Open Source SOC Mastery:

Membangun Security Operations Center Modern dengan Tools Gratis

Onno W. Purbo
onno@itts.ac.id
onno@indo.net.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 sekadar kumpulan tools; SOC adalah disiplin membaca sinyal, membuktikan serangan, dan merespons dengan cepat, terukur, dan etis.

Kata Pengantar

Buku **Open Source SOC Mastery: Membangun Security Operations Center Modern dengan Tools Gratis, 2026** disusun untuk membantu mahasiswa memahami bagaimana sebuah *Security Operations Center* dapat dibangun secara bertahap menggunakan **tool open source**. Buku ini tidak hanya membahas konsep, tetapi juga mengajak pembaca membangun lab, membaca log, membuat deteksi, mengelola insiden, menggunakan *threat intelligence*, serta menyusun otomasi SOC yang aman dan terdokumentasi.

Fokus utama buku ini adalah membentuk cara berpikir analis SOC: **tajam membaca sinyal, disiplin memvalidasi bukti, hati-hati mengambil respons, dan rapi dalam dokumentasi**. Karena itu, pembahasan tidak berhenti pada instalasi tools seperti Wazuh, Security Onion, OpenSearch, MISP, DFIR-IRIS, dan Shuffle, tetapi juga menekankan alur kerja: dari event menjadi alert, dari alert menjadi case, dari case menjadi laporan, dan dari laporan menjadi pembelajaran.

Buku ini ditujukan bagi mahasiswa, dosen, praktisi pemula, dan siapa pun yang ingin memahami SOC secara praktis tanpa bergantung pada platform proprietary. Semua contoh dirancang untuk **lab tertutup, legal, dan aman**, sehingga pembaca dapat bereksperimen tanpa membahayakan sistem orang lain.

Akhirnya, semoga buku ini menjadi pijakan awal bagi lahirnya analis SOC Indonesia yang **kritis, etis, teknis, dan siap menjaga keamanan digital dengan pendekatan terbuka**.

Jakarta, Mei 2026
Onno W. Purbo

Executive Summary

Buku **Open Source SOC Mastery: Membangun Security Operations Center Modern dengan Tools Gratis, 2026** menegaskan bahwa keamanan siber modern tidak cukup dibangun dengan membeli alat, tetapi harus dibangun melalui **disiplin membaca data, membuktikan ancaman, merespons secara terukur, dan belajar dari setiap insiden**. Buku ini dirancang untuk mahasiswa agar mampu memahami dan membangun *Security Operations Center* berbasis **open source** secara bertahap: mulai dari lab Linux, Wazuh sebagai fondasi SIEM/XDR, deteksi *brute force*, FIM, vulnerability, monitoring jaringan dengan Security Onion, Suricata, dan Zeek, pengelolaan log dengan OpenSearch, case management dengan DFIR-IRIS, threat intelligence dengan MISP, hingga otomasi SOC dengan Shuffle. Fokus utama buku ini bukan sekadar instalasi tools, tetapi membentuk pola pikir analis SOC yang **kritis, etis, teknis, dan strategis**. Setiap alert harus dibaca sebagai sinyal, bukan vonis; setiap incident harus dibuktikan dengan log, JSON, dashboard, IOC, timeline, dan evidence; setiap respons harus dikendalikan dengan approval, audit trail, dan lesson learned. AI open source ditempatkan sebagai asisten analisis untuk merangkum alert, membantu triage, menyusun laporan, dan mempercepat pemahaman, tetapi tidak pernah menggantikan bukti teknis dan keputusan analis. Melalui proyek akhir Mini SOC, pembaca diarahkan menghasilkan portofolio nyata: arsitektur SOC, dashboard, rule deteksi, case incident, playbook automation, KPI sederhana, laporan, dan presentasi. Dengan pendekatan ini, buku ini menjadi panduan praktis dan strategis untuk melahirkan generasi analis SOC Indonesia yang tidak hanya mampu mengoperasikan tools, tetapi juga mampu berpikir tajam, bertindak bertanggung jawab, dan membangun pertahanan digital yang terbuka, mandiri, serta dapat dipertanggungjawabkan.

Daftar Isi

Kata Pengantar **4**
Executive Summary **5**
Daftar Isi **6**
Bab 1 — Mengapa Perlu Membangun SOC Open Source **21**
1.1 Arah Bab 21
1.2 Apa Itu Security Operations Center? 22
1.3 Mengapa SOC Penting? 23
1.4 SOC Bukan Sekadar Tools 24
1.5 Mengapa Open Source Penting? 25
1.6 Tantangan SOC Modern 26
1. Biaya Tinggi 26
2. Skill Gap 26
3. Volume Log 26
4. Alert Fatigue 27
5. Serangan Makin Cepat 27
1.7 Peran AI dalam SOC Open Source 28
1.8 Perbedaan Istilah Penting dalam SOC 29
SIEM 29
XDR 29
EDR 29
IDS/IPS 30
SOAR 30
Threat Intelligence 30
1.9 Peta Tools Open Source SOC 31
Wazuh — Fondasi SIEM/XDR 31
Security Onion — Visibilitas Jaringan 31
Suricata dan Zeek — Deteksi dan Analisis Trafik 32
OpenSearch — Mesin Pencarian dan Analitik Log 32
DFIR-IRIS atau TheHive/Cortex — Case Management 33
MISP — Threat Intelligence 33
Shuffle — SOAR Open Source 33
1.10 Cara Berpikir Analis SOC 34
1.11 Etika Lab SOC 34
1.12 Mini Hands-on 1 — Membuat Peta Arsitektur SOC Sederhana 35
Komponen Minimum 35
Alur Log 36
Pertanyaan Refleksi 36
1.13 Mini Hands-on 2 — Menentukan Kebutuhan Minimum Lab 37
Kebutuhan Minimal 37
VM Minimum 37
1.14 Mini Hands-on 3 — Checklist Awal SOC Lab 38
Checklist Infrastruktur 38

Checklist Data..... 38
Checklist Etika..... 38
1.15 Rangkuman Bab..... 39
1.16 Output Bab..... 39
Bab 2 — Membangun SOC Lab: Linux, VM, Docker, dan Jaringan Lokal..... 40
2.1 Arah Bab..... 40
2.2 Mengapa Lab SOC Harus Dibangun Bertahap?..... 41
2.3 Desain Minimum Lab SOC..... 42
Komponen Minimum..... 42
2.4 Arsitektur Sederhana Lab SOC..... 44
2.5 Pilihan Platform Virtualisasi Open Source..... 45
KVM/libvirt..... 45
Proxmox VE..... 45
Docker dan Docker Compose..... 46
2.6 Segmentasi Jaringan Lab..... 47
NAT..... 47
Host-only / Internal Network..... 47
Bridge..... 47
Isolasi Trafik..... 48
2.7 Prinsip Keamanan Lab..... 49
1. Snapshot Sebelum Eksperimen..... 49
2. Backup Konfigurasi..... 49
3. Dashboard Jangan Diekspos ke Internet..... 50
4. Password Kuat dan Akses Terbatas..... 50
5. Dokumentasi Perubahan..... 50
2.8 Spesifikasi Minimum Lab..... 51
Minimum Pribadi..... 51
Ideal untuk Mahasiswa Serius..... 51
Ideal untuk Lab Kampus..... 51
2.9 Struktur Folder Proyek SOC..... 52
2.10 Hands-on 1 — Instalasi Ubuntu Server untuk SOC Lab..... 53
Tujuan..... 53
Spesifikasi VM SOC Server..... 53
Langkah Ringkas..... 53
Perintah Awal Setelah Instalasi..... 54
2.11 Hands-on 2 — Instalasi Docker dan Docker Compose..... 55
Tujuan..... 55
Perintah Instalasi Ringkas..... 55
2.12 Hands-on 3 — Membuat Struktur Folder SOC Lab..... 57
2.13 Hands-on 4 — Membuat Inventaris Aset Lab dalam JSON..... 58
2.14 Hands-on 5 — Membuat Diagram Alur Log..... 60
2.15 Checklist Keamanan Sebelum Lanjut Bab 3..... 61
Infrastruktur..... 61
Docker..... 61

Dokumentasi..... 61
Keamanan..... 61
2.16 Kesalahan Umum Mahasiswa..... 62
2.17 Peran AI Open Source dalam Lab SOC..... 62
2.18 Rangkuman Bab..... 63
2.19 Output Bab..... 63
Bab 3 — Wazuh sebagai Fondasi SIEM/XDR Open Source..... 64
3.1 Arah Bab..... 64
3.2 Mengapa Wazuh Menjadi Fondasi SOC?..... 65
3.3 Arsitektur Wazuh..... 66
1. Wazuh Agent..... 66
2. Wazuh Server..... 66
3. Wazuh Indexer..... 67
4. Wazuh Dashboard..... 67
3.4 Alur Data Wazuh..... 68
3.5 Konsep Inti: Log, Event, Alert, Rule, Decoder, Group, Agent..... 69
Log..... 69
Event..... 69
Alert..... 69
Decoder..... 70
Rule..... 70
Group..... 70
Agent..... 71
3.6 Fungsi Utama Wazuh..... 71
1. Log Collection..... 71
2. File Integrity Monitoring..... 71
3. Vulnerability Detection..... 72
4. Malware Detection..... 72
5. Active Response..... 72
6. Compliance..... 73
3.7 Wazuh sebagai Pondasi AI untuk SOC..... 74
3.8 Desain Lab Wazuh Single-Node..... 75
Komponen..... 75
Alur Lab..... 75
Kelebihan Single-Node..... 75
Keterbatasan Single-Node..... 75
3.9 Hands-on 1 — Instalasi Wazuh Server Single-Node..... 76
Tujuan..... 76
Prasyarat..... 76
Perintah Instalasi Cepat..... 76
3.10 Hands-on 2 — Cek Status Service Wazuh..... 78
3.11 Hands-on 3 — Menambahkan Endpoint Linux..... 79
Tujuan..... 79
Langkah Umum..... 79

Perintah Cek Agent..... 79
3.12 Membaca Agent di Dashboard.....80
3.13 Hands-on 4 — Mengaktifkan FIM untuk File Penting.....81
Tujuan.....81
Konsep..... 81
Contoh Konfigurasi FIM.....81
3.14 Membaca Alert dalam Format JSON..... 83
3.15 Hands-on 5 — Membuat Custom Rule Sederhana.....84
Tujuan.....84
Lokasi Rule Lokal..... 84
Contoh Rule Sederhana.....84
Uji Rule.....85
3.16 Hands-on 6 — Menguji Rule dengan Wazuh Logtest..... 86
3.17 Dasar Wazuh API.....87
Prinsip Keamanan API..... 87
3.18 Hands-on 7 — Mengambil Token API dengan curl..... 88
3.19 Hands-on 8 — Mengakses Wazuh API dengan Python.....89
3.20 Membaca Alert untuk Analisis SOC..... 90
Checklist Triage Alert..... 90
3.21 Kesalahan Umum Saat Belajar Wazuh..... 91
3.22 Mini Template Laporan Praktikum Bab 3.....92
3.23 Rangkuman Bab.....93
3.24 Output Bab..... 93
Bab 4 — Deteksi Serangan dengan Wazuh: Brute Force, Malware, FIM, dan Vulnerability.....94
4.1 Arah Bab..... 94
4.2 Mengapa Deteksi Lebih Penting daripada Dashboard.....94
4.3 Dari Event ke Alert ke Incident..... 95
Event..... 95
Alert..... 95
Incident.....96
4.4 Cara Kerja Deteksi Berbasis Rule.....96
4.5 File Integrity Monitoring sebagai Deteksi Dasar.....98
4.6 Deteksi Brute Force.....99
Indikator yang Dicari.....99
Pertanyaan Analis..... 99
4.7 Deteksi Malware Sederhana dengan YARA dan FIM..... 100
Contoh Folder Latihan.....100
Prinsip Aman Deteksi Malware..... 100
4.8 Vulnerability Detection.....101
Cara Membaca Hasil.....101
4.9 Active Response: Potensi dan Risiko.....102
Risiko Active Response.....102
4.10 Mapping Alert ke MITRE ATT&CK..... 103

