour ago"
sudo grep "Failed password" /var/log/auth.log
sudo jq 'select(.rule.level >= 10)' /var/ossec/logs/alerts/alerts.json
sudo jq 'select(.event_type=="alert")' /var/log/suricata/eve.json
cat /opt/zeek/logs/current/conn.log | zeek-cut id.orig_h id.resp_h
id.resp_p proto service
```

4. Rule dan Deteksi — 20%

Minimal 3 deteksi:

- login gagal berulang;
- perubahan file penting;
- port scan;
- service mati;
- akses web mencurigakan.

Nilai penuh jika:

- rule jelas;
- threshold jelas;
- `wazuh-logtest` digunakan;
- alert muncul;
- false positive dibahas;
- evidence disimpan.

5. Incident Report — 15%

Nilai penuh jika laporan punya:

- judul case;
- severity;
- deskripsi;
- timeline;
- observables;
- evidence;
- analyst note;
- recommendation;
- response;
- closure;
- lesson learned.

6. Dokumentasi dan Etika — 15%

Nilai penuh jika:

- semua langkah terdokumentasi;
- evidence diberi hash;
- tidak menyerang sistem publik;
- tidak memakai malware asli;
- tidak membocorkan password/token;
- respons otomatis tidak agresif;
- approval dicatat.

Formula Nilai

```
Nilai akhir =
instalasi + agent/sensor + log CLI + rule/deteksi + incident
report + dokumentasi/etika
```

Contoh:

```
Instalasi: 18/20
Agent dan sensor: 12/15
Log CLI: 14/15
Rule dan deteksi: 17/20
Incident report: 13/15
Dokumentasi dan etika: 15/15

Total: 89/100
```

10.7 Pesan Penutup

SOC yang kuat bukan dibangun dari lisensi mahal.

SOC yang kuat dibangun dari admin yang:

- disiplin membaca log;
- memahami jaringan;
- menguji rule;
- menyimpan bukti;
- membuat timeline;
- menulis case;
- memahami risiko;
- merespons dengan tanggung jawab;
- mau belajar dari setiap alert.

Tools hanyalah alat. Disiplin analis adalah inti SOC.

Admin jaringan yang naik kelas menjadi analis SOC harus mengubah cara berpikir:

dari menjaga koneksi
menjadi membaca tanda bahaya

dari melihat dashboard
menjadi membuktikan dari log

dari panik karena alert
menjadi triage berbasis bukti

dari respons cepat tanpa catatan
menjadi respons terukur dan bisa diaudit

Mini SOC dalam buku ini bukan akhir.
Ia adalah fondasi.

Dari sini pembaca dapat berkembang ke:

- SOC multi-endpoint;
- sensor jaringan lebih lengkap;
- threat intelligence;
- case management kolaboratif;
- SOAR dengan approval;
- dashboard operasional;
- detection engineering;
- incident response matang;
- portofolio profesional.

Kalimat penutup:

> **Admin jaringan yang hebat bukan hanya menjaga jaringan tetap hidup, tetapi mampu membaca tanda bahaya, membuktikan insiden, menyimpan evidence, dan merespons dengan etika.**

10.8 Checklist Proyek Akhir

Gunakan checklist ini sebelum menyerahkan laporan.

Infrastruktur

- SOC server aktif.
- Endpoint agent aktif.
- Target lab aktif.
- IP static terdokumentasi.
- Timezone benar.
- SSH aman.
- Firewall aktif.
- Backup konfigurasi dibuat.

Wazuh

- Wazuh Manager aktif.
- Wazuh Indexer aktif.
- Dashboard aktif.
- Agent terhubung.
- Alert JSON terbaca.
- FIM aktif.
- Minimal 3 deteksi diuji.

Sensor Opsional

- Security Onion aktif.
- Suricata menghasilkan `eve.json`.
- Zeek menghasilkan `conn.log`.
- PCAP dapat diambil.
- Alert jaringan tersimpan.

Skenario Uji

- Login gagal berulang.
- Perubahan file penting.
- Port scan di lab.
- Service mati.
- Akses web mencurigakan.

Evidence

- Screenshot dashboard.
- Alert JSON.
- Log CLI.
- Timeline.
- Case note.
- IOC.
- Response log.
- Lesson learned.
- Hash evidence.

Laporan

- Ringkasan arsitektur.
- Daftar tools.
- Topologi jaringan.
- Konfigurasi utama.
- Skenario uji.
- Hasil alert.
- Evidence.
- Case incident.
- Response.
- Lesson learned.
- KPI.
- Etika.

10.9 Template Manifest Proyek

Buat manifest agar portofolio mudah diperiksa.

```
nano ~/soc-lab/final-project/MANIFEST.md
```

Isi:

```
# Manifest Proyek Akhir Mini SOC

## Identitas
Nama:
Tanggal:
Versi lab:

## Infrastruktur
- SOC server:
- Endpoint:
- Target lab:
- Sensor:

## Tools
- Wazuh:
- Wazuh Agent:
- OpenSearch / Indexer:
- Dashboard:
- Suricata:
- Zeek:
- MISP:
- Case Management:
- SOAR:
- Python:

## Skenario
- [ ] Login gagal berulang
- [ ] Perubahan file penting
- [ ] Port scan di lab
- [ ] Service mati
- [ ] Akses web mencurigakan

## Evidence Utama
- evidence/json/wazuh-high-alerts.json
- evidence/json/wazuh-fim-alerts.json
- evidence/cli/auth-failed-password.log
- evidence/timeline/timeline-incident-001.md
- evidence/case/case-incident-001.md
- evidence/hash/SHA256SUMS.txt
```

```
## Laporan
- reports/laporan-akhir-mini-soc.md
- reports/kpi-mini-soc.md
- reports/kpi-ringkas.md

## Status
[ ] Siap dinilai
[ ] Perlu perbaikan
```

10.10 Perintah Final Validasi

Jalankan sebelum laporan dikumpulkan.

Cek Folder

```
tree ~/soc-lab/final-project
```

Cek Agent

```
sudo /var/ossec/bin/agent_control -lc
```

Cek Alert

```
sudo jq 'select(.rule.level >= 10)'
/var/ossec/logs/alerts/alerts.json | head
```

Cek FIM

```
sudo jq 'select(.rule.groups[]? == "syscheck")'
/var/ossec/logs/alerts/alerts.json | head
```

Cek Auth Log

```
sudo grep "Failed password" /var/log/auth.log | tail
```

Cek Dashboard / Index

```
curl -k -u admin:<password>
https://localhost:9200/_cat/indices?v
```

Cek Evidence Hash

```
cat ~/soc-lab/final-project/evidence/hash/SHA256SUMS.txt
```

Cek Laporan

```
ls -lah ~/soc-lab/final-project/reports/
```

10.11 Kesalahan Umum Proyek Akhir

1. Terlalu Fokus Instalasi, Lupa Evidence

Instalasi penting.
Namun proyek akhir dinilai dari bukti kerja SOC.
Evidence wajib ada.

2. Dashboard Ada, CLI Tidak Bisa

Dashboard bukan pengganti CLI.

Pembaca harus bisa membaca:

- `alerts.json`;
- `auth.log`;
- `journalctl`;
- `eve.json`;
- `conn.log`.

3. Tidak Ada Case Incident

Alert tanpa case belum menunjukkan proses SOC.
Minimal buat satu case lengkap.

4. Rule Tidak Diuji

Rule harus diuji.

Gunakan:

```
sudo /var/ossec/bin/wazuh-logtest
```

5. Tidak Ada Timeline

Timeline adalah tulang punggung incident report.
Tanpa timeline, laporan lemah.

6. Tidak Ada Lesson Learned

Lesson learned menunjukkan kedewasaan SOC.

Tuliskan:

- apa yang berhasil;
- apa yang gagal;
- apa yang perlu diperbaiki.

7. Simulasi Tidak Etis

Jangan scan sistem publik.
Jangan memakai malware asli.
Jangan membuka dashboard ke jaringan luas.
Jangan membocorkan token/password.

Etika adalah bagian dari kompetensi SOC.

10.12 Rangkuman Bab

Bab ini menyatukan seluruh isi buku menjadi proyek akhir.

Pembaca tidak hanya memasang tools, tetapi membangun Mini SOC yang bisa diuji:

- server aktif;
- agent aktif;
- FIM berjalan;
- rule diuji;
- alert muncul;
- log dibaca dari CLI;
- dashboard minimum dibuat;
- evidence dikumpulkan;
- case incident ditulis;
- KPI dihitung;
- laporan akhir disusun;
- etika dijaga.

Mini SOC ini menjadi portofolio teknis nyata.

Ia membuktikan bahwa pembaca mampu menjalankan siklus SOC:

```
log
→ event
→ alert
→ triage
→ case
→ evidence
→ response
→ report
→ lesson learned
```

Output Bab

Setelah menyelesaikan Bab 10, pembaca memiliki:

- Mini SOC Ubuntu 26.04 siap operasi;
- Wazuh server aktif;
- Wazuh Agent aktif;
- FIM berjalan;
- minimal 3 deteksi diuji;
- dashboard minimum SOC;
- evidence lengkap;
- satu case incident;
- KPI Mini SOC;
- laporan akhir 5–10 halaman;
- manifest proyek;
- hash evidence;
- portofolio SOC nyata.

Kalimat kunci Bab 10:

> **SOC yang kuat bukan dibangun dari lisensi mahal, tetapi dari admin yang disiplin membaca log, memahami jaringan, menguji rule, menyimpan bukti, dan merespons dengan tanggung jawab.**

Lampiran A — Cheatsheet CLI Ubuntu SOC

Lampiran ini mengikuti arah dokumen **Instalasi & Konfigurasi Open Source SOC di Ubuntu 26.04, 2026**, khususnya bagian lampiran yang menempatkan `systemctl`, `journalctl`, `grep`, `awk`, `jq`, `curl`, `ss`, `tcpdump`, `ip`, `ufw`, `auditctl`, dan `ausearch` sebagai command inti admin SOC.

A.1 systemctl — Cek dan Kelola Service

Fungsi: melihat, menjalankan, menghentikan, restart, dan mengaktifkan service Linux.

```
sudo systemctl status ssh
sudo systemctl status wazuh-manager
sudo systemctl status wazuh-agent
sudo systemctl restart wazuh-agent
sudo systemctl enable --now ssh
```

Contoh penggunaan SOC:

- cek SSH aktif;
- cek Wazuh Manager;
- cek Wazuh Agent;
- restart service setelah konfigurasi berubah;
- melihat service gagal start.

A.2 journalctl — Membaca Log Systemd

Fungsi: membaca log service, kernel, dan sistem.

```
journalctl
journalctl -xe
journalctl -u ssh
journalctl -u wazuh-agent -f
journalctl --since "1 hour ago"
journalctl --since today
sudo journalctl -k
```

Contoh penggunaan SOC:

- membaca error service;
- membaca aktivitas SSH;
- membaca log Wazuh Agent;
- membatasi pencarian berdasarkan waktu;
- membaca log kernel.

A.3 grep — Mencari Pola di Log

Fungsi: mencari teks tertentu dalam file log.

```
sudo grep "Failed password" /var/log/auth.log
sudo grep "Accepted password" /var/log/auth.log
sudo grep "sudo" /var/log/auth.log
sudo grep "<IP_MENCURIGAKAN>"
/var/ossec/logs/alerts/alerts.json
```

Contoh penggunaan SOC:

- mencari login gagal;
- mencari login sukses;
- mencari aktivitas sudo;
- mencari IP mencurigakan di alert Wazuh;
- mencari IOC di log.

A.4 awk — Ambil Kolom Tertentu

Fungsi: memproses teks berbasis kolom.

```
sudo grep "Failed password" /var/log/auth.log | awk
'{print $(NF-3)}'
```

Contoh ringkasan IP login gagal:

```
sudo grep "Failed password" /var/log/auth.log \
| awk '{print $(NF-3)}' \
| sort \
| uniq -c \
| sort -nr
```

Catatan:

- posisi kolom IP bisa berbeda tergantung format log;
- selalu validasi manual.

A.5 jq — Membaca JSON

Fungsi: membaca, memfilter, dan merapikan file JSON.

```
sudo jq . /var/ossec/logs/alerts/alerts.json | less
sudo jq 'select(.rule.level >= 10)'
/var/ossec/logs/alerts/alerts.json
sudo jq '.agent.name, .rule.description, .rule.level'
/var/ossec/logs/alerts/alerts.json
```

Contoh Suricata:

```
sudo jq 'select(.event_type=="alert")'
/var/log/suricata/eve.json
sudo jq '.src_ip, .dest_ip, .alert.signature'
/var/log/suricata/eve.json
```

Untuk SOC modern, `jq` wajib dikuasai.

A.6 curl — Query API dan Service

Fungsi: mengakses API, dashboard backend, dan service berbasis HTTP/HTTPS dari CLI.

```
curl -k https://localhost:9200
curl -k -u admin:<password>
https://localhost:9200/_cat/indices?v
curl -k -u admin:<password>
https://localhost:9200/_cluster/health?pretty
```

Contoh Wazuh API:

```
TOKEN=$(curl -k -u wazuh:wazuh -X POST \
"https://localhost:55000/security/user/authenticate?raw=
true")
```

Wazuh API secara default mendengarkan pada port `55000/TCP`, dan dashboard Wazuh memakai API ini untuk menampilkan konfigurasi serta status server dan agent. (Dokumentasi Wazuh)

A.7 ss — Cek Port Aktif

Fungsi: melihat port TCP/UDP yang sedang listening.

```
sudo ss -tulpn
sudo ss -tulpn | grep ssh
sudo ss -tulpn | grep -E '1514|1515|55000|9200|443'
```

Makna opsi:

- `-t`: TCP;
- `-u`: UDP;
- `-l`: listening;
- `-p`: proses;
- `-n`: angka, tanpa resolve nama.

A.8 tcpdump — Capture Trafik

Fungsi: menangkap dan membaca trafik jaringan.

```
sudo tcpdump -i any -nn
sudo tcpdump -i any -nn port 22
sudo tcpdump -i any -nn host <IP_MENCURIGAKAN>
sudo tcpdump -i <interface> -w evidence.pcap host
<IP_TARGET>
tcpdump -nn -r evidence.pcap
```

Contoh evidence:

```
sudo tcpdump -i ens19 -c 200 -w
~/soc-lab/evidence/scan.pcap host 192.168.56.30
sha256sum ~/soc-lab/evidence/scan.pcap
```

A.9 ip — Cek Jaringan Linux

Fungsi: melihat IP address, route, dan interface.

```
ip a
ip route
ip link
```

Contoh SOC:

- cek IP SOC server;
- cek IP endpoint;
- cek route ke Wazuh Manager;
- cek interface sensor.

A.10 ufw — Firewall Host

Fungsi: mengatur firewall host Ubuntu.