4.11 Peran AI Open Source dalam Deteksi Wazuh....104
4.12 Hands-on 1 — Simulasi Login Gagal di Lab Tertutup.... 105
Tujuan....105
Lingkungan....105
Langkah Aman.... 105
Yang Dicatat.... 105
Pertanyaan Analisis....105
4.13 Hands-on 2 — Membaca Alert Brute Force di Wazuh.... 106
4.14 Hands-on 3 — Custom Rule Login Gagal....107
Tujuan....107
4.15 Hands-on 4 — Mengaktifkan FIM untuk File Penting....108
Tujuan.... 108
4.16 Hands-on 5 — Deteksi File Dummy dengan YARA.... 109
Tujuan....109
Buat Folder Latihan.... 109
Buat File Dummy....109
Buat Rule YARA Sederhana....110
Catatan Keamanan....110
4.17 Hands-on 6 — Mengaktifkan Vulnerability Detection....111
Tujuan.... 111
4.18 Hands-on 7 — Active Response Sederhana di Lab....112
Tujuan....112
4.19 Hands-on 8 — Ekspor Alert JSON untuk Laporan.... 113
Tujuan....113
4.20 Template Analisis Alert.... 114
4.21 Kesalahan Umum dalam Deteksi.... 115
4.22 Peran AI Lokal untuk Membantu Bab Ini....116
4.23 Rangkuman Bab....117
4.24 Output Bab.... 117
Bab 5 — Network Security Monitoring dengan Security Onion, Suricata, dan Zeek...118
5.1 Arah Bab....118
5.2 Mengapa Log Endpoint Saja Tidak Cukup?.... 118
5.3 Konsep Network Security Monitoring.... 120
5.4 NIDS dan NIPS.... 121
NIDS....121
NIPS.... 121
5.5 Peran Security Onion.... 122
5.6 Peran Suricata....123
Konsep Rule Suricata....123
Eve JSON....124
5.7 Peran Zeek....125
Log Penting Zeek.... 125
Mengapa Zeek Penting?.... 126
5.8 Sensor Interface dan Traffic Mirroring....127

Management Interface..... 127
Monitor Interface..... 127
5.9 PCAP, Alert, Flow, dan Pivot Investigasi..... 128
Alert..... 128
Flow..... 128
PCAP..... 128
Pivot Investigasi..... 129
5.10 Hubungan Security Onion, Suricata, Zeek, Wazuh, dan OpenSearch..... 130
Wazuh..... 130
Security Onion..... 130
Suricata..... 130
Zeek..... 130
OpenSearch..... 131
5.11 Peran AI Open Source dalam Network Monitoring..... 132
5.12 Hands-on 1 — Desain VM Security Onion..... 133
Tujuan..... 133
Spesifikasi Awal..... 133
Topologi Sederhana..... 133
Catatan..... 133
5.13 Hands-on 2 — Instalasi Security Onion di VM..... 134
Tujuan..... 134
Langkah Ringkas..... 134
Validasi Awal..... 134
5.14 Hands-on 3 — Simulasi Port Scanning di Lab..... 135
Tujuan..... 135
Skenario..... 135
Langkah Aman..... 135
Pertanyaan Analisis..... 135
5.15 Hands-on 4 — Membaca Alert Suricata..... 136
Pertanyaan Triage..... 136
5.16 Hands-on 5 — Membaca Log Zeek..... 137
Field Penting conn.log..... 137
5.17 Hands-on 6 — Membaca DNS dan HTTP Log..... 138
DNS Log..... 138
HTTP Log..... 138
5.18 Hands-on 7 — Mengambil Contoh Eve JSON..... 139
Tujuan..... 139
Analisis Singkat..... 139
5.19 Hands-on 8 — Catatan Investigasi Sederhana..... 140
5.20 Kesalahan Umum dalam Network Monitoring..... 141
5.21 Rangkuman Bab..... 141
5.22 Output Bab..... 142
Bab 6 — OpenSearch untuk Log Management, Query, Dashboard, dan Alert.....143
6.1 Arah Bab..... 143

6.2 Mengapa Log Perlu Diindeks? 143
6.3 OpenSearch dalam Arsitektur SOC 145
6.4 Struktur Data Log SOC 146
1. timestamp 146
2. host 146
3. source_ip 147
4. destination_ip 147
5. rule_id 147
6. severity 147
7. event_type 148
8. mitre_tactic 148
9. mitre_technique 148
6.5 Konsep Index, Mapping, Field, dan Query 149
Index 149
Mapping 149
Field 150
Query 150
6.6 JSON sebagai Bahasa Pertukaran Data SOC 151
6.7 Normalisasi Field 152
6.8 Retensi Log dan Kapasitas Storage 153
6.9 OpenSearch Dashboard untuk SOC 154
6.10 Alert Rule Berbasis Query 155
6.11 Peran AI Open Source dalam Bab Ini 156
6.12 Hands-on 1 — Instalasi OpenSearch dan OpenSearch Dashboards 157
Tujuan 157
Pilihan Instalasi 157
Struktur Folder 157
Catatan Keamanan 157
6.13 Hands-on 2 — Mengirim Sample Log JSON ke OpenSearch 158
Tujuan 158
Contoh Log JSON 158
Kirim ke Index 158
6.14 Hands-on 3 — Membuat Index Pattern 159
Langkah 159
Validasi 159
6.15 Hands-on 4 — Query Dasar untuk SOC 160
Cari Alert Severity Tinggi 160
Cari Login Gagal 160
Cari Alert dari Host Tertentu 161
Cari Alert dari Source IP Tertentu 161
Cari Alert MITRE Tactic Tertentu 161
6.16 Hands-on 5 — Membuat Dashboard SOC Sederhana 162
Visualisasi 1 — Alert per Host 162
Visualisasi 2 — Alert per Severity 162

Visualisasi 3 — Top Source IP 162
Visualisasi 4 — Top Rule ID 163
Visualisasi 5 — Alert per MITRE Tactic 163
Prinsip Dashboard 163
6.17 Hands-on 6 — Membuat Alert Sederhana 164
Skenario 164
Konsep 164
Query Konseptual 164
Catatan 165
6.18 Hands-on 7 — Ekspor Hasil Query untuk Laporan Insiden 166
Tujuan 166
6.19 Kesalahan Umum dalam Log Management 167
6.20 Checklist Bab 6 167
6.21 Rangkuman Bab 168
6.22 Output Bab 168
Bab 7 — Incident Response dan Case Management dengan DFIR-IRIS atau TheHive/Cortex 169
7.1 Arah Bab 169
7.2 Mengapa Alert Tidak Sama dengan Incident? 170
7.3 Perbedaan Event, Alert, Incident, dan Case 171
Event 171
Alert 171
Incident 171
Case 172
7.4 Tahapan Incident Response 173
1. Preparation 173
2. Identification 173
3. Containment 174
4. Eradication 174
5. Recovery 174
6. Lessons Learned 175
7.5 Triage: Memilih Alert yang Layak Menjadi Case 176
1. Severity 176
2. Confidence 176
3. Asset Criticality 177
4. Impact 177
7.6 Case Management 178
Komponen Case 178
7.7 DFIR-IRIS sebagai Pilihan Utama Open Source 181
7.8 TheHive/Cortex sebagai Referensi Integrasi SOC 182
7.9 Alur Kerja Case dari Alert Wazuh 183
7.10 Peran AI Open Source dalam Case Management 184
7.11 Hands-on 1 — Instalasi DFIR-IRIS dengan Docker Compose 185
Tujuan 185

Prasyarat..... 185
Struktur Folder.....185
Instalasi..... 185
Validasi..... 186
7.12 Hands-on 2 — Membuat Case dari Alert Wazuh..... 187
Tujuan.....187
Sumber Alert..... 187
Data Minimum Case.....187
Prinsip Penulisan.....187
7.13 Hands-on 3 — Menambahkan Observables..... 188
Pertanyaan Analis..... 188
7.14 Hands-on 4 — Membuat Timeline Insiden..... 189
7.15 Hands-on 5 — Membuat Task untuk Analis..... 190
Pembagian Peran Mahasiswa.....190
7.16 Hands-on 6 — Menambahkan Evidence.....191
Format Evidence Ringkas..... 191
7.17 Hands-on 7 — Membuat Laporan Insiden Satu Halaman.....192
7.18 Hands-on 8 — Menutup Case dengan Status dan Rekomendasi.....193
7.19 Template Case Incident.....194
7.20 Kesalahan Umum dalam Incident Response..... 195
7.21 Checklist Bab 7..... 196
7.22 Rangkuman Bab.....196
7.23 Output Bab..... 197
Bab 8 — Threat Intelligence dengan MISP dan Mapping MITRE ATT&CK.....198
8.1 Arah Bab..... 198
8.2 Apa Itu Threat Intelligence?..... 199
8.3 Data, Informasi, dan Intelligence.....200
Data.....200
Informasi.....200
Intelligence..... 200
8.4 IOC vs TTP.....201
IOC — Indicator of Compromise..... 201
TTP — Tactics, Techniques, and Procedures..... 202
8.5 Jenis Threat Intelligence..... 203
1. Strategic Intelligence.....203
2. Operational Intelligence.....203
3. Tactical Intelligence..... 204
4. Technical Intelligence..... 204
8.6 Mengapa MISP?.....205
8.7 Konsep MISP: Event, Attribute, Tag, Galaxy, Feed.....206
Event..... 206
Attribute..... 206
Tag.....207
Galaxy..... 207

Feed..... 207
8.8 IOC dalam Format JSON..... 208
8.9 MITRE ATT&CK untuk Membaca Taktik dan Teknik..... 209
8.10 Cyber Kill Chain sebagai Model Alur Serangan..... 210
8.11 Etika Berbagi IOC..... 210
8.12 Peran AI Open Source dalam Threat Intelligence..... 211
8.13 Hands-on 1 — Instalasi MISP..... 212
Tujuan..... 212
Prasyarat..... 212
Langkah Ringkas..... 212
Catatan Keamanan..... 212
8.14 Hands-on 2 — Mengaktifkan Feed Threat Intelligence..... 213
Tujuan..... 213
Prinsip Kurasi Feed..... 213
8.15 Hands-on 3 — Membuat Event MISP Sederhana..... 214
Tujuan..... 214
Prinsip..... 214
8.16 Hands-on 4 — Menambahkan IOC..... 215
IP Address..... 215
Domain..... 215
Hash..... 215
URL..... 215
Catatan..... 215
8.17 Hands-on 5 — Ekspor IOC dalam JSON..... 216
Tujuan..... 216
8.18 Hands-on 6 — Menghubungkan IOC ke Case Incident..... 217
8.19 Hands-on 7 — Mapping Alert Wazuh ke MITRE ATT&CK..... 218
8.20 Hands-on 8 — Membuat Tabel IOC untuk Laporan..... 219
8.21 Template Prompt AI Lokal untuk IOC..... 220
8.22 Kesalahan Umum dalam Threat Intelligence..... 220
8.23 Checklist Bab 8..... 221
8.24 Rangkuman Bab..... 221
8.25 Output Bab..... 222
Bab 9 — SOAR dan Automation: Dari Alert ke Respons Terukur..... 223
9.1 Arah Bab..... 223
9.2 Apa Itu SOAR?..... 224
9.3 Mengapa SOC Membutuhkan Otomasi?..... 225
9.4 Script, Workflow, dan Playbook..... 226
Script..... 226
Workflow..... 226
Playbook..... 227
9.5 Konsep Dasar SOAR: Trigger, Condition, Action, Approval, Logging..... 228
1. Trigger..... 228
2. Condition..... 228

3. Action..... 229
4. Approval..... 229
5. Logging.....229
9.6 Kapan Respons Boleh Otomatis?..... 230
Prinsip Respons Otomatis Aman..... 230
9.7 Kapan Harus Menunggu Persetujuan Analis?..... 231
9.8 Risiko Otomasi SOC..... 232
1. False Positive.....232
2. Blokir Aset Penting.....232
3. Kehilangan Bukti..... 232
4. Respons Berlebihan.....232
5. Ketergantungan pada Tools..... 233
9.9 Shuffle SOAR dan Webhook.....234
9.10 n8n sebagai Alternatif Workflow Automation Umum.....235
9.11 Format JSON untuk Integrasi Tools SOC.....236
9.12 Playbook Dasar SOC.....237
Playbook 1 — Enrich IP..... 237
Playbook 2 — Cek Reputasi Domain.....237
Playbook 3 — Buat Case..... 238
Playbook 4 — Kirim Notifikasi..... 238
Playbook 5 — Blok IP di Lab.....239
9.13 Audit Trail Otomasi.....240
9.14 Peran AI Open Source dalam SOAR..... 241
9.15 Hands-on 1 — Instalasi Shuffle.....242
Tujuan.....242
Prasyarat.....242
Struktur Folder.....242
Prinsip Instalasi..... 242
9.16 Hands-on 2 — Membuat Webhook Penerima Alert JSON.....243
Tujuan.....243
Langkah Konseptual.....243
9.17 Hands-on 3 — Parsing JSON..... 244
Tujuan.....244
9.18 Hands-on 4 — Enrich IOC.....245
Tujuan.....245
9.19 Hands-on 5 — Membuat Case Otomatis..... 246
Tujuan.....246
9.20 Hands-on 6 — Membuat Playbook “SSH Brute Force Detected”..... 247
Struktur Playbook.....247
Prinsip..... 247
9.21 Hands-on 7 — Approval Sebelum Blok IP..... 248
Tujuan.....248
9.22 Hands-on 8 — Membuat Log Hasil Automation.....249
9.23 Kesalahan Umum dalam SOAR.....250