```
sudo ufw status verbose
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow ssh
sudo ufw allow from 192.168.56.0/24 to any port 1514
sudo ufw allow from 192.168.56.0/24 to any port 1515
sudo ufw enable
```

Prinsip:

> **Buka port sesuai kebutuhan. Jangan buka semua port karena malas troubleshooting.**

A.11 auditctl — Rule Audit Linux

Fungsi: membuat dan melihat rule audit sistem.

```
sudo auditctl -s
sudo auditctl -l
sudo auditctl -w /etc/passwd -p wa -k passwd_change
sudo auditctl -w /etc/ssh/sshd_config -p wa -k
sshd_config_change
```

Makna:

- `-w`: watch file;
- `-p wa`: pantau write dan attribute change;
- `-k`: key agar mudah dicari.

A.12 ausearch — Cari Event Audit

Fungsi: mencari event audit berdasarkan tipe atau key.

```
sudo ausearch -m USER_LOGIN
sudo ausearch -m USER_AUTH
sudo ausearch -k passwd_change
sudo ausearch -k sshd_config_change
```

Contoh laporan file:

```
sudo aureport -f
```

Lampiran B — Port Penting SOC

Port harus dicatat sejak awal karena SOC melibatkan banyak service. Jangan membuka port ke semua jaringan. Gunakan prinsip **least privilege**: buka hanya dari subnet yang perlu.

Beberapa port Wazuh yang sering dipakai: API Wazuh pada `55000/TCP`, enrollment agent dapat memakai API pada `55000/TCP`, sedangkan deployment tradisional sering memakai port agent communication dan enrollment manager. Dokumentasi Wazuh menyebut API server Wazuh default di `55000/TCP`; dokumentasi enrollment juga menyebut `55000/TCP` untuk enrollment via Wazuh server API. (Dokumentasi Wazuh)

Layanan	Port Umum	Protokol	Fungsi	Catatan Keamanan
SSH	22	TCP	Administrasi server	Batasi dari IP admin
Wazuh agent communication	1514	TCP/UDP, tergantung konfigurasi	Agent mengirim event ke manager	Izinkan hanya dari subnet endpoint
Wazuh enrollment	1515	TCP	Pendaftaran agent tradisional	Buka hanya saat enrollment bila perlu
Wazuh API	55000	TCP	API manager, status, agent, auth token	Jangan dibuka publik
OpenSearch / Wazuh Indexer	9200	TCP	Query index dan cluster API	Batasi ke localhost/admin network
OpenSearch transport / indexer cluster	9300	TCP	Komunikasi antar node indexer	Hanya antar node tepercaya
Wazuh Dashboard	443 / 5601, sesuai setup	TCP	Dashboard SOC	Wajib TLS, batasi akses
Security Onion web	80 / 443	TCP	Web interface / Nginx default	Security Onion docs menyebut nginx ports 80 dan 443 default untuk analyst hostgroup (Dokumentasi Security Onion)
Suricata sensor	Tidak wajib port listening	-	Membaca trafik dari interface monitor	Sensor biasanya membaca trafik, bukan menerima koneksi
Zeek sensor	Tidak wajib port listening	-	Menghasilkan metadata jaringan	Fokus pada log, bukan port service
MISP web	80 / 443, sesuai instalasi	TCP	Threat intelligence platform	Dokumentasi resmi menyediakan install guide;

				gunakan TLS dan batasi akses (MISP Intelligence Platform)
DFIR-IRIS	443 default	TCP	Case management / incident response	Dokumentasi DFIR-IRIS menyebut akses HTTPS port 443 default (DFIR-IRIS Docs)
Shuffle	3001 / 3443 / sesuai compose	TCP	SOAR workflow	Cek file compose; jangan buka publik
Syslog	514	UDP/TCP	Terima log dari perangkat	Gunakan segmentasi dan sumber tepercaya
NTP	123	UDP	Sinkronisasi waktu	Penting untuk timeline incident
DNS	53	UDP/TCP	Resolusi nama	Log DNS penting untuk investigasi

Contoh UFW Minimum untuk Lab

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow from 192.168.56.0/24 to any port 22 proto tcp
sudo ufw allow from 192.168.56.0/24 to any port 1514
sudo ufw allow from 192.168.56.0/24 to any port 1515 proto
tcp
sudo ufw allow from 192.168.56.0/24 to any port 55000 proto
tcp
sudo ufw status verbose
```

Prinsip Port SOC

- **Jangan ekspos dashboard ke internet.**
- **Jangan buka OpenSearch `9200` ke jaringan luas.**
- **Batasi SSH dari IP admin.**
- **Buka port agent hanya dari subnet endpoint.**
- **Catat semua port aktif dengan `ss -tulpn`.**
- **Sinkronisasi waktu wajib untuk timeline incident.**

Lampiran C — Lokasi Log Penting

Lampiran ini merangkum lokasi log utama yang dipakai sepanjang buku. Dokumen terlampir juga menempatkan lokasi log Ubuntu, Wazuh, Suricata, dan Zeek sebagai komponen wajib lampiran.

C.1 Ubuntu

Lokasi	Fungsi	Command Cepat
`/var/log/auth.log`	Login, SSH, sudo, PAM	`sudo tail -f /var/log/auth.log`
`/var/log/syslog`	Log sistem umum	`sudo tail -f /var/log/syslog`
`journalctl`	Log systemd	`journalctl -xe`
`journalctl -u ssh`	Log service SSH	`journalctl -u ssh -f`
`journalctl -k`	Log kernel	`sudo journalctl -k`
`/var/log/audit/audit.log`	Audit Linux	`sudo tail -f /var/log/audit/audit.log`

Contoh pencarian:

```
sudo grep "Failed password" /var/log/auth.log
sudo grep "Accepted password" /var/log/auth.log
sudo grep "sudo" /var/log/auth.log
journalctl --since "1 hour ago"
```

C.2 Wazuh

Wazuh menyimpan alert dalam format teks dan JSON di direktori alert manager; dokumentasi Wazuh juga menyebut `/var/ossec/logs/alerts/alerts.log` dan `/var/ossec/logs/alerts/alerts.json` sebagai lokasi alert. (Dokumentasi Wazuh)

Lokasi	Fungsi	Command Cepat
`/var/ossec/logs/ossec.log`	Log operasional Wazuh	`sudo tail -f /var/ossec/logs/ossec.log`
`/var/ossec/logs/alerts/alerts.log`	Alert format teks	`sudo tail -f /var/ossec/logs/alerts/alerts.log`
`/var/ossec/logs/alerts/alerts.json`	Alert format JSON	`sudo tail -f /var/ossec/logs/alerts/alerts.json`
`/var/ossec/etc/ossec.conf`	Konfigurasi Wazuh / agent	`sudo less /var/ossec/etc/ossec.conf`
`/var/ossec/etc/rules/local_rules.xml`	Rule lokal	`sudo nano /var/ossec/etc/rules/local_rules.xml`
`/var/ossec/etc/decoders/local_decoder.xml`	Decoder lokal	`sudo nano /var/ossec/etc/decoders/local_decoder.xml`

Contoh filter alert:

```
sudo jq 'select(.rule.level >= 10)'
/var/ossec/logs/alerts/alerts.json
sudo jq '{time:.timestamp, agent:.agent.name,
level:.rule.level, rule:.rule.description}'
/var/ossec/logs/alerts/alerts.json
```