9.24 Checklist Bab 9 250
9.25 Rangkuman Bab 251
9.26 Output Bab 251
Bab 10 — Proyek Akhir: Mini SOC Berbasis Open Source 252
10.1 Arah Bab 252
10.2 Tujuan Proyek Akhir 253
10.3 Prinsip Proyek 254
1. Open Source First 254
2. Lab Tertutup 254
3. Bukti Lebih Penting daripada Klaim 255
4. AI sebagai Asisten, Bukan Pengganti Analis 255
10.4 Spesifikasi Minimum Proyek 256
Komponen Minimal 256
Komponen Wajib Minimum 256
Sumber Data Minimum 256
Output Minimum 257
10.5 Arsitektur Mini SOC 257
10.6 Dataset dan Skenario Uji 258
Dataset Minimum 258
Format Dataset 258
Prinsip Dataset 258
10.7 Skenario Uji Proyek 259
Skenario 1 — SSH Brute Force 259
Skenario 2 — Port Scan 259
Skenario 3 — Perubahan File Konfigurasi 260
Skenario 4 — Web Attack Sederhana di Lab 260
Skenario 5 — Aktivitas Script Mencurigakan di Endpoint Linux 261
10.8 Deteksi Minimal yang Harus Dibuat 262
Deteksi 1 — Login Gagal Berulang 262
Deteksi 2 — Perubahan File Penting 262
Deteksi 3 — Port Scan 262
Deteksi Tambahan Opsional 263
10.9 Dashboard Minimum 264
Contoh Panel 264
10.10 Case Incident Minimum 265
Isi Case 265
Contoh Case 265
10.11 Playbook Automation Minimum 266
Playbook: “SSH Brute Force Detected” 266
Syarat Aman 266
10.12 KPI SOC Sederhana 267
1. MTTD — Mean Time To Detect 267
2. MTTR — Mean Time To Respond 267
3. Jumlah Alert 267

4. False Positive..... 267
5. Severity..... 268
6. Incident Closed..... 268
10.13 Dokumentasi Proyek..... 269
Struktur Folder..... 269
Isi Minimal README..... 269
Dokumentasi yang Wajib Ada..... 270
10.14 Publikasi Aman ke GitHub..... 271
Yang Boleh Dipublikasikan..... 271
Yang Tidak Boleh Dipublikasikan..... 271
Checklist Sebelum Publish..... 271
10.15 Etika dan Legalitas..... 272
10.16 Roadmap Karier SOC Analyst..... 273
Kompetensi Teknis..... 273
Kompetensi Analitis..... 273
Kompetensi Komunikasi..... 273
10.17 Hands-on Proyek — Checklist Implementasi..... 274
Tahap 1 — Infrastruktur..... 274
Tahap 2 — Deploy Komponen SOC..... 274
Tahap 3 — Log Collection..... 274
Tahap 4 — Deteksi..... 274
Tahap 5 — Dashboard..... 275
Tahap 6 — Case Management..... 275
Tahap 7 — Automation..... 275
Tahap 8 — Laporan dan Presentasi..... 275
10.18 Format Laporan Akhir 5–10 Halaman..... 276
10.19 Format Presentasi 10 Menit..... 277
Menit 1 — Masalah..... 277
Menit 2 — Arsitektur..... 277
Menit 3 — Data..... 277
Menit 4 — Deteksi..... 277
Menit 5 — Dashboard..... 277
Menit 6 — Investigasi..... 277
Menit 7 — Respons..... 277
Menit 8 — KPI..... 278
Menit 9 — AI Open Source..... 278
Menit 10 — Lesson Learned..... 278
10.20 Rubrik Penilaian Proyek..... 279
Nilai Tambahan..... 279
10.21 Kesalahan Umum Proyek Akhir..... 280
10.22 Peran AI Open Source dalam Proyek Akhir..... 281
10.23 Rangkuman Bab..... 282
10.24 Output Bab..... 282
Lampiran A — Kebutuhan Hardware dan Software Lab..... 283

A.1 Spesifikasi Minimum Laptop 283
A.2 Spesifikasi Ideal Lab 284
A.3 Daftar VM 285
A.4 Estimasi RAM dan Storage 286
A.5 Rekomendasi OS 286
A.6 Tools Utama 287
Lampiran B — Cheatsheet Command Dasar SOC 288
B.1 Command Linux Dasar 288
B.2 Docker dan Docker Compose 289
B.3 Cek Service 289
B.4 Cek Log 290
B.5 Cek Port 290
B.6 Cek Koneksi 290
B.7 Cek Resource 291
B.8 Cek File Berubah 291
B.9 Query Sederhana curl 292
B.10 Query API JSON 292
Lampiran C — Template Alert JSON 293
Penjelasan Field 294
Lampiran D — Template Case Incident 295
Checklist Case 296
Lampiran E — Template Dashboard SOC 297
E.1 Panel Dashboard Minimum 297
E.2 Field yang Disarankan 297
E.3 Contoh Layout Dashboard 298
E.4 Query Dasar Dashboard 298
Alert per Severity 298
Top Source IP 298
Alert Brute Force 299
Alert FIM 299
E.5 Prinsip Dashboard yang Baik 299
Lampiran F — 30 Prompt AI untuk SOC Analyst 300
F.1 Ringkasan Alert 300
F.2 Triage Insiden 301
F.3 Analisis Log 301
F.4 Mapping MITRE ATT&CK 302
F.5 Pembuatan Laporan Manajemen 303
F.6 Pembuatan Timeline 303
F.7 Validasi IOC 304
F.8 Pembuatan Playbook 304
F.9 Deteksi False Positive 305
F.10 Lesson Learned 306
Catatan Akhir Lampiran 306
Glossary 307

Referensi.....315
Tentang Penulis.....319

Bab 1 — Mengapa Perlu Membangun SOC Open Source

> **SOC yang kuat bukan dibangun dari mahalnya alat, tetapi dari disiplin membaca sinyal, membuktikan serangan, dan merespons secara cepat, terukur, dan etis.**

1.1 Arah Bab

Bab ini membuka buku dengan satu pesan utama: **SOC bukan ruang dashboard, tetapi sistem kerja pertahanan siber**. Di dalam SOC, data dikumpulkan, log dibaca, pola dicari, alert divalidasi, insiden ditangani, lalu organisasi belajar agar serangan berikutnya lebih cepat dikenali.

Outline buku ini memang diarahkan untuk mahasiswa yang ingin membangun SOC berbasis **open source tools**, dengan jalur belajar dari konsep, lab, Wazuh, Security Onion, OpenSearch, incident response, threat intelligence, SOAR, sampai proyek akhir mini SOC.

Bab ini tidak langsung masuk ke instalasi. Sebelum memasang tools, mahasiswa perlu memahami **mengapa SOC dibangun**, **apa fungsi SOC**, dan **bagaimana cara berpikir seorang analis SOC**.

1.2 Apa Itu Security Operations Center?

Security Operations Center atau **SOC** adalah pusat koordinasi untuk **monitoring, deteksi, analisis, respons, dan perbaikan keamanan**. SOC bertugas menjaga agar organisasi dapat melihat tanda bahaya lebih awal, memahami apa yang sedang terjadi, lalu mengambil tindakan yang tepat.

SOC bekerja seperti ruang kendali. Bedanya, yang dipantau bukan mesin industri atau lalu lintas kota, tetapi **aset digital**, **endpoint**, **server**, **aplikasi**, **jaringan**, **akun pengguna**, dan **log aktivitas**.

Dalam rujukan open-source SOC, SOC dijelaskan sebagai garis depan pertahanan yang melakukan pemantauan berkelanjutan, identifikasi ancaman, dan respons insiden untuk melindungi aset digital dan jaringan organisasi.

Secara sederhana, SOC menjawab pertanyaan berikut:

- **Apa yang terjadi?**
- **Di mana terjadi?**
- **Siapa atau apa yang terlibat?**
- **Apakah ini normal atau mencurigakan?**
- **Apakah ini hanya alert biasa atau sudah menjadi incident?**
- **Apa bukti teknisnya?**
- **Apa tindakan yang aman dan proporsional?**

SOC bukan tempat untuk panik. SOC adalah tempat untuk **berpikir jernih dengan bukti**.

1.3 Mengapa SOC Penting?

Serangan siber tidak selalu terlihat sebagai ledakan besar. Banyak serangan dimulai dari hal kecil:

- login gagal berulang,
- koneksi aneh ke alamat luar,
- file konfigurasi berubah,
- proses tidak dikenal berjalan,
- akun biasa tiba-tiba mengakses data sensitif,
- trafik jaringan melonjak pada jam tidak wajar,
- aplikasi menulis log error yang tidak biasa.

Tanpa SOC, sinyal-sinyal kecil ini sering hilang. Organisasi baru sadar ketika sistem sudah rusak, data sudah keluar, layanan berhenti, atau reputasi sudah jatuh.

Dengan SOC, organisasi memiliki kemampuan untuk:

- **melihat ancaman lebih awal**,
- **mengurangi waktu deteksi**,
- **mengurangi waktu respons**,
- **mencatat bukti teknis**,
- **membangun laporan insiden**,
- **memperbaiki kelemahan sistem**,
- **belajar dari pola serangan sebelumnya**.

Rujukan SOC menekankan bahwa pilar penting SOC adalah **people, process, technology, dan data**. Artinya, SOC tidak cukup hanya membeli atau memasang tools; SOC perlu orang yang mampu menganalisis, proses yang jelas, teknologi yang tepat, dan data yang dapat dipercaya.

1.4 SOC Bukan Sekadar Tools

Kesalahan umum pemula adalah menganggap SOC sama dengan SIEM, dashboard, atau alert. Ini keliru.

> **Tools hanya memperlihatkan sinyal. Analis yang memberi makna.**

Satu alert bisa berarti banyak hal:

- false positive,
- salah konfigurasi,
- aktivitas admin biasa,
- percobaan serangan,
- kompromi awal,
- serangan aktif,
- atau insiden besar yang sedang berkembang.

Karena itu, analis SOC harus memiliki tiga kemampuan dasar:

- **membaca log**, bukan hanya melihat dashboard;
- **membedakan noise dan sinyal**, bukan mengejar semua alert;
- **menghubungkan konteks**, bukan mengambil kesimpulan dari satu data.

SOC modern harus menggabungkan data endpoint, jaringan, log aplikasi, threat intelligence, dan catatan insiden. Rujukan SOC menekankan bahwa data adalah fondasi penting karena SOC membutuhkan data jaringan, endpoint, security logs, threat intelligence, dan perilaku pengguna untuk mendeteksi dan merespons ancaman.

1.5 Mengapa Open Source Penting?

Bagi mahasiswa, **open source adalah jalan paling realistis untuk belajar SOC secara serius**. Bukan sekadar karena gratis, tetapi karena terbuka untuk dipelajari, diuji, dimodifikasi, dan diintegrasikan.

Open source penting karena:

- **biaya masuk rendah**, cocok untuk kampus dan lab pribadi;
- **transparan**, konsep dan konfigurasi dapat dipelajari;
- **fleksibel**, bisa diintegrasikan melalui API, JSON, dan script;
- **komunitas luas**, dokumentasi dan contoh praktik mudah ditemukan;
- **melatih engineering mindset**, bukan hanya menjadi operator dashboard;
- **mendorong portofolio**, karena hasil lab bisa didokumentasikan dengan aman.

Wazuh, misalnya, secara resmi menyebut dirinya sebagai platform keamanan **free and open source** yang menyatukan kemampuan XDR dan SIEM untuk endpoint dan beban kerja lain. (Dokumentasi Wazuh) Security Onion juga diposisikan sebagai platform gratis dan terbuka untuk *network visibility*, *host visibility*, *intrusion detection*, *log management*, dan *case management*. (Dokumentasi Security Onion)

Bagi mahasiswa, ini strategis. Dengan open source, belajar SOC tidak harus menunggu akses ke alat mahal. Cukup dengan laptop yang memadai, Linux, virtualisasi, Docker, dan kemauan membaca log.

1.6 Tantangan SOC Modern

SOC modern menghadapi masalah yang makin berat. Ancaman bertambah cepat, sistem makin kompleks, dan volume data makin besar.

Tantangan utama SOC adalah sebagai berikut.

1. Biaya Tinggi

SOC komersial sering membutuhkan biaya lisensi, infrastruktur, storage, pelatihan, dan tenaga ahli. Bagi kampus, organisasi kecil, atau mahasiswa, biaya ini sering menjadi penghalang.