C.3 Suricata

Suricata EVE JSON dapat memuat alert, anomaly, metadata, informasi file, dan record protokol dalam format JSON. (GitHub)

Lokasi	Fungsi	Command Cepat
`/var/log/suricata/eve.json`	Output JSON utama Suricata	`sudo jq . /var/log/suricata/eve.json
`/var/log/suricata/fast.log`	Alert ringkas	`sudo tail -f /var/log/suricata/fast.log`
`/etc/suricata/suricata.yaml`	Konfigurasi Suricata	`sudo less /etc/suricata/suricata.yaml`
`/var/lib/suricata/rules/`	Rule Suricata	`ls -lah /var/lib/suricata/rules/`

Contoh filter:

```
sudo jq 'select(.event_type=="alert")'
/var/log/suricata/eve.json
sudo jq 'select(.event_type=="alert") |
{src:.src_ip,dst:.dest_ip,signature:.alert.signature,severity
:.alert.severity}' /var/log/suricata/eve.json
```

C.4 Zeek

Zeek menyediakan log koneksi, DNS, HTTP, TLS/SSL, file, dan notice yang penting untuk investigasi jaringan. (GitHub)

Log	Fungsi	Command Cepat
`conn.log`	Ringkasan koneksi	`cat conn.log
`dns.log`	Query dan jawaban DNS	`cat dns.log
`http.log`	Metadata HTTP	`cat http.log
`ssl.log`	Metadata TLS/SSL	`cat ssl.log
`files.log`	Metadata file lewat jaringan	`cat files.log
`notice.log`	Event penting Zeek	`cat notice.log

Lokasi umum:

```
ls -lah /opt/zeek/logs/current/
cd /opt/zeek/logs/current/
```

Cari IP:

```
grep "<IP_MENCURIGAKAN>" conn.log
grep "<IP_MENCURIGAKAN>" dns.log
grep "<IP_MENCURIGAKAN>" http.log
```

Lampiran D — Template Incident Report

Gunakan template ini untuk setiap alert yang naik menjadi incident.

D.1 Template Tabel Incident Report

Waktu	Sumber Alert	Host	User	Source IP	Destination IP	Rule	Severity	Bukti	Analisis	Tindakan	Status

D.2 Template Markdown

```
# Incident Report

## 1. Identitas Incident
- Incident ID:
- Judul:
- Tanggal:
- Analis:
- Severity:
- Status:

## 2. Ringkasan
Tuliskan ringkasan singkat incident.

## 3. Detail Alert
| Waktu | Sumber Alert | Host | User | Source IP |
Destination IP | Rule | Severity |
|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |

## 4. Evidence
| Bukti | Lokasi File | Hash SHA256 | Catatan |
|---|---|---|---|
| Alert JSON |  |  |  |
| Log CLI |  |  |  |
| Timeline |  |  |  |
| PCAP |  |  |  |
| Screenshot dashboard |  |  |  |

## 5. Analisis
- Apa yang terjadi?
- Kapan terjadi?
- Host mana terdampak?
- User apa yang terlibat?
- Source IP siapa?
```

```
- Apakah ada bukti tambahan?
- Apakah ada dampak?
- Apakah ini false positive?

## 6. Tindakan
| Waktu | Tindakan | Pelaksana | Approval | Hasil |
|---|---|---|---|---|
|  |  |  |  |  |

## 7. Status
- [ ] False positive
- [ ] Simulasi lab
- [ ] Monitoring lanjut
- [ ] Incident confirmed
- [ ] Closed

## 8. Rekomendasi
- Perbaikan konfigurasi:
- Perbaikan rule:
- Perbaikan monitoring:
- Perbaikan respons:

## 9. Lesson Learned
Tuliskan pelajaran utama dari incident ini.
```

D.3 Prinsip Incident Report

- **Jangan menulis klaim tanpa evidence.**
- **Timeline wajib ada.**
- **Hash evidence jika file penting.**
- **Bedakan alert, incident, dan response.**
- **Catat approval sebelum tindakan berisiko.**
- **Tuliskan status akhir dengan jelas.**

Lampiran E — Template Checklist Instalasi

Gunakan checklist ini sebelum masuk tahap operasional Mini SOC.

E.1 Checklist Instalasi Utama

No	Item	Status	Bukti / Command	Catatan
1	Ubuntu update selesai	[ ]	`sudo apt update && sudo apt full-upgrade -y`	
2	IP static aktif	[ ]	`ip a`, `ip route`	
3	SSH aman	[ ]	`sudo systemctl status ssh`	
4	Firewall aktif	[ ]	`sudo ufw status verbose`	
5	Wazuh server aktif	[ ]	`sudo systemctl status wazuh-manager`	
6	Wazuh indexer aktif	[ ]	`sudo systemctl status wazuh-indexer`	
7	Dashboard aktif	[ ]	`sudo systemctl status wazuh-dashboard`	
8	Agent connected	[ ]	`sudo /var/ossec/bin/agent_control -lc`	
9	Alert terbaca	[ ]	`sudo tail -f /var/ossec/logs/alerts/alerts.json`	
10	FIM aktif	[ ]	alert `syscheck` muncul	
11	Log CLI bisa diakses	[ ]	`journalctl`, `auth.log`, `alerts.json`	
12	Dashboard dapat diakses	[ ]	URL dashboard	
13	Backup konfigurasi dibuat	[ ]	`~/soc-lab/backup/`	
14	Evidence folder dibuat	[ ]	`~/soc-lab/evidence/`	
15	Hash evidence dibuat	[ ]	`sha256sum`	

E.2 Checklist Keamanan Minimum

No	Item	Status	Catatan
1	Root login langsung dimatikan	[ ]	
2	SSH dibatasi dari IP admin	[ ]	
3	Port dashboard tidak dibuka publik	[ ]	
4	OpenSearch tidak dibuka publik	[ ]	
5	Password default diganti	[ ]	
6	Token/API key tidak masuk laporan	[ ]	
7	Timezone benar	[ ]	`Asia/Jakarta`
8	Semua simulasi di lab sendiri	[ ]	
9	Tidak memakai malware asli	[ ]	
10	Tidak scan sistem publik	[ ]	

E.3 Checklist Setelah Reboot

```
sudo reboot
```

Setelah server hidup kembali:

Item	Command	Status
SSH bisa login	`ssh user@IP_SOC`	[ ]
Wazuh Manager aktif	`sudo systemctl status wazuh-manager`	[ ]
Indexer aktif	`sudo systemctl status wazuh-indexer`	[ ]
Dashboard aktif	`sudo systemctl status wazuh-dashboard`	[ ]
Agent aktif	`sudo /var/ossec/bin/agent_control -lc`	[ ]
Firewall aktif	`sudo ufw status verbose`	[ ]
Alert JSON terbaca	`sudo tail /var/ossec/logs/alerts/alerts.json`	[ ]

Lampiran F — 30 Command Wajib Admin SOC

Lampiran ini berisi command inti untuk cek service, cek port, cek log, cek agent, cek alert JSON, filter severity, cari IP, capture trafik, audit file, dan backup konfigurasi.