Open source menurunkan hambatan masuk. Namun, biaya tidak hilang sepenuhnya. Tetap dibutuhkan waktu, kemampuan teknis, server, storage, dokumentasi, dan disiplin operasional.

2. Skill Gap

SOC membutuhkan kombinasi keahlian:

- jaringan komputer,
- sistem operasi,
- keamanan aplikasi,
- analisis log,
- scripting,
- threat intelligence,
- incident response,
- komunikasi teknis,
- penulisan laporan.

Banyak pemula bisa menjalankan tools, tetapi belum tentu mampu menjelaskan **apa arti alert** dan **apa bukti teknisnya**.

3. Volume Log

Satu endpoint dapat menghasilkan ribuan event. Satu server dapat menghasilkan log autentikasi, aplikasi, sistem, jaringan, dan keamanan. Dalam SOC, masalahnya bukan kekurangan data, tetapi **terlalu banyak data**.

Tanpa struktur, data berubah menjadi kebisingan.

4. Alert Fatigue

Alert fatigue terjadi ketika analis menerima terlalu banyak alert, banyak di antaranya tidak penting. Akibatnya, alert penting bisa terlewat.

SOC yang baik tidak hanya menambah rule. SOC yang baik juga **mengurangi noise**.

5. Serangan Makin Cepat

Serangan modern sering bergerak cepat dari akses awal ke eskalasi hak akses, lateral movement, pencurian data, dan sabotase. Karena itu, SOC harus mampu mengurangi dua ukuran penting:

- **MTTD** — *Mean Time To Detect*.
- **MTTR** — *Mean Time To Respond*.

Tujuan SOC bukan membuat dashboard terlihat ramai. Tujuan SOC adalah **memperpendek waktu antara tanda bahaya dan tindakan yang benar**.

1.7 Peran AI dalam SOC Open Source

AI dapat membantu SOC, tetapi tidak boleh menggantikan bukti teknis. Dalam buku ini, AI diposisikan sebagai **asisten analis**, bukan hakim terakhir.

AI open source dapat membantu untuk:

- merangkum alert,
- menjelaskan log,
- mengelompokkan incident,
- membantu membuat timeline,
- membantu mapping ke MITRE ATT&CK,
- membantu membuat laporan,
- membantu menyusun playbook,
- membantu membaca pola dari banyak alert.

Namun AI memiliki risiko:

- salah menyimpulkan,
- terlalu percaya diri,
- tidak memahami konteks lokal,
- menghasilkan rekomendasi tanpa bukti,
- membocorkan data jika dipakai sembarangan.

Karena itu, aturan utamanya:

> **AI boleh membantu berpikir, tetapi keputusan SOC harus berbasis log, bukti, dan prosedur.**

Untuk buku ini, AI yang digunakan harus diarahkan ke ekosistem **open source** dan lokal sejauh mungkin. Data sensitif tidak boleh dikirim sembarangan ke layanan eksternal.

1.8 Perbedaan Istilah Penting dalam SOC

Sebelum masuk ke lab, mahasiswa perlu memahami istilah inti. Banyak istilah SOC terlihat mirip, tetapi fungsinya berbeda.

SIEM

Security Information and Event Management adalah sistem untuk mengumpulkan, menyimpan, mengindeks, mengkorelasikan, dan menganalisis log keamanan.

Fungsi utama SIEM:

- mengumpulkan log,
- mencari event,
- membuat alert,
- membuat dashboard,
- membantu investigasi,
- mendukung compliance.

Dalam buku ini, **Wazuh** dan **OpenSearch** menjadi fondasi SIEM open source.

XDR

Extended Detection and Response menggabungkan data dari beberapa sumber seperti endpoint, jaringan, server, workload, API, dan log lain untuk deteksi dan respons yang lebih luas. Wazuh secara resmi menyebut platformnya menyatukan kemampuan SIEM dan XDR. (Dokumentasi Wazuh)

Dalam praktek mahasiswa, XDR dipahami sebagai kemampuan melihat ancaman dari banyak titik, bukan hanya satu log.

EDR

Endpoint Detection and Response berfokus pada endpoint. Endpoint bisa berupa laptop, server, atau mesin virtual. EDR melihat aktivitas proses, file, login, konfigurasi, dan perilaku endpoint.

Dalam buku ini, kemampuan endpoint akan dipelajari melalui Wazuh agent dan osquery.

IDS/IPS

Intrusion Detection System mendeteksi aktivitas mencurigakan. *Intrusion Prevention System* berupaya mencegah atau memblokir aktivitas tertentu.

Dalam buku ini, **Suricata** dan **Zeek** digunakan untuk memahami trafik jaringan dan mendeteksi pola serangan. Security Onion juga memasukkan kemampuan intrusion detection dan monitoring jaringan dalam satu platform terbuka. (Dokumentasi Security Onion)

SOAR

Security Orchestration, Automation, and Response membantu mengotomasi alur respons. SOAR dapat menerima alert, memperkaya data, membuat case, mengirim notifikasi, atau menjalankan tindakan tertentu.

Namun otomasi harus hati-hati.

Tidak semua alert boleh langsung diblokir otomatis. Untuk lab mahasiswa, otomasi harus memakai mode aman, terdokumentasi, dan dapat diaudit.

Threat Intelligence

Threat intelligence adalah informasi terstruktur tentang ancaman, indikator, teknik, dan pola serangan. MISP adalah platform open source untuk mengumpulkan, menyimpan, mendistribusikan, dan berbagi indikator serta ancaman siber. (misp-project.org)

Dalam SOC, threat intelligence membantu menjawab:

- apakah IP ini dikenal berbahaya?
- apakah hash ini terkait malware?
- apakah domain ini bagian dari kampanye serangan?
- teknik MITRE apa yang mungkin digunakan?
- seberapa serius alert ini?

1.9 Peta Tools Open Source SOC

Buku ini memakai tools open source yang saling melengkapi. Tidak semua harus dipasang sekaligus di Bab 1. Bab ini hanya memberi peta besar.

Wazuh — Fondasi SIEM/XDR

Wazuh menjadi tulang punggung awal karena cocok untuk belajar:

- log collection,
- rule,
- decoder,
- alert,
- agent,
- FIM,
- vulnerability detection,
- active response,
- API,
- integrasi dengan tool lain.

Rujukan Wazuh menjelaskan bahwa Wazuh menyatukan SIEM dan XDR untuk threat detection, incident response, threat hunting, dan compliance management. Wazuh juga memiliki fitur intrusion detection, log analysis, FIM, vulnerability detection, dan security configuration assessment.

Security Onion — Visibilitas Jaringan

Security Onion dipakai untuk melihat sisi jaringan. Ia membantu mahasiswa memahami:

- trafik jaringan,
- alert jaringan,
- metadata koneksi,
- packet capture,
- threat hunting,
- case management.

Security Onion secara resmi menjelaskan dirinya sebagai platform terbuka untuk network visibility, host visibility, intrusion detection, honeypot, log management, dan case management. (Dokumentasi Security Onion)

Suricata dan Zeek — Deteksi dan Analisis Trafik

Suricata kuat untuk deteksi berbasis rule. Zeek kuat untuk metadata dan analisis protokol.

Kombinasi keduanya membantu mahasiswa memahami bahwa serangan tidak hanya tampak di endpoint. Banyak serangan meninggalkan jejak pada:

- koneksi,
- DNS,
- HTTP,
- TLS,
- port scanning,
- pola komunikasi,
- anomali trafik.

OpenSearch — Mesin Pencarian dan Analitik Log

OpenSearch dipakai untuk menyimpan, mencari, memvisualisasikan, dan menganalisis log. OpenSearch secara resmi dijelaskan sebagai suite open source untuk ingest, search, visualize, dan analyze data. (OpenSearch)

Dalam SOC, OpenSearch membantu membuat:

- dashboard alert,
- pencarian log,
- filter severity,
- grafik tren,
- analisis host,
- top source IP,
- top rule ID,
- laporan insiden.

DFIR-IRIS atau TheHive/Cortex — Case Management

Case management dibutuhkan agar SOC tidak hanya melihat alert, tetapi juga mengelola investigasi.

Fungsi case management:

- membuat case,
- mencatat evidence,
- membuat timeline,
- membagi task,
- menulis catatan analis,
- menutup incident,
- membuat laporan.

Dalam buku ini, penggunaan tool case management akan diarahkan ke opsi open source yang sesuai dengan kebutuhan lab.

MISP — Threat Intelligence

MISP digunakan untuk belajar IOC, threat intelligence, dan berbagi indikator ancaman. MISP dirancang untuk membantu analis insiden, profesional keamanan, dan malware analyst menyusun informasi ancaman secara terstruktur. (misp-project.org)

Shuffle — SOAR Open Source

Shuffle digunakan untuk memperkenalkan otomasi SOC. Fokusnya bukan membuat sistem otomatis yang agresif, tetapi membangun *workflow* sederhana:

- menerima alert JSON,
- melakukan parsing,
- enrich IOC,
- membuat case,
- mengirim notifikasi,
- mencatat hasil.

1.10 Cara Berpikir Analis SOC

Mahasiswa perlu memahami bahwa analis SOC bukan “penjaga dashboard”. Analis SOC adalah orang yang berpikir dengan pola investigasi.

Pola berpikir analis SOC:

1. **Lihat sinyal.**
   Alert apa yang muncul?
2. **Cari konteks.**
   Host apa? User apa? Jam berapa? Dari mana? Ke mana?
3. **Validasi bukti.**
   Apakah ada log pendukung? Apakah alert ini konsisten dengan event lain?
4. **Tentukan dampak.**
   Apakah ini menyentuh aset penting? Apakah ada tanda eskalasi?
5. **Klasifikasikan.**
   Event biasa, alert valid, false positive, atau incident?
6. **Respons.**
   Apa tindakan paling aman dan proporsional?
7. **Dokumentasikan.**
   Apa yang terjadi, bukti apa, tindakan apa, dan pelajaran apa?

Rujukan karier SOC juga menekankan pentingnya metode analis: alasan alert, bukti pendukung, analisis, kesimpulan, dan langkah berikutnya.

1.11 Etika Lab SOC

SOC lab harus dibangun dengan prinsip legal dan aman. Mahasiswa tidak boleh menggunakan latihan SOC sebagai alasan untuk menyerang sistem publik.

Aturan etika lab:

- **gunakan jaringan lokal atau virtual**;
- **gunakan target milik sendiri**;
- **jangan memindai sistem publik tanpa izin**;
- **jangan menjalankan eksploit di internet**;
- **jangan mengumpulkan data pribadi tanpa izin**;
- **jangan membagikan log sensitif**;
- **jangan membuka dashboard SOC ke internet**;
- **dokumentasikan semua eksperimen**;
- **pisahkan lab dari perangkat produksi**.

Prinsipnya sederhana:

> **Belajar menyerang hanya boleh dilakukan untuk memahami pertahanan, di lingkungan yang sah, aman, dan terkendali.**

1.12 Mini Hands-on 1 — Membuat Peta Arsitektur SOC Sederhana

Hands-on pertama tidak memakai instalasi. Mahasiswa cukup menggambar arsitektur SOC sederhana. Tujuannya agar pembaca memahami alur data sebelum menyentuh tools.

Komponen Minimum

Gunakan komponen berikut:

- **Endpoint Linux**
  Mesin yang menghasilkan log.
- **Server aplikasi lab**
  Mesin target untuk simulasi aktivitas normal dan tidak normal.
- **Network sensor**
  Mesin yang membaca trafik jaringan.
- **Wazuh server**
  Pusat pengumpulan alert dan log endpoint.
- **OpenSearch**
  Mesin pencarian dan dashboard log.
- **Case management**
  Tempat mencatat incident.
- **Threat intelligence**
  Sumber IOC dan konteks ancaman.
- **SOAR**
  Workflow otomasi sederhana.

Alur Log

Gambarkan alur berikut:

```
Endpoint / Server
      ↓
Agent / Sensor
      ↓
Wazuh / Security Onion
      ↓
OpenSearch
      ↓
Alert
      ↓
Triage Analis
      ↓
Case Management
      ↓
Response / Lesson Learned
```

Pertanyaan Refleksi

Jawab singkat:

- Data apa yang dikirim endpoint?
- Data apa yang dikirim network sensor?
- Di mana alert dibuat?
- Di mana case dibuat?
- Siapa yang memutuskan respons?
- Apa risiko jika respons otomatis dilakukan tanpa validasi?

1.13 Mini Hands-on 2 — Menentukan Kebutuhan Minimum Lab

Sebelum membangun SOC, mahasiswa harus tahu kebutuhan minimum.

Kebutuhan Minimal

- Laptop/PC dengan RAM minimal **16 GB**.
- Storage kosong minimal **100 GB**.
- Linux sebagai host atau VM utama.
- Virtualisasi berbasis open source.
- Docker dan Docker Compose.
- Koneksi internet untuk mengunduh paket.
- Jaringan lokal/virtual yang terisolasi.
- Editor teks open source.
- Python untuk script ringan.
- Git untuk dokumentasi dan versioning.

VM Minimum

Untuk tahap awal:

- **1 VM SOC server**.
- **1 VM endpoint Linux**.
- **1 VM target lab**.
- **1 VM network sensor**.

Untuk pemula, mulai dari kecil. Jangan langsung memasang semua tools. SOC yang baik dibangun bertahap.

1.14 Mini Hands-on 3 — Checklist Awal SOC Lab

Gunakan checklist berikut sebelum lanjut ke Bab 2.

Checklist Infrastruktur

- Host memiliki RAM dan storage cukup.
- Virtualisasi aktif.
- Jaringan lab terisolasi.
- VM dapat saling berkomunikasi.
- Snapshot tersedia.
- Password lab tidak memakai password pribadi.
- Dashboard tidak dibuka ke internet.

Checklist Data

- Ada endpoint penghasil log.
- Ada server target lab.
- Ada rencana pengumpulan log.
- Ada format alert JSON.
- Ada folder untuk menyimpan laporan.
- Ada catatan eksperimen.

Checklist Etika

- Tidak menyerang sistem publik.
- Tidak memakai target tanpa izin.
- Tidak membagikan data sensitif.
- Semua eksperimen dilakukan di jaringan lokal/virtual.
- Semua langkah ditulis sebagai dokumentasi belajar.

1.15 Rangkuman Bab

SOC adalah **sistem kerja pertahanan siber**, bukan sekadar tools. SOC menggabungkan manusia, proses, teknologi, dan data untuk mendeteksi ancaman, melakukan analisis, merespons insiden, dan memperbaiki keamanan organisasi.

Open source membuat SOC lebih mudah dipelajari oleh mahasiswa karena transparan, fleksibel, terjangkau, dan cocok untuk eksperimen. Tools seperti Wazuh, Security Onion, Suricata, Zeek, OpenSearch, MISP, DFIR-IRIS, TheHive/Cortex, dan Shuffle membentuk ekosistem pembelajaran SOC yang kuat, terbuka, dan praktis.

Bab ini menekankan bahwa AI boleh membantu SOC, tetapi **tidak boleh menggantikan bukti teknis**. AI harus dipakai sebagai asisten analis: membantu merangkum, menjelaskan, mengelompokkan, dan membuat laporan, tetapi keputusan tetap harus berbasis log, konteks, prosedur, dan tanggung jawab etis.

1.16 Output Bab

Setelah menyelesaikan Bab 1, pembaca diharapkan mampu:

- menjelaskan apa itu SOC;
- memahami peran SOC dalam pertahanan organisasi;
- menjelaskan mengapa open source penting untuk pembelajaran SOC;
- membedakan SIEM, XDR, EDR, IDS/IPS, SOAR, dan threat intelligence;
- memahami peta tools open source SOC;
- memahami etika lab SOC;
- membuat peta arsitektur SOC sederhana;
- membuat checklist awal untuk membangun lab SOC.

Pesan akhir Bab 1:
SOC modern bukan tentang melihat semua alert, tetapi tentang menemukan alert yang benar, membuktikannya dengan data, dan meresponsnya dengan disiplin.

Bab 2 — Membangun SOC Lab: Linux, VM, Docker, dan Jaringan Lokal

> **Lab SOC harus aman dulu, baru canggih; jangan pernah menguji deteksi dengan mengorbankan legalitas, etika, dan kontrol jaringan.**

2.1 Arah Bab

Bab ini menjadi fondasi praktik seluruh buku. Sebelum memasang Wazuh, Security Onion, OpenSearch, MISP, DFIR-IRIS, atau Shuffle, pembaca harus memiliki **lingkungan lab yang aman, terisolasi, dan mudah diulang**.

Arah buku dari dokumen utama menempatkan Bab 2 sebagai tahap pembangunan lab: **SOC server, endpoint, mesin simulasi, sensor jaringan, segmentasi jaringan, snapshot, backup konfigurasi, dan struktur folder proyek**.

Bab ini tidak mengejar banyak tools. Fokusnya adalah **membangun pondasi yang benar**.

SOC lab yang baik harus memenuhi empat prinsip:

- **aman**, tidak menyentuh sistem publik;
- **terisolasi**, tidak mengganggu jaringan utama;
- **terdokumentasi**, semua IP, VM, konfigurasi, dan perubahan dicatat;
- **mudah diulang**, jika rusak bisa dikembalikan dari snapshot atau backup.

2.2 Mengapa Lab SOC Harus Dibangun Bertahap?

Mahasiswa sering ingin langsung memasang semua komponen sekaligus. Ini berbahaya untuk proses belajar. SOC bukan lomba memasang tools terbanyak. SOC adalah latihan membaca data, memahami alur log, dan membangun respons yang benar.

Rujukan SOC menekankan bahwa desain SOC harus memperhatikan **skalabilitas**, **fleksibilitas**, SIEM, threat intelligence, incident response, vulnerability management, security analytics, serta organisasi kerja analis.

Untuk mahasiswa, prinsipnya sederhana:

- mulai dari **lab kecil**;
- pastikan jaringan **terkendali**;
- pastikan log **masuk**;
- pastikan alert **terbaca**;
- baru tambah sensor, dashboard, threat intelligence, dan otomasi.

  Jangan membangun SOC seperti menumpuk aplikasi. Bangun SOC seperti membangun alur investigasi.

2.3 Desain Minimum Lab SOC

Desain minimum lab SOC dalam buku ini terdiri dari beberapa komponen. Semua komponen dapat dijalankan sebagai VM atau container sesuai kemampuan hardware.

Komponen Minimum

1. SOC Server

SOC server adalah pusat layanan keamanan. Pada tahap awal, server ini dapat dipakai untuk:

- Wazuh server;
- Wazuh dashboard;
- Wazuh indexer;
- OpenSearch;
- tool pendukung lain pada bab berikutnya.

Dalam rujukan Wazuh, komponen utama Wazuh terdiri dari **Wazuh server**, **Wazuh indexer**, **Wazuh dashboard**, dan **Wazuh agent**. Wazuh indexer menyimpan alert, dashboard dipakai untuk visualisasi dan analisis, sedangkan agent mengirim event dari endpoint.

2. Endpoint Linux

Endpoint Linux adalah mesin yang dimonitor. Mesin ini dipakai untuk belajar:

- log sistem;
- login berhasil dan gagal;
- perubahan file;
- service berjalan;
- konfigurasi keamanan;
- agent SOC.

Endpoint ini penting karena SOC selalu membutuhkan sumber data. Tanpa endpoint, SOC hanya dashboard kosong.

3. Endpoint Tambahan Opsional

Endpoint tambahan dapat memakai sistem operasi lain yang sah untuk lab. Namun, buku ini menjaga arah open source. Karena itu, fokus utama tetap pada **Linux endpoint**.

Endpoint tambahan hanya dipakai jika pembaca ingin memahami variasi log dari lingkungan berbeda. Untuk buku ini, seluruh praktik inti tetap bisa dijalankan dengan Linux.

4. Mesin Simulasi Terkendali

Mesin ini dipakai untuk simulasi aktivitas keamanan dalam lab. Fungsinya bukan untuk menyerang sistem publik, tetapi untuk menghasilkan event agar SOC dapat belajar membaca sinyal.

Contoh fungsi:

- membuat koneksi ke server lab;
- menguji login gagal;
- membuat trafik jaringan;
- memicu alert sederhana;
- menguji apakah sensor berjalan.

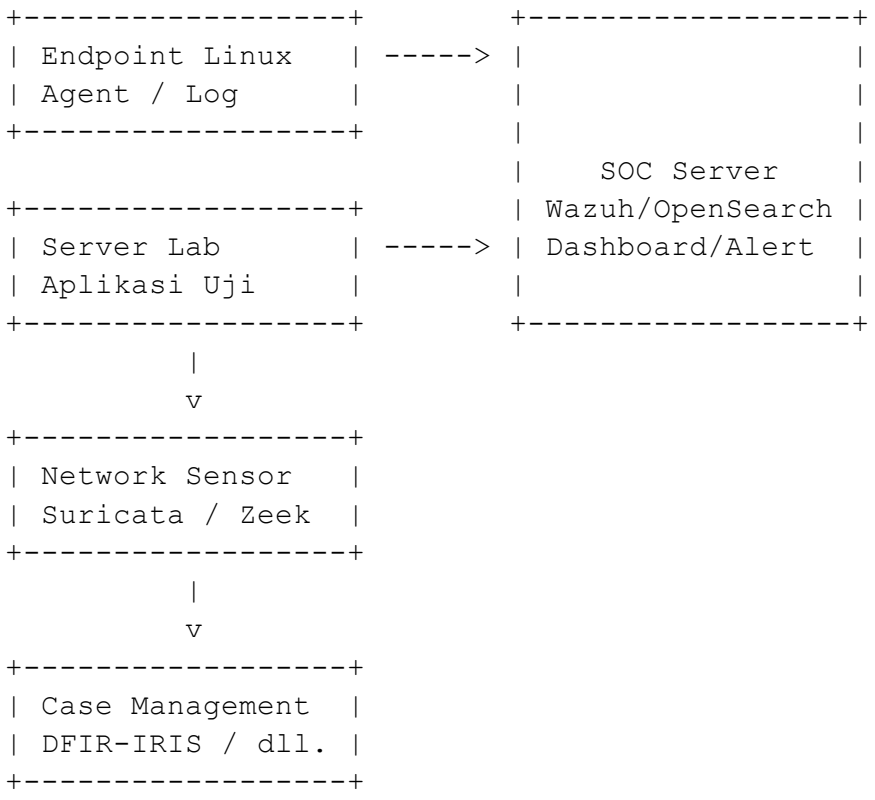
5. Network Sensor

Network sensor membaca trafik jaringan di lab. Sensor ini membantu mahasiswa melihat aktivitas dari sisi jaringan, bukan hanya endpoint.

Dalam rujukan Wazuh, salah satu rancangan lab untuk deteksi serangan web menggunakan empat bagian: mesin simulasi, server korban lab, TAP server yang menjalankan Suricata dan Wazuh agent, serta Wazuh server. Sensor ditempatkan dalam mode yang mampu membaca trafik jaringan lab.

2.4 Arsitektur Sederhana Lab SOC

Arsitektur awal dibuat sederhana agar mudah dipahami.



Alur sederhananya:

- endpoint menghasilkan log;
- agent mengirim log ke SOC server;
- sensor membaca trafik jaringan;
- SOC server membuat alert;
- analis melakukan triage;
- case dibuat jika alert valid;
- respons dilakukan secara terukur;
- hasil dicatat sebagai pelajaran.

Tujuan Bab 2 bukan membuat SOC lengkap, tetapi membuat jalur data yang siap dipakai.

2.5 Pilihan Platform Virtualisasi Open Source

Lab SOC membutuhkan virtualisasi. Virtualisasi memungkinkan satu laptop atau server menjalankan beberapa mesin lab sekaligus.

Untuk menjaga arah open source, buku ini mengutamakan:

KVM/libvirt

KVM adalah teknologi virtualisasi di Linux. libvirt adalah toolkit untuk mengelola platform virtualisasi dan mendukung KVM/QEMU. Dokumentasi libvirt menjelaskan bahwa libvirt adalah toolkit untuk mengelola platform virtualisasi, tersedia melalui banyak bahasa pemrograman, dan berlisensi open source. (libvirt.org)

Cocok untuk:

- pengguna Linux;
- mahasiswa yang ingin belajar serius;
- lab lokal;
- otomasi VM;
- integrasi scripting.

Proxmox VE

Proxmox VE adalah platform virtualisasi server open source berbasis QEMU/KVM dan LXC. Dokumentasi resmi Proxmox menyebut Proxmox VE sebagai solusi manajemen virtualisasi server open source. (pve.proxmox.com)

Cocok untuk:

- lab kampus;
- server bersama;
- banyak VM;
- pengelolaan lewat antarmuka web;
- snapshot dan backup yang lebih nyaman.

Docker dan Docker Compose

Docker dipakai untuk menjalankan layanan dalam container. Docker Engine mendukung Ubuntu versi 64-bit modern, termasuk Ubuntu LTS, dan dokumentasi resminya menyediakan panduan instalasi untuk Ubuntu. (Docker Documentation)

Cocok untuk:

- menjalankan layanan pendukung;
- lab cepat;
- deployment berulang;
- eksperimen API;
- integrasi tool SOC.

Namun perlu diingat:

VM dan container berbeda.
VM menjalankan sistem operasi lengkap. Container menjalankan aplikasi dalam lingkungan terisolasi yang lebih ringan.