F.1 Daftar 30 Command

No	Command	Fungsi
1	`hostnamectl`	Cek hostname dan OS
2	`timedatectl`	Cek waktu dan timezone
3	`ip a`	Cek IP address
4	`ip route`	Cek routing
5	`ping -c 4 <IP>`	Tes konektivitas
6	`sudo ss -tulpn`	Cek port listening
7	`sudo ufw status verbose`	Cek firewall
8	`sudo systemctl status ssh`	Cek SSH
9	`sudo systemctl status wazuh-manager`	Cek Wazuh Manager
10	`sudo systemctl status wazuh-indexer`	Cek Wazuh Indexer
11	`sudo systemctl status wazuh-dashboard`	Cek Wazuh Dashboard
12	`sudo systemctl status wazuh-agent`	Cek Wazuh Agent
13	`journalctl -xe`	Cek error sistem terbaru
14	`journalctl -u wazuh-agent -f`	Pantau log agent real-time
15	`sudo tail -f /var/log/auth.log`	Pantau log autentikasi
16	`sudo grep "Failed password" /var/log/auth.log`	Cari login gagal
17	`sudo grep "Accepted password" /var/log/auth.log`	Cari login sukses
18	`sudo /var/ossec/bin/agent_control -l`	List agent Wazuh

19	`sudo /var/ossec/bin/agent_control -lc`	List agent aktif
20	`sudo tail -f /var/ossec/logs/ossec.log`	Pantau log Wazuh
21	`sudo tail -f /var/ossec/logs/alerts/alerts.json`	Pantau alert JSON
22	`sudo jq 'select(.rule.level >= 10)' /var/ossec/logs/alerts/alerts.json`	Filter high alert
23	`sudo jq '.agent.name, .rule.description, .rule.level' /var/ossec/logs/alerts/alerts.json`	Ambil field alert
24	`sudo grep "<IP>" /var/ossec/logs/alerts/alerts.json`	Cari IP di alert Wazuh
25	`sudo jq 'select(.event_type=="alert")' /var/log/suricata/eve.json`	Filter alert Suricata
26	`sudo tcpdump -i any -nn host <IP>`	Capture trafik IP
27	`sudo tcpdump -i <interface> -w evidence.pcap host <IP>`	Simpan PCAP evidence
28	`sudo auditctl -w /etc/passwd -p wa -k passwd_change`	Audit perubahan /etc/passwd
29	`sudo ausearch -k passwd_change`	Cari event audit file
30	`sudo cp -a /etc/netplan /etc/ssh /etc/ufw ~/soc-lab/backup/`	Backup konfigurasi awal

F.2 Command Paket Cepat untuk Triage SSH

```
mkdir -p ~/soc-lab/evidence/triage-ssh

sudo grep "Failed password" /var/log/auth.log \
| tee ~/soc-lab/evidence/triage-ssh/failed-password.log

sudo grep "Accepted password" /var/log/auth.log \
| tee ~/soc-lab/evidence/triage-ssh/accepted-password.log

sudo jq 'select(.rule.description |
test("SSH|password|authentication|login"; "i"))' \
/var/ossec/logs/alerts/alerts.json \
| tee ~/soc-lab/evidence/triage-ssh/wazuh-ssh-alerts.json
```

F.3 Command Paket Cepat untuk FIM

```
mkdir -p ~/soc-lab/evidence/fim

sudo jq 'select(.rule.groups[]? == "syscheck")' \
/var/ossec/logs/alerts/alerts.json \
| tee ~/soc-lab/evidence/fim/fim-alerts.json

sudo jq 'select(.syscheck.path? |
test("/etc/passwd|/etc/sudoers|/etc/ssh/sshd_config"))' \
/var/ossec/logs/alerts/alerts.json \
| tee ~/soc-lab/evidence/fim/fim-critical-files.json
```

F.4 Command Paket Cepat untuk Network Alert

```
mkdir -p ~/soc-lab/evidence/network

sudo jq 'select(.event_type=="alert")'
/var/log/suricata/eve.json \
| tee ~/soc-lab/evidence/network/suricata-alerts.json

cat /opt/zeek/logs/current/conn.log \
| zeek-cut id.orig_h id.resp_h id.resp_p proto service \
| tee ~/soc-lab/evidence/network/zeek-conn-summary.log
```

F.5 Command Paket Cepat untuk Hash Evidence

```
find ~/soc-lab/evidence -type f -exec sha256sum {} \; \
> ~/soc-lab/evidence/SHA256SUMS.txt

cat ~/soc-lab/evidence/SHA256SUMS.txt
```

F.6 Golden Rule Command Line SOC

- **Cek waktu dulu:** `timedatectl`
- **Cek service:** `systemctl status`
- **Cek port:** `ss -tulpn`
- **Cek log:** `journalctl`, `tail`, `grep`
- **Cek JSON:** `jq`
- **Cek agent:** `agent_control`
- **Cek jaringan:** `ip`, `tcpdump`
- **Cek audit:** `auditctl`, `ausearch`
- **Simpan evidence:** `tee`
- **Jaga integritas:** `sha256sum`

Kalimat kunci lampiran:

> **Admin SOC yang kuat bukan yang hafal semua command, tetapi yang tahu command mana dipakai untuk membuktikan kejadian, menyimpan evidence, dan membuat keputusan yang bisa diaudit.**

.

Glossary

Istilah	Definisi
Active Response	Tindakan otomatis atau semi-otomatis yang dijalankan saat alert tertentu muncul, misalnya menulis log, memblokir IP di lab, atau menjalankan script respons.
Agent	Program kecil yang dipasang di endpoint untuk mengirim log, status sistem, perubahan file, dan event keamanan ke SOC server.
Alert	Sinyal keamanan yang dihasilkan oleh rule, sensor, agent, atau sistem monitoring ketika ditemukan aktivitas mencurigakan.
Analyst Note	Catatan analis berisi interpretasi, dugaan awal, bukti pendukung, dan keputusan selama investigasi.
API	Antarmuka yang memungkinkan tool, script, atau sistem lain berkomunikasi dengan aplikasi secara terstruktur.
Artifact	Objek digital yang menjadi bukti atau hasil investigasi, seperti file log, hash, screenshot, PCAP, atau alert JSON.
Asset	Aset yang dipantau SOC, seperti server, endpoint, aplikasi, database, sensor, atau perangkat jaringan.
ATT&CK	Kerangka pengetahuan taktik dan teknik serangan yang digunakan untuk memetakan perilaku attacker dalam incident report.
Audit Trail	Jejak lengkap tindakan yang dilakukan, termasuk siapa melakukan apa, kapan dilakukan, alasan, dan hasilnya.
auditd	Service audit Linux untuk mencatat aktivitas penting sistem, seperti login, autentikasi, perubahan file, dan akses file sensitif.
auditctl	Command untuk membuat, melihat, dan mengatur rule audit pada Linux.
ausearch	Command untuk mencari event audit berdasarkan tipe event, key, user, waktu, atau parameter lain.
Baseline	Kondisi awal sistem yang dianggap normal dan dipakai sebagai pembanding saat terjadi perubahan atau anomali.
Brute Force	Percobaan login berulang dengan banyak kombinasi username atau password untuk menebak kredensial.
Case	Wadah investigasi yang berisi alert, timeline, evidence, observables, analyst note, task, respons, dan closure.

Case Management	Proses mengelola incident secara terstruktur dari triage, investigasi, task, evidence, sampai penutupan.
CIRCL	Organisasi yang banyak berkontribusi pada ekosistem threat intelligence terbuka, termasuk MISP dan PyMISP.
CLI	*Command Line Interface*, antarmuka berbasis teks untuk menjalankan perintah sistem, membaca log, dan mengelola SOC.
Cluster Health	Status kesehatan cluster OpenSearch atau indexer, biasanya ditandai dengan kondisi seperti green, yellow, atau red.
Correlation	Proses menghubungkan beberapa sumber log atau alert untuk membentuk gambaran kejadian yang lebih lengkap.
Credential Access	Taktik serangan untuk mencuri, menebak, atau mendapatkan kredensial akun.
curl	Command untuk mengakses URL, API, dan service HTTP/HTTPS dari terminal.
CVE	Identitas standar untuk kerentanan keamanan yang diketahui secara publik.
Dashboard	Tampilan visual untuk melihat alert, tren, severity, top agent, top source IP, dan status SOC.
Decoder	Komponen Wazuh yang membaca log mentah dan mengekstrak field penting seperti user, IP, action, atau status.
Detection Engineering	Praktik merancang, menguji, menulis, dan menyempurnakan rule deteksi agar sesuai dengan risiko organisasi.
DFIR	*Digital Forensics and Incident Response*, bidang yang fokus pada forensik digital dan respons insiden.
DFIR-IRIS	Platform open source untuk case management, investigasi, evidence, observables, dan kolaborasi incident response.
DNS Log	Log yang mencatat query DNS, jawaban DNS, domain yang ditanya, dan host yang melakukan query.
Docker	Platform container untuk menjalankan aplikasi dalam lingkungan terisolasi dan mudah direplikasi.
Endpoint	Mesin pengguna, server, atau host yang dipantau oleh SOC melalui agent atau log collection.
Enrollment	Proses pendaftaran agent ke Wazuh Manager agar agent dapat dipercaya dan mengirim data.

EVE JSON	Format output JSON utama Suricata yang berisi alert, flow, metadata, file, anomaly, dan event jaringan lain.
Evidence	Bukti teknis yang dikumpulkan selama investigasi, seperti log, alert JSON, PCAP, hash, timeline, dan screenshot.
False Positive	Alert yang muncul tetapi setelah dianalisis tidak menunjukkan ancaman nyata.
Field	Elemen data dalam log atau JSON, misalnya timestamp, source IP, destination IP, rule ID, severity, atau agent name.
FIM	*File Integrity Monitoring*, pemantauan perubahan file atau direktori penting seperti `/etc/passwd`, `/etc/sudoers`, dan konfigurasi SSH.
Flow	Ringkasan koneksi jaringan yang berisi IP sumber, IP tujuan, port, protokol, durasi, dan jumlah byte.
Forensics	Proses pemeriksaan bukti digital untuk memahami kejadian, dampak, dan urutan peristiwa.
Galaxy	Fitur MISP untuk memberi konteks lebih luas pada threat intelligence, seperti aktor, malware, campaign, atau teknik.
grep	Command untuk mencari pola teks dalam file log.
Hardening	Proses memperkuat keamanan sistem dengan menutup risiko dasar, mengurangi service tidak perlu, mengatur firewall, dan membatasi akses.
Hash	Nilai sidik jari digital file, misalnya SHA256, untuk membuktikan integritas evidence.
Host	Mesin atau sistem yang memiliki hostname dan IP address, seperti server SOC, endpoint, sensor, atau target lab.
Host Visibility	Kemampuan SOC melihat aktivitas pada endpoint atau server melalui agent, log, audit, dan FIM.
Hunt	Aktivitas pencarian aktif terhadap indikator atau pola serangan yang belum tentu memicu alert otomatis.
Incident	Kejadian keamanan yang sudah memiliki bukti, konteks, dampak, dan keputusan analis untuk ditangani secara formal.
Incident Handling	Proses menangani incident dari triage, investigasi, containment, eradication, recovery, hingga lesson learned.
Incident Report	Dokumen laporan insiden yang berisi timeline, sumber alert, host, user, IP, rule, severity, evidence, analisis, tindakan, dan status.

Index	Kumpulan dokumen log yang disimpan dalam OpenSearch agar dapat dicari, difilter, divisualisasikan, dan dijadikan alert.
Indexer	Komponen penyimpan dan pencari log, seperti Wazuh Indexer atau OpenSearch.
IOC	*Indicator of Compromise*, indikator teknis seperti IP, domain, URL, hash, email, atau filename yang terkait aktivitas mencurigakan.
IP Static	Alamat IP tetap yang tidak berubah, penting untuk SOC server, sensor, dan endpoint lab.
journalctl	Command untuk membaca log systemd, service, kernel, dan event sistem.
jq	Tool CLI untuk membaca, merapikan, memfilter, dan mengekstrak data JSON.
Kernel Log	Log dari kernel Linux, berguna untuk melihat driver, interface, disk, firewall kernel, dan error level rendah.
Kill Chain	Model tahapan serangan dari reconnaissance sampai tujuan akhir attacker.
Log Collection	Proses mengumpulkan log dari endpoint, server, aplikasi, sensor, atau perangkat jaringan.
Log Management	Pengelolaan log dari pengumpulan, penyimpanan, pencarian, retensi, hingga analisis.
Log Mentah	Log asli sebelum diproses, difilter, atau divisualisasikan.
Logtest	Tool Wazuh untuk menguji apakah decoder dan rule cocok terhadap contoh log tertentu.
Mapping	Proses menghubungkan alert atau event ke field, rule, teknik, taktik, atau kategori tertentu.
Metadata	Informasi ringkas tentang aktivitas, seperti DNS query, HTTP host, URI, TLS SNI, port, durasi koneksi, dan byte.
Mini SOC	Lingkungan SOC kecil untuk lab atau portofolio yang terdiri dari SOC server, endpoint agent, target lab, rule, dashboard, evidence, dan laporan.
MISP	Platform open source untuk mengelola threat intelligence, IOC, event, attribute, tag, galaxy, dan feed.
MITRE ATT&CK	Kerangka kerja taktik dan teknik serangan untuk memetakan perilaku attacker dalam deteksi dan laporan incident.

Monitor Interface	Interface sensor jaringan yang digunakan untuk membaca trafik, biasanya tanpa IP address.
MTTD	*Mean Time To Detect*, rata-rata waktu dari kejadian awal sampai alert atau deteksi muncul.
MTTR	*Mean Time To Respond*, rata-rata waktu dari deteksi sampai tindakan respons dilakukan.
Netplan	Sistem konfigurasi jaringan di Ubuntu berbasis file YAML, digunakan untuk mengatur IP static, gateway, dan DNS.
Network Sensor	Mesin atau sistem yang membaca trafik jaringan untuk menghasilkan alert, flow, metadata, dan PCAP.
Network Visibility	Kemampuan SOC melihat aktivitas jaringan melalui Suricata, Zeek, tcpdump, flow, dan PCAP.
OpenSearch	Platform open source untuk menyimpan, mencari, menganalisis, dan memvisualisasikan log.
OpenSearch Dashboards	Antarmuka visual untuk membuat dashboard, query, panel, dan eksplorasi data dari OpenSearch.
OSSEC	Fondasi historis dari Wazuh; banyak direktori Wazuh masih memakai path `/var/ossec`.
PCAP	File hasil capture paket jaringan mentah, biasanya dibuat dengan `tcpdump`, digunakan sebagai evidence jaringan.
Playbook	Langkah operasional tertulis untuk menangani alert atau incident tertentu secara konsisten.
Port Scan	Aktivitas memeriksa port terbuka pada host target, sering dipakai untuk reconnaissance.
Promiscuous Mode	Mode interface jaringan agar dapat membaca paket yang lewat, termasuk trafik yang bukan ditujukan langsung ke interface tersebut.
PyMISP	Library Python untuk mengakses dan mengotomasi MISP melalui API.
Query	Perintah pencarian terhadap log atau index untuk menemukan event tertentu.
Rule	Logika deteksi yang menentukan kapan event menjadi alert.
Rule ID	Nomor unik untuk mengidentifikasi rule deteksi.
Security Onion	Platform open source untuk network visibility, intrusion detection, log management, hunting, dashboard, dan case management.