Dalam SOC lab, keduanya bisa dipakai bersama:

- VM untuk SOC server, endpoint, dan sensor;
- container untuk layanan pendukung seperti dashboard, database, atau automation.

2.6 Segmentasi Jaringan Lab

Segmentasi jaringan adalah inti keamanan lab. Tanpa segmentasi, eksperimen bisa bocor ke jaringan utama.

NAT

NAT membuat VM dapat keluar ke internet melalui host, tetapi tidak mudah diakses langsung dari luar.

Cocok untuk:

- update paket;
- install tools;
- download image;
- akses internet terbatas.

Kelemahannya:

- kurang ideal untuk sensor jaringan;
- trafik antar VM bisa tidak terlihat lengkap;
- sulit untuk simulasi monitoring jaringan.

Host-only / Internal Network

Mode ini membuat VM hanya berkomunikasi dalam jaringan lokal virtual.

Cocok untuk:

- lab aman;
- simulasi endpoint dan server;
- pengujian sensor;
- eksperimen deteksi.

Ini mode paling aman untuk mahasiswa.

Bridge

Mode bridge membuat VM berada seolah-olah satu jaringan dengan host fisik.

Cocok untuk:

- lab lanjutan;
- server kampus;
- kebutuhan akses dari perangkat lain.

Namun bridge harus hati-hati. Jangan gunakan bridge untuk eksperimen yang menghasilkan trafik uji tanpa kontrol.

Isolasi Trafik

Isolasi trafik berarti membatasi alur komunikasi. Contoh:

- endpoint hanya boleh mengirim log ke SOC server;
- mesin simulasi hanya boleh mengakses server lab;
- dashboard hanya boleh dibuka dari host tertentu;
- sensor hanya membaca trafik lab;
- tidak ada layanan SOC terbuka ke internet.

Lab SOC yang baik adalah lab yang bisa “berisik” di dalam, tetapi tetap diam ke luar.

2.7 Prinsip Keamanan Lab

Sebelum instalasi, pembaca harus memahami prinsip keamanan dasar.

1. Snapshot Sebelum Eksperimen

Snapshot adalah titik kembali. Setiap kali akan mengubah konfigurasi besar, buat snapshot.

Gunakan snapshot sebelum:

- memasang komponen besar;
- mengubah konfigurasi jaringan;
- mengaktifkan sensor;
- mengubah rule;
- menguji alert;
- mengintegrasikan API.

Snapshot menghemat waktu. Tanpa snapshot, satu kesalahan konfigurasi bisa memaksa instal ulang.

2. Backup Konfigurasi

Snapshot bukan pengganti backup. Snapshot menjaga kondisi VM, tetapi backup menjaga file penting.

Backup minimal:

- file konfigurasi;
- file compose;
- rule;
- decoder;
- script;
- laporan;
- inventaris aset;
- catatan eksperimen.

3. Dashboard Jangan Diekspos ke Internet

Dashboard SOC sering berisi informasi sensitif:

- nama host;
- IP internal;
- username;
- alert;
- vulnerability;
- struktur jaringan;
- case incident.

Karena itu:

- jangan buka dashboard ke internet;
- gunakan akses lokal;
- gunakan firewall;
- gunakan password kuat;
- aktifkan pembatasan akses;
- jangan memakai kredensial default.

4. Password Kuat dan Akses Terbatas

Gunakan password unik untuk lab. Jangan memakai password pribadi.

Aturan sederhana:

- minimal 14 karakter;
- tidak memakai nama pribadi;
- tidak memakai tanggal lahir;
- tidak dipakai ulang;
- simpan di password manager open source jika perlu;
- pisahkan akun admin dan akun biasa.

5. Dokumentasi Perubahan

Setiap perubahan harus dicatat.

Minimal catat:

- tanggal;
- VM yang diubah;
- konfigurasi yang diganti;
- alasan perubahan;
- hasil setelah perubahan;
- error yang muncul;
- solusi.

SOC bukan hanya kemampuan teknis. SOC adalah **disiplin dokumentasi**.

2.8 Spesifikasi Minimum Lab

Spesifikasi berikut cukup untuk memulai.

Minimum Pribadi

- CPU 4 core.
- RAM 16 GB.
- Storage kosong 100–150 GB.
- Linux sebagai host utama.
- Koneksi internet stabil.
- Virtualisasi aktif di firmware.
- Docker dan Docker Compose.

Ideal untuk Mahasiswa Serius

- CPU 6–8 core.
- RAM 32 GB.
- SSD 250–500 GB khusus lab.
- Proxmox VE atau KVM/libvirt.
- Jaringan lab terpisah.
- Backup eksternal.

Ideal untuk Lab Kampus

- Server RAM 64 GB atau lebih.
- Storage SSD untuk VM.
- Proxmox VE.
- VLAN atau jaringan lab terpisah.
- Akun mahasiswa terkontrol.
- Template VM siap clone.
- Kebijakan reset lab.

Ubuntu Server dapat menjadi basis server minimal. Dokumentasi resmi Ubuntu Server mencakup instalasi, konfigurasi, keamanan, networking, virtualisasi, container, dan manajemen sistem. (Ubuntu)

2.9 Struktur Folder Proyek SOC

Struktur folder penting agar lab tidak berantakan. Semua file konfigurasi dan laporan harus disimpan rapi.

Gunakan struktur berikut:

```
soc-lab/
├── docker/
├── configs/
├── logs/
├── rules/
├── reports/
├── inventory/
├── diagrams/
└── notes/
```

Penjelasan:

- **docker/**
  Menyimpan file Docker Compose dan konfigurasi container.
- **configs/**
  Menyimpan konfigurasi service SOC.
- **logs/**
  Menyimpan contoh log untuk latihan.
- **rules/**
  Menyimpan rule deteksi.
- **reports/**
  Menyimpan laporan insiden dan laporan praktikum.
- **inventory/**
  Menyimpan daftar aset lab.
- **diagrams/**
  Menyimpan diagram arsitektur.
- **notes/**
  Menyimpan catatan eksperimen.

Struktur folder adalah fondasi portofolio.
Mahasiswa yang rapi dalam dokumentasi akan lebih mudah menjelaskan proyeknya.

2.10 Hands-on 1 — Instalasi Ubuntu Server untuk SOC Lab

Bagian ini memberi alur praktik. Perintah detail dapat disesuaikan dengan versi sistem.

Tujuan

Membuat satu VM Linux sebagai server SOC awal.

Spesifikasi VM SOC Server

- CPU: 2–4 core.
- RAM: 8–12 GB.
- Disk: 80–120 GB.
- Network 1: NAT untuk update.
- Network 2: jaringan lab internal.
- OS: Ubuntu Server LTS.

Ubuntu menyediakan panduan resmi instalasi server, termasuk kebutuhan sistem dasar, ISO, media boot, dan proses instalasi. (Ubuntu)

Langkah Ringkas

1. Buat VM baru.
2. Pasang ISO Ubuntu Server.
3. Pilih instalasi minimal.
4. Buat user admin lab.
5. Aktifkan SSH hanya untuk jaringan lab.
6. Update sistem.
7. Catat IP address.
8. Buat snapshot awal.

Perintah Awal Setelah Instalasi

```
sudo apt update
sudo apt upgrade -y
hostnamectl
ip addr
df -h
free -h
```

Penjelasan singkat:

- `sudo` menjalankan perintah dengan hak admin.
- `apt update` memperbarui daftar paket.
- `apt upgrade -y` memperbarui paket yang sudah terpasang.
- `hostnamectl` melihat nama host dan informasi sistem.
- `ip addr` melihat alamat jaringan.
- `df -h` melihat kapasitas disk.
- `free -h` melihat penggunaan memori.

Catatan: jangan lanjut memasang tools besar sebelum update berhasil dan snapshot dibuat.

2.11 Hands-on 2 — Instalasi Docker dan Docker Compose

Docker akan dipakai untuk beberapa layanan pendukung SOC. Docker Engine resmi mendukung instalasi di Ubuntu 64-bit modern dan memiliki dokumentasi instalasi khusus untuk Ubuntu. (Docker Documentation)

Tujuan

Menyiapkan container runtime untuk menjalankan layanan SOC berbasis open source.

Perintah Instalasi Ringkas

```
sudo apt update
sudo apt install -y ca-certificates curl gnupg
```

Penjelasan:

- `apt update` memperbarui daftar paket.
- `apt install` memasang paket.
- `ca-certificates` membantu validasi koneksi HTTPS.
- `curl` dipakai mengambil data dari URL.
- `gnupg` dipakai untuk verifikasi kunci paket.

Setelah repository resmi Docker disiapkan sesuai dokumentasi, lanjutkan instalasi:

```
sudo apt install -y docker-ce docker-ce-cli
containerd.io docker-buildx-plugin docker-compose-plugin
```

Penjelasan:

- `docker-ce` adalah Docker Engine.
- `docker-ce-cli` adalah command line Docker.
- `containerd.io` adalah runtime container.
- `docker-buildx-plugin` membantu proses build image.
- `docker-compose-plugin` menjalankan layanan multi-container.

Cek hasil instalasi:

```
docker --version
docker compose version
sudo systemctl status docker
```

Penjelasan:

- `docker --version` melihat versi Docker.
- `docker compose version` melihat versi Compose.
- `systemctl status docker` melihat status service Docker.

Agar user lab dapat menjalankan Docker tanpa `sudo`:

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

Penjelasan:

- `usermod` mengubah user.
- `-aG docker` menambahkan user ke grup Docker.
- `$USER` berarti user yang sedang aktif.

Setelah itu logout dan login kembali.

Catatan keamanan: akses ke grup Docker setara dengan akses kuat ke host. Jangan berikan ke user yang tidak dipercaya.

2.12 Hands-on 3 — Membuat Struktur Folder SOC Lab

Jalankan:

```
mkdir -p
~/soc-lab/{docker,configs,logs,rules,reports,inventory,diagra
ms,notes}
cd ~/soc-lab
tree -L 2
```

Penjelasan:

- `mkdir` membuat folder.
- `-p` membuat folder bertingkat tanpa error jika sudah ada.
- `~/soc-lab` berarti folder `soc-lab` di home user.
- `{docker,configs,...}` membuat banyak folder sekaligus.
- `cd` masuk ke folder.
- `tree -L 2` menampilkan struktur folder sampai dua level.

Jika `tree` belum ada:

```
sudo apt install -y tree
```

Output yang diharapkan:

```
soc-lab/
├── configs
├── diagrams
├── docker
├── inventory
├── logs
├── notes
├── reports
└── rules
```

2.13 Hands-on 4 — Membuat Inventaris Aset Lab dalam JSON

Inventaris aset penting agar SOC tahu mesin apa saja yang dipantau.

Buat file:

```
nano ~/soc-lab/inventory/assets.json
```

Isi contoh:

```
{
  "lab_name": "open-source-soc-lab",
  "owner": "mahasiswa",
  "network": {
    "lab_cidr": "10.10.10.0/24",
    "internet_access": "restricted"
  },
  "assets": [
    {
      "hostname": "soc-server",
      "role": "siem-server",
      "ip": "10.10.10.10",
      "os": "ubuntu-server",
      "criticality": "high"
    },
    {
      "hostname": "linux-endpoint-01",
      "role": "monitored-endpoint",
      "ip": "10.10.10.21",
      "os": "linux",
      "criticality": "medium"
    },
    {
      "hostname": "lab-target-01",
      "role": "test-target",
      "ip": "10.10.10.30",
      "os": "linux",
      "criticality": "low"
    },
    {
      "hostname": "network-sensor-01",
      "role": "network-sensor",
      "ip": "10.10.10.40",
      "os": "linux",
      "criticality": "high"
    }
  ]
}
```

Penjelasan bagian:

- `lab_name` adalah nama lab.
- `owner` adalah pemilik atau kelompok praktikum.
- `network` menyimpan informasi jaringan lab.
- `assets` berisi daftar mesin.
- `hostname` adalah nama mesin.
- `role` menjelaskan fungsi mesin.
- `ip` adalah alamat jaringan.
- `os` adalah sistem operasi.
- `criticality` menunjukkan tingkat kepentingan aset.

Validasi JSON:

```
python3 -m json.tool ~/soc-lab/inventory/assets.json
```

Penjelasan:

- `python3` menjalankan Python.
- `-m json.tool` memvalidasi dan merapikan JSON.
- Jika JSON salah, akan muncul pesan error.