Sensor Deployment	Desain penempatan sensor, termasuk management interface, monitoring interface, mirror port, dan alur trafik.
Severity	Tingkat prioritas atau tingkat risiko alert, biasanya digunakan untuk menentukan prioritas triage.
SHA256	Algoritma hash yang umum dipakai untuk membuktikan integritas file evidence.
Shuffle	Platform SOAR open source untuk membuat workflow otomasi berbasis trigger, parsing, enrichment, case creation, notifikasi, approval, dan audit trail.
SIEM	*Security Information and Event Management*, sistem untuk mengumpulkan, mengindeks, menganalisis, dan memberi alert dari event keamanan.
SOAR	*Security Orchestration, Automation, and Response*, otomasi workflow keamanan dengan kontrol, enrichment, approval, dan audit trail.
SOC	*Security Operations Center*, fungsi atau tim yang memantau, mendeteksi, menganalisis, merespons, dan melaporkan incident keamanan.
Source IP	Alamat IP asal aktivitas atau koneksi.
SPAN Port	Port switch yang mengirim salinan trafik ke sensor jaringan.
SSH	Protokol administrasi jarak jauh yang sering dipantau karena menjadi target login gagal, brute force, dan akses admin.
Suricata	Mesin IDS/IPS open source untuk deteksi trafik jaringan berbasis rule dan output JSON.
Syscheck	Komponen Wazuh yang menjalankan File Integrity Monitoring.
Syscollector	Komponen Wazuh Agent yang mengumpulkan inventaris sistem seperti paket, proses, port, dan informasi OS.
syslog	Sistem logging Linux yang menyimpan log umum seperti `/var/log/syslog`.
systemctl	Command untuk mengelola service systemd seperti start, stop, restart, enable, dan status.
Tactic	Tujuan besar attacker dalam MITRE ATT&CK, misalnya Credential Access, Discovery, atau Persistence.
Tag	Label konteks untuk event, IOC, alert, atau case agar mudah dikategorikan.

tcpdump	Tool CLI untuk menangkap dan membaca paket jaringan.
Technique	Cara teknis attacker mencapai taktik tertentu, misalnya Brute Force atau Network Service Discovery.
TheHive	Platform case management open source untuk SOC, CERT, dan CSIRT.
Threat Intelligence	Informasi tentang ancaman, IOC, aktor, teknik, malware, campaign, atau infrastruktur attacker.
Timeline	Urutan waktu kejadian yang menghubungkan alert, log, evidence, tindakan, dan keputusan analis.
Triage	Proses memilah dan menilai alert untuk menentukan apakah alert menjadi incident atau false positive.
UFW	*Uncomplicated Firewall*, tool firewall host pada Ubuntu untuk mengatur aturan akses jaringan.
Vulnerability Detection	Kemampuan mendeteksi kerentanan berdasarkan inventaris paket, versi software, CVE, dan severity.
Wazuh	Platform open source SIEM/XDR untuk endpoint monitoring, log analysis, FIM, vulnerability detection, rules, decoder, alert, dan API.
Wazuh Agent	Agent endpoint yang mengirim log, event, FIM, inventory, dan telemetry ke Wazuh Manager.
Wazuh API	API Wazuh untuk autentikasi, query agent, query manager, integrasi, dan otomasi.
Wazuh Dashboard	Dashboard untuk melihat agent, alert, inventory, FIM, vulnerability, dan status Wazuh.
Wazuh Indexer	Komponen indexer berbasis OpenSearch untuk menyimpan dan mencari alert Wazuh.
Wazuh Manager	Komponen utama Wazuh yang menerima data agent, menjalankan decoder/rule, menghasilkan alert, dan menyediakan API.
Wazuh Rule	Rule deteksi dalam Wazuh untuk mengubah event menjadi alert berdasarkan pola, field, frequency, timeframe, atau rule sebelumnya.
XDR	*Extended Detection and Response*, pendekatan deteksi dan respons lintas endpoint, jaringan, log, dan telemetry lain.
YAML	Format konfigurasi berbasis indentasi yang digunakan antara lain oleh Netplan.

YARA	Tool rule-based untuk mendeteksi atau mengklasifikasikan file berdasarkan pola string atau biner.
Zeek	Network security monitor open source yang menghasilkan metadata jaringan seperti `conn.log`, `dns.log`, `http.log`, `ssl.log`, dan `notice.log`.
zeek-cut	Tool untuk mengambil field tertentu dari log Zeek agar lebih mudah dianalisis dari CLI.

Referensi

Canonical. (2026). *Ubuntu 26.04 LTS release notes*. Canonical. https://documentation.ubuntu.com/release-notes/26.04/

Canonical. (2026). *Configuring networks*. Ubuntu Server Documentation. Canonical. https://ubuntu.com/server/docs/explanation/networking/configuring-networks/

DFIR-IRIS. (2026). *DFIR-IRIS documentation*. DFIR-IRIS Project. https://docs.dfir-iris.org/

DFIR-IRIS. (2026). *DFIR-IRIS: Collaborative incident response platform*. DFIR-IRIS Project. https://www.dfir-iris.org/

Docker Inc. (2026). *Docker Compose documentation*. Docker Documentation. https://docs.docker.com/compose/

Docker Inc. (2026). *Compose file reference*. Docker Documentation. https://docs.docker.com/reference/compose-file/

Free Software Foundation. (2026). *GNU awk user's guide*. GNU Project. https://www.gnu.org/software/gawk/manual/

Free Software Foundation. (2026). *GNU grep manual*. GNU Project. https://www.gnu.org/software/grep/manual/

Free Software Foundation. (2026). *GNU Wget manual*. GNU Project. https://www.gnu.org/software/wget/manual/

FreeDesktop.org. (2026). *journalctl*. systemd Manual Pages. https://www.freedesktop.org/software/systemd/man/journalctl.html

FreeDesktop.org. (2026). *systemd-journald.service*. systemd Manual Pages. https://www.freedesktop.org/software/systemd/man/systemd-journald.service.html

Git Development Community. (2026). *Git documentation*. Git SCM. https://git-scm.com/docs

Insecure.Com LLC. (2026). *Nmap reference guide*. Nmap Project. https://nmap.org/book/man.html

jq Project. (2026). *jq manual*. jq Project. https://jqlang.org/manual/

Lyon, G. "Fyodor". (2022). *Nmap network scanning: The official Nmap Project guide to network discovery and security scanning*. Nmap Software LLC. https://nmap.org/book/

MISP Project. (2026). *MISP OpenAPI specification*. MISP Project. https://www.misp-project.org/openapi/

MISP Project. (2026). *MISP core format*. MISP Project. https://www.misp-standard.org/rfc/misp-standard-core.html

Netplan Project. (2026). *Netplan documentation*. Canonical. https://netplan.readthedocs.io/

Netplan Project. (2026). *Netplan: Network configuration abstraction renderer*. Canonical. https://netplan.io/

Open Information Security Foundation. (2026). *Suricata user guide: Eve JSON output*. Suricata Documentation. https://docs.suricata.io/en/latest/output/eve/eve-json-output.html

Open Information Security Foundation. (2026). *Suricata user guide: Eve JSON format*. Suricata Documentation. https://docs.suricata.io/en/latest/output/eve/eve-json-format.html

OpenSearch Project. (2026). *Alerting API*. OpenSearch Documentation. https://docs.opensearch.org/latest/observing-your-data/alerting/api/

OpenSearch Project. (2026). *Alerting*. OpenSearch Documentation. https://docs.opensearch.org/latest/observing-your-data/alerting/

OpenSearch Project. (2026). *API reference*. OpenSearch Documentation. https://docs.opensearch.org/latest/api-reference/

OpenSearch Project. (2026). *OpenSearch Dashboards*. OpenSearch Documentation. https://docs.opensearch.org/latest/dashboards/

OpenSearch Project. (2026). *OpenSearch documentation*. OpenSearch Project. https://docs.opensearch.org/

Purbo, O. W. (2026). *Instalasi & konfigurasi open source SOC di Ubuntu 26.04*. Institut Teknologi Tangerang Selatan / OnnoCenter.