2.14 Hands-on 5 — Membuat Diagram Alur Log

Buat file diagram teks:

```
nano ~/soc-lab/diagrams/log-flow.txt
```

Isi:

```
Endpoint Linux
    |
    | log / event
    v
Agent / Collector
    |
    | normalized event
    v
SOC Server
    |
    | alert
    v
Dashboard / Search
    |
    | triage
    v
Case Management
    |
    | response note
    v
Lessons Learned
```

Diagram ini sederhana, tetapi penting. Ia menunjukkan bahwa SOC bukan hanya menerima log. SOC harus mengubah log menjadi alert, alert menjadi triage, triage menjadi case, dan case menjadi pembelajaran.

2.15 Checklist Keamanan Sebelum Lanjut Bab 3

Sebelum masuk ke Wazuh, pastikan checklist berikut selesai.

Infrastruktur

- VM SOC server sudah terpasang.
- Endpoint Linux sudah tersedia.
- Mesin simulasi terkendali sudah tersedia.
- Network sensor sudah direncanakan.
- Jaringan lab terisolasi.
- IP setiap mesin sudah dicatat.
- Snapshot awal sudah dibuat.

Docker

- Docker Engine terpasang.
- Docker Compose tersedia.
- Service Docker aktif.
- User lab dapat menjalankan Docker.
- Hak akses Docker dibatasi.

Dokumentasi

- Folder `soc-lab/` sudah dibuat.
- File `assets.json` sudah ada.
- Diagram alur log sudah ada.
- Catatan instalasi disimpan.
- Error dan solusi dicatat.

Keamanan

- Dashboard belum dibuka ke internet.
- Password default tidak digunakan.
- SSH hanya untuk jaringan lab.
- Tidak ada eksperimen ke sistem publik.
- Backup konfigurasi awal sudah dibuat.

2.16 Kesalahan Umum Mahasiswa

Beberapa kesalahan yang sering terjadi:

- memasang terlalu banyak tools sekaligus;
- tidak mencatat IP VM;
- tidak membuat snapshot;
- mencampur lab dengan jaringan utama;
- memakai password lemah;
- membuka dashboard ke internet;
- tidak memahami perbedaan VM dan container;
- langsung percaya dashboard tanpa membaca log;
- tidak menyimpan konfigurasi;
- tidak membuat laporan praktikum.

SOC lab yang kacau akan menghasilkan analisis yang kacau.

2.17 Peran AI Open Source dalam Lab SOC

Pada Bab 2, AI belum menjadi komponen utama. Namun lab sudah disiapkan agar pada bab berikutnya dapat mendukung AI lokal untuk membantu SOC.

AI open source nantinya dapat membantu:

- merangkum alert;
- menjelaskan log;
- membuat laporan awal;
- membantu klasifikasi severity;
- menyusun timeline;
- memberi saran playbook.

Namun sejak awal perlu ditegaskan:

> **AI tidak boleh menjadi sumber kebenaran utama. Log tetap bukti utama.**

Dalam struktur folder, pembaca dapat menambahkan folder:

ai-prompts/

```
Folder ini nanti dipakai untuk menyimpan prompt SOC yang
aman, tidak mengandung data sensitif, dan dapat diuji ulang.
```

2.18 Rangkuman Bab

Bab ini membangun fondasi lab SOC berbasis open source. Pembaca belajar bahwa lab SOC tidak boleh dibangun sembarangan. Lab harus aman, terisolasi, terdokumentasi, dan dapat diulang.

Komponen minimum lab terdiri dari SOC server, endpoint Linux, mesin simulasi terkendali, server target lab, dan network sensor. Platform yang disarankan adalah KVM/libvirt, Proxmox VE, Docker, dan Docker Compose. Semua eksperimen dilakukan di jaringan lokal/virtual agar aman dan legal.

Bab ini juga menyiapkan struktur folder, inventaris aset JSON, dan diagram alur log. Ketiganya akan menjadi dasar untuk bab berikutnya saat pembaca mulai memasang Wazuh sebagai fondasi SIEM/XDR open source.

2.19 Output Bab

Setelah menyelesaikan Bab 2, pembaca diharapkan mampu:

- menjelaskan desain minimum SOC lab;
- memilih platform virtualisasi open source;
- memahami NAT, jaringan internal, bridge, dan isolasi trafik;
- memasang Ubuntu Server untuk SOC lab;
- memasang Docker dan Docker Compose;
- membuat struktur folder proyek SOC;
- membuat inventaris aset lab dalam JSON;
- membuat diagram alur log;
- menerapkan prinsip keamanan lab;
- siap masuk ke Bab 3: Wazuh sebagai fondasi SIEM/XDR open source.

Pesan akhir Bab 2:
SOC lab bukan tempat bermain sembarangan; SOC lab adalah ruang latihan profesional untuk membaca bukti, menjaga etika, dan membangun kemampuan pertahanan siber yang nyata.

Bab 3 — Wazuh sebagai Fondasi SIEM/XDR Open Source

> **Wazuh bukan sekadar dashboard alert; Wazuh adalah fondasi untuk membaca log, membuktikan ancaman, dan membangun respons SOC berbasis data.**

3.1 Arah Bab

Bab ini menjadi inti awal dari buku. Setelah Bab 2 membangun *lab*, Bab 3 mulai memasang fondasi utama SOC: **Wazuh**.

Dalam outline buku, Wazuh ditempatkan sebagai fondasi **SIEM/XDR open source** karena mampu menggabungkan *log analysis*, *intrusion detection*, *malware detection*, *file integrity monitoring*, *vulnerability detection*, *security configuration assessment*, *active response*, dan *compliance*.

Wazuh dipilih karena cocok untuk mahasiswa:

- **open source**;
- dokumentasi kuat;
- bisa dipasang di lab lokal;
- memiliki agent;
- alert dapat dibaca dalam format JSON;
- mendukung API;
- dapat diintegrasikan dengan tools lain;
- cocok untuk latihan SOC dari dasar sampai proyek akhir.

Wazuh secara resmi dijelaskan sebagai platform keamanan *free and open source* yang menyatukan kemampuan **XDR** dan **SIEM** untuk endpoint dan *workload* lain. (Wazuh)

3.2 Mengapa Wazuh Menjadi Fondasi SOC?

SOC membutuhkan tiga hal pada tahap awal:

- **sumber data**;
- **mesin analisis**;
- **tempat membaca alert**.

Wazuh menyediakan ketiganya dalam satu ekosistem. Agent mengumpulkan data dari endpoint. Server menganalisis data. Indexer menyimpan alert. Dashboard membantu analis membaca, mencari, dan memahami kondisi keamanan.

Rujukan Wazuh menjelaskan bahwa Wazuh adalah platform open source yang menyatukan kemampuan SIEM dan XDR untuk meningkatkan *threat detection*, *incident response*, *threat hunting*, dan *compliance management*. Wazuh juga menggabungkan fitur seperti *intrusion detection*, *log analysis*, *file integrity monitoring*, *vulnerability detection*, dan *security configuration assessment*.

Bagi mahasiswa, Wazuh memberi pengalaman nyata:

- memasang server SOC;
- mendaftarkan endpoint;
- membaca alert;
- memahami rule;
- menguji FIM;
- membuat custom rule;
- mengambil data melalui API;
- menulis laporan berbasis bukti.

> **SOC tidak dimulai dari AI. SOC dimulai dari log yang benar.**

AI baru berguna jika data SOC sudah rapi.

3.3 Arsitektur Wazuh

Arsitektur Wazuh terdiri dari **Wazuh agent** dan tiga komponen pusat: **Wazuh server**, **Wazuh indexer**, dan **Wazuh dashboard**. Agent dipasang pada endpoint untuk mengumpulkan dan mengirim data keamanan ke Wazuh server; data yang sudah dianalisis dikirim ke indexer untuk disimpan, lalu ditampilkan melalui dashboard. (Dokumentasi Wazuh)

1. Wazuh Agent

Wazuh agent dipasang pada endpoint yang ingin di monitor.

Fungsinya:

- mengumpulkan log sistem;
- mengirim event ke Wazuh server;
- memonitor perubahan file;
- mengirim inventaris sistem;
- membantu deteksi aktivitas mencurigakan;
- menjalankan respons tertentu jika dikonfigurasi.

Dalam lab buku ini, endpoint utama adalah **Linux endpoint**.

Agent adalah mata dan telinga SOC. Tanpa agent atau sensor, SOC tidak punya data.

2. Wazuh Server

Wazuh server adalah pusat analisis.

Fungsinya:

- menerima data dari agent;
- melakukan parsing log;
- menjalankan decoder;
- menjalankan rule;
- menghasilkan alert;
- mengelola agent;
- menyediakan API;
- mendukung integrasi dengan tools lain.

Rujukan Wazuh menjelaskan bahwa Wazuh server mengumpulkan log dari berbagai sumber, menormalisasi log, memakai decoder untuk parsing, dan menyediakan API untuk interaksi seperti melihat agent aktif, informasi vulnerability, dan verifikasi rule.

3. Wazuh Indexer

Wazuh indexer menyimpan dan mengindeks alert.

Fungsinya:

- menyimpan alert dari Wazuh server;
- membuat data mudah dicari;
- mendukung pencarian cepat;
- menyimpan data untuk dashboard;
- membantu analisis historis.

Indexer penting karena SOC tidak hanya membaca alert saat ini. SOC perlu melihat pola dari waktu ke waktu.

4. Wazuh Dashboard

Wazuh dashboard adalah antarmuka visual.

Fungsinya:

- melihat status agent;
- membaca alert;
- mencari event;
- melihat FIM;
- melihat vulnerability;
- memantau compliance;
- membantu triage awal.

Dokumentasi instalasi Wazuh menjelaskan bahwa dashboard adalah antarmuka web untuk visualisasi, analisis, dan manajemen data, termasuk dashboard untuk compliance, vulnerability, file integrity, configuration assessment, dan event infrastruktur. (Wazuh)

3.4 Alur Data Wazuh

Alur data Wazuh dapat dibaca sederhana:

```
Endpoint Linux
    ↓
Wazuh Agent
    ↓
Wazuh Server
    ↓
Decoder + Rule
    ↓
Alert
    ↓
Wazuh Indexer
    ↓
Wazuh Dashboard
    ↓
Analisis SOC
```

Maknanya:

- endpoint menghasilkan event;
- agent mengirim event;
- server membaca dan menganalisis;
- decoder mengekstrak field penting;
- rule menentukan apakah event menjadi alert;
- indexer menyimpan alert;
- dashboard menampilkan hasil;
- analis mengambil keputusan.

Alert bukan akhir proses. Alert adalah awal investigasi.

3.5 Konsep Inti: Log, Event, Alert, Rule, Decoder, Group, Agent

Sebelum praktik, mahasiswa harus memahami istilah inti Wazuh.

Log

Log adalah catatan aktivitas.

Contoh log:

- login berhasil;
- login gagal;
- service berjalan;
- file berubah;
- paket terpasang;
- koneksi jaringan;
- konfigurasi berubah.

Log adalah bahan mentah. Belum tentu berbahaya.

Event

Event adalah kejadian yang diproses dari log.

Tidak semua event menjadi alert. Banyak event hanya aktivitas normal.

Contoh:

- user login;
- service restart;
- file dibaca;
- paket diperbarui.

Alert

Alert adalah event yang dianggap penting oleh rule.

Alert memiliki informasi seperti:

- waktu;
- agent;
- rule ID;
- level;
- deskripsi;
- lokasi log;
- field terkait;
- mapping taktik/teknik jika tersedia.

Decoder

Decoder adalah mekanisme untuk mengekstrak informasi dari log. Wazuh menggunakan decoder untuk mengambil field dari pesan log agar siap dianalisis. (Dokumentasi Wazuh)

Contoh sederhana:

```
Log mentah:
Failed password for user admin from 10.10.10.50
```

Decoder membantu mengambil:

- aktivitas: failed password;
- user: admin;
- source IP: 10.10.10.50.

Rule

Rule adalah kondisi deteksi. Wazuh rules ditulis dalam XML dan digunakan oleh Wazuh manager untuk mendeteksi pola atau perilaku tertentu dalam log, lalu menghasilkan alert atau respons. (Dokumentasi Wazuh)

Rule menjawab:

- pola apa yang dicari?
- level berapa?
- deskripsi alert apa?
- masuk ke grup apa?
- apakah terkait rule lain?

Group

Group mengelompokkan rule dan alert.

Contoh group:

- `authentication;`
- `ssh;`
- `syslog;`
- `fim;`
- `vulnerability;`
- `compliance.`

Group memudahkan filter, dashboard, dan analisis.

Agent

Agent adalah komponen pada endpoint. Agent mengirim event ke Wazuh server. Agent membuat SOC bisa melihat endpoint secara terpusat.

3.6 Fungsi Utama Wazuh

Wazuh memiliki banyak fungsi. Untuk buku ini, fokus pada fungsi yang paling penting untuk lab mahasiswa.