Purbo, O. W. (2026). *JSON mastery untuk akses API & log data Wazuh: Panduan praktis mahasiswa*. Institut Teknologi Tangerang Selatan / OnnoCenter.

Purbo, O. W. (2026). *Open source SOC mastery: Membangun Security Operations Center modern dengan tools gratis*. Institut Teknologi Tangerang Selatan / OnnoCenter.

Security Onion Solutions. (2026). *Security Onion documentation*. Security Onion Solutions. https://docs.securityonion.net/

Security Onion Solutions. (2026). *Security Onion 2.4 introduction*. Security Onion Solutions. https://docs.securityonion.net/en/2.4/introduction.html

Security Onion Solutions. (2026). *Zeek*. Security Onion Documentation. https://docs.securityonion.net/en/2.4/zeek.html

Shuffle. (2026). *Shuffle documentation*. Shuffle Automation. https://shuffler.io/docs

Shuffle. (2026). *Shuffle: The open source SOAR for all purposes*. Shuffle Automation. https://shuffler.io/docs

Shuffle Project. (2026). *Shuffle: A general purpose security automation platform*. GitHub. https://github.com/shuffle/shuffle

StrangeBee. (2026). *TheHive 5 documentation*. StrangeBee. https://docs.strangebee.com/

StrangeBee. (2026). *TheHive: Collaborative security case management platform*. StrangeBee. https://strangebee.com/

Tcpdump Group. (2026). *tcpdump manual page*. tcpdump & libpcap Project. https://www.tcpdump.org/manpages/tcpdump.1.html

The Audit Project. (2026). *auditd manual page*. Linux manual pages. https://man7.org/linux/man-pages/man8/auditd.8.html

The Audit Project. (2026). *auditctl manual page*. Linux manual pages. https://man7.org/linux/man-pages/man8/auditctl.8.html

The Audit Project. (2026). *ausearch manual page*. Linux manual pages. https://man7.org/linux/man-pages/man8/ausearch.8.html

The Open Group. (2024). *The Open Group base specifications issue 8*. The Open Group. https://pubs.opengroup.org/onlinepubs/9799919799/

Ubuntu Documentation Team. (2026). *Ubuntu Server documentation*. Canonical. https://documentation.ubuntu.com/server/

Ubuntu Documentation Team. (2026). *Ubuntu Server: OpenSSH server*. Canonical. https://documentation.ubuntu.com/server/how-to/security/openssh-server/

Ubuntu Documentation Team. (2026). *Ubuntu Server: Firewalls*. Canonical. https://documentation.ubuntu.com/server/how-to/security/firewalls/

VirusTotal. (2026). *What is YARA?* VirusTotal Documentation. https://docs.virustotal.com/docs/what-is-yara

Wazuh, Inc. (2026). *Active response*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/capabilities/active-response/index.html

Wazuh, Inc. (2026). *Architecture*. Wazuh Documentation. https://documentation.wazuh.com/current/getting-started/architecture.html

Wazuh, Inc. (2026). *Custom decoders*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/ruleset/decoders/custom.html

Wazuh, Inc. (2026). *Custom rules*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/ruleset/rules/custom.html

Wazuh, Inc. (2026). *File integrity monitoring*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/capabilities/file-integrity/index.html

Wazuh, Inc. (2026). *Installing the Wazuh agent on Linux endpoints*. Wazuh Documentation. https://documentation.wazuh.com/current/installation-guide/wazuh-agent/wazuh-agent-package-linux.html

Wazuh, Inc. (2026). *Quickstart*. Wazuh Documentation. https://documentation.wazuh.com/current/quickstart.html

Wazuh, Inc. (2026). *Testing decoders and rules*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/ruleset/testing.html

Wazuh, Inc. (2026). *Vulnerability detection: How it works*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/capabilities/vulnerability-detection/how-it-works.html

Wazuh, Inc. (2026). *Wazuh API getting started*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/api/getting-started.html

Wazuh, Inc. (2026). *Wazuh indexer indices*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/wazuh-indexer/wazuh-indexer-indices.html

Wazuh, Inc. (2026). *Wazuh server*. Wazuh Documentation. https://documentation.wazuh.com/current/getting-started/components/wazuh-server.html

YARA Project. (2026). *YARA documentation*. Read the Docs. https://yara.readthedocs.io/

Zeek Project. (2026). *Book of Zeek*. Zeek Documentation. https://docs.zeek.org/

Zeek Project. (2026). *conn.log*. Zeek Documentation. https://docs.zeek.org/en/master/reference/logs/conn.html

Zeek Project. (2026). *DNS log*. Zeek Documentation. https://docs.zeek.org/en/master/reference/logs/dns.html

Zeek Project. (2026). *Zeek logs*. Zeek Documentation. https://docs.zeek.org/en/lts/logs/index.html

Tentang Penulis



Onno W. Purbo, saat ini bertugas sebagai rektor di Institut Teknologi Tangerang Selatan (ITTS). Onno memperoleh gelar Ph.D bidang Electrical Engineering dari University of Waterloo, Canada, adalah seorang copyleft, educator dan ICT evangelist. Dia sudah mempublikasikan 50++ buku, termasuk free ICT ebook untuk sekolah tahun 2008. Beberapa buku terakhirnya adalah "Internet-TCP/IP: Konsep Dan Implementasi", 2018; "Sistem Operasi, Konsep Dan Membuat Linux OpenWRT Dan ROM Android", 2019; "IPv6 Untuk Mendukung Operasi Jaringan Dan Domain Name System", 2019; "Kubernetes untuk Pemula", 2024 dan "Membuat Operator Seluler 5G Sendiri", 2024.

Dia memimpin sambungan pertama Internet di Institut Teknologi Bandung, tahun 1993-2000, dan menggunakannya untuk membuat jaringan Internet pendidikan yang pertama di Indonesia. Dia membebaskan frekuensi WiFi, memperkenalkan RT/RW-net, antenna Wajanbolic dan jaringan selular OpenBTS dan private 5G sendiri. Dia memimpin jaringan telepon pertama di atas Internet, VoIP Merdeka, yang kemudian hari dikenal sebagai VoIP Rakyat berbasis SIP dan menggunakan kode area +62520 dan +62521. Dia saat ini aktif memperkenalkan e-Learning, dan menjalankan server e-Learning di http://lms.onnocenter.or.id/moodle/ 66,000++ siswa / mahasiswa dan https://opencourse.itts.ac.id 39.000+ mahasiswa secara gratis. Email: onno@indo.net.id