1. Log Collection

Wazuh mengumpulkan log dari endpoint dan sumber lain. Dalam konteks SIEM, Wazuh mengumpulkan event dari endpoint, perangkat jaringan, workload, dan aplikasi untuk analisis keamanan. (Wazuh)

Fungsi ini penting karena SOC dimulai dari data.

Tanpa log:

- tidak ada alert;
- tidak ada bukti;
- tidak ada investigasi;
- tidak ada laporan.

2. File Integrity Monitoring

File Integrity Monitoring atau **FIM** memonitor perubahan file dan direktori. Dokumentasi Wazuh menjelaskan bahwa modul FIM berjalan pada beberapa sistem operasi dan dapat dikonfigurasi melalui pengaturan global maupun spesifik endpoint. (Dokumentasi Wazuh)

FIM berguna untuk mendeteksi:

- file konfigurasi berubah;
- file baru muncul;
- file penting dihapus;
- permission berubah;
- indikasi modifikasi tidak sah.

Dalam lab ini, contoh file yang dipantau:

```
/etc/ssh/sshd_config
```

3. Vulnerability Detection

Wazuh dapat membantu mendeteksi vulnerability pada endpoint. Rujukan Wazuh menjelaskan bahwa modul Vulnerability Detector menggunakan inventaris sistem dari agent, termasuk informasi paket dan aplikasi, untuk mengidentifikasi CVE yang relevan.

Bagi mahasiswa, fungsi ini penting untuk memahami bahwa SOC bukan hanya mendeteksi serangan. SOC juga membantu melihat kelemahan yang bisa dimanfaatkan penyerang.

4. Malware Detection

Wazuh dapat dipakai dalam skenario deteksi malware melalui kombinasi:

- FIM;
- rule;
- integrasi analisis file;
- deteksi perilaku mencurigakan;
- integrasi tool open source tambahan.

Rujukan Wazuh mencatat beberapa metode malware detection, termasuk perilaku rootkit, rule dan FIM, integrasi YARA, serta integrasi analisis file.

Untuk buku ini, pembahasan malware dibuat aman dan edukatif. Tidak perlu menjalankan malware nyata.

5. Active Response

Active response memungkinkan Wazuh menjalankan tindakan ketika kondisi tertentu terpenuhi.

Contoh:

- menulis log response;
- menjalankan script lokal;
- memblokir sumber serangan dalam lab;
- memberi tanda pada endpoint.

Namun active response harus hati-hati.

> **Respons otomatis tanpa validasi dapat merusak layanan.**

Untuk mahasiswa, active response hanya dilakukan di lab tertutup.

6. Compliance

Wazuh menyediakan fitur untuk membantu pemantauan compliance dan konfigurasi keamanan. Rujukan Wazuh menyebut dashboard dapat menampilkan regulatory compliance, vulnerability, file integrity, dan configuration assessment. (Wazuh)

Untuk mahasiswa, compliance dipahami sebagai:

- checklist kontrol keamanan;
- bukti konfigurasi;
- bukti monitoring;
- bukti perubahan;
- bukti respons.

Compliance bukan sekadar memenuhi aturan. Compliance adalah disiplin membuktikan bahwa kontrol keamanan berjalan.

3.7 Wazuh sebagai Pondasi AI untuk SOC

AI untuk SOC tidak boleh berdiri di udara. AI membutuhkan data yang bersih, terstruktur, dan dapat ditelusuri. Wazuh memberi fondasi itu melalui alert JSON, rule ID, timestamp, agent, severity, dan field log.

AI open source dapat membantu:

- merangkum alert;
- menjelaskan pola login gagal;
- membuat draft laporan;
- membantu klasifikasi severity;
- menyusun timeline;
- memberi saran langkah triage;
- membantu membuat query;
- membantu membaca alert JSON.

Namun keputusan tetap milik analis.

> **AI boleh mempercepat analisis, tetapi tidak boleh menggantikan bukti.**

Prinsip penggunaan AI dalam Bab 3:

- jangan kirim data sensitif ke layanan eksternal;
- gunakan data dummy untuk latihan;
- hapus username, IP publik, token, dan informasi rahasia;
- validasi semua output AI dengan log asli;
- simpan prompt sebagai bagian dari dokumentasi.

3.8 Desain Lab Wazuh Single-Node

Untuk mahasiswa, desain awal yang paling realistis adalah **single-node**. Artinya, Wazuh server, indexer, dan dashboard berjalan pada satu server lab.

Komponen

- 1 VM SOC server.
- 1 endpoint Linux.
- 1 mesin simulasi terkendali.
- 1 jaringan lab internal.
- Docker atau instalasi langsung sesuai kebutuhan.
- Browser hanya dari mesin admin lab.

Alur Lab

```
Endpoint Linux
     ↓
Wazuh Agent
     ↓
SOC Server
     ├── Wazuh Server
     ├── Wazuh Indexer
     └── Wazuh Dashboard
```

Kelebihan Single-Node

- mudah dipasang;
- cocok untuk mahasiswa;
- hemat resource;
- mudah di-reset;
- cukup untuk belajar agent, alert, FIM, rule, dan API.

Keterbatasan Single-Node

- tidak cocok untuk produksi besar;
- resource terbatas;
- jika server mati, semua komponen berhenti;
- tidak mengajarkan cluster sejak awal.

Untuk buku ini, single-node cukup. Tujuannya adalah **belajar alur SOC**, bukan membangun data center SOC.

3.9 Hands-on 1 — Instalasi Wazuh Server Single-Node

Tujuan

Memasang komponen pusat Wazuh pada SOC server.

Prasyarat

Pastikan dari Bab 2:

- VM SOC server aktif;
- RAM cukup;
- storage cukup;
- IP server dicatat;
- sistem sudah diperbarui;
- snapshot awal sudah dibuat;
- jaringan lab terisolasi.

Perintah Instalasi Cepat

Dokumentasi Wazuh menyediakan *quickstart* untuk memasang komponen pusat Wazuh pada host yang sama menggunakan *installation assistant*. Dokumentasi itu juga menyatakan Wazuh bersifat free dan open source dengan lisensi GNU GPLv2 dan Apache License 2.0 untuk komponennya. (Dokumentasi Wazuh)

Contoh alur instalasi:

```
curl -sO
https://packages.wazuh.com/4.12/wazuh-install.sh
sudo bash ./wazuh-install.sh -a
```

Penjelasan:

- `curl` mengambil file dari alamat web.
- `-sO` berarti mode senyap dan menyimpan file dengan nama aslinya.
- `sudo` menjalankan perintah dengan hak admin.
- `bash` menjalankan script shell.
- `./wazuh-install.sh` adalah script instalasi.
- `-a` berarti memasang komponen utama secara otomatis pada satu host.

Setelah instalasi selesai, simpan informasi akses dashboard di folder aman:

```
mkdir -p ~/soc-lab/configs/wazuh
nano ~/soc-lab/configs/wazuh/akses-dashboard.txt
```

Penjelasan:

- `mkdir -p` membuat folder jika belum ada.
- `nano` membuka editor teks di terminal.
- file ini dipakai untuk catatan lokal lab.

Catatan keamanan: jangan unggah file kredensial ke repository publik.

3.10 Hands-on 2 — Cek Status Service Wazuh

Setelah instalasi, cek service utama.

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

Penjelasan:

- `systemctl` mengelola service.
- `status` melihat kondisi service.
- `wazuh-manager` adalah komponen server analisis.
- `wazuh-indexer` menyimpan dan mengindeks alert.
- `wazuh-dashboard` menyediakan antarmuka visual.

Jika service belum aktif:

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

Penjelasan:

- `restart` memulai ulang service.
- gunakan setelah perubahan konfigurasi atau jika service bermasalah.

3.11 Hands-on 3 — Menambahkan Endpoint Linux

Tujuan

Menghubungkan endpoint Linux ke Wazuh server.

Dokumentasi Wazuh menjelaskan bahwa deployment agent dapat dilakukan melalui dashboard pada menu *Agents management* lalu *Deploy new agent*. (Dokumentasi Wazuh)

Langkah Umum

1. Masuk ke Wazuh dashboard.
2. Buka menu agent management.
3. Pilih deploy new agent.
4. Pilih sistem endpoint Linux.
5. Masukkan alamat Wazuh server.
6. Salin perintah instalasi agent.
7. Jalankan di endpoint.
8. Start agent.
9. Cek status agent di dashboard.

Perintah Cek Agent

Pada endpoint:

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

Penjelasan:

- `sudo` menjalankan perintah sebagai admin.
- `systemctl` mengelola service.
- `status` melihat keadaan service.
- `wazuh-agent` adalah agent endpoint.

Jika agent belum aktif:

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

Penjelasan:

- `restart` memulai ulang agent.
- gunakan setelah konfigurasi berubah.

3.12 Membaca Agent di Dashboard

Setelah agent aktif, buka dashboard.

Yang perlu diperiksa:

- nama agent;
- IP agent;
- status aktif;
- sistem operasi;
- waktu terakhir terkoneksi;
- jumlah alert;
- group agent.

Pertanyaan analisis:

- Apakah agent sudah aktif?
- Apakah IP sesuai inventory?
- Apakah nama host jelas?
- Apakah agent mengirim event?
- Apakah ada alert awal?
- Apakah ada error komunikasi?

Agent aktif belum tentu monitoring sudah benar.
SOC harus memastikan data benar-benar masuk.

3.13 Hands-on 4 — Mengaktifkan FIM untuk File Penting

Pada endpoint Linux, FIM dapat digunakan untuk memonitor file penting. Contoh:

```
/etc/ssh/sshd_config
```

Tujuan

Mendeteksi perubahan file konfigurasi penting.

Konsep

FIM menjawab:

- file apa yang berubah?
- kapan berubah?
- siapa yang mengubah?
- apa path file?
- apakah perubahan ini sah?

Contoh Konfigurasi FIM

Pada agent, edit konfigurasi agent sesuai dokumentasi Wazuh FIM. Konsepnya adalah menambahkan direktori atau file yang ingin dipantau. Dokumentasi Wazuh menjelaskan bahwa FIM memiliki pengaturan global dan pengaturan spesifik endpoint untuk memonitor file dan direktori. (Dokumentasi Wazuh)

Contoh bentuk konfigurasi:

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

Penjelasan:

- `<syscheck>` adalah blok konfigurasi FIM.
- `<directories>` menentukan file atau direktori yang dipantau.
- `check_all="yes"` meminta pemeriksaan atribut penting.
- `/etc/ssh/sshd_config` adalah file konfigurasi yang dipantau.

Restart agent:

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

Uji perubahan secara aman:

```
sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak
sudo touch /etc/ssh/sshd_config
```

Penjelasan:

- `cp` menyalin file.
- file `.bak` adalah backup.
- `touch` mengubah timestamp file tanpa mengubah isi besar.
- ini cukup untuk memicu perubahan sederhana.

Cek dashboard pada bagian FIM atau alert terkait.

3.14 Membaca Alert dalam Format JSON

SOC modern harus terbiasa membaca JSON. Dashboard membantu, tetapi JSON memberi bukti lebih detail.

Contoh struktur alert sederhana:

```
{
  "timestamp": "2026-01-01T10:00:00Z",
  "agent": {
    "name": "linux-endpoint-01",
    "ip": "10.10.10.21"
  },
  "rule": {
    "id": "550",
    "level": 7,
    "description": "File modified"
  },
  "syscheck": {
    "path": "/etc/ssh/sshd_config",
    "event": "modified"
  }
}
```

Penjelasan:

- `timestamp` adalah waktu kejadian.
- `agent` menunjukkan endpoint sumber.
- `name` adalah nama agent.
- `ip` adalah alamat agent.
- `rule` berisi informasi rule.
- `id` adalah nomor rule.
- `level` adalah tingkat alert.
- `description` menjelaskan alert.
- `syscheck` adalah bagian FIM.
- `path` menunjukkan file.
- `event` menunjukkan jenis perubahan.

Pertanyaan analis:

- file apa yang berubah?
- endpoint mana yang berubah?
- kapan berubah?
- apakah perubahan ini sah?
- apakah ada alert lain di waktu berdekatan?

3.15 Hands-on 5 — Membuat Custom Rule Sederhana

Custom rule penting karena setiap organisasi memiliki konteks berbeda. Wazuh memungkinkan pengguna membuat rule khusus untuk pola log yang relevan dengan kebutuhan lokal. (Dokumentasi Wazuh)

Tujuan

Membuat rule sederhana untuk mendeteksi pola login gagal pada endpoint Linux.

Lokasi Rule Lokal

Pada Wazuh server, custom rule umumnya diletakkan pada:

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

Contoh Rule Sederhana

```
<group name="local,authentication,">
  <rule id="100100" level="8">
    <if_sid>5716</if_sid>
    <description>Login gagal terdeteksi pada endpoint
Linux</description>
    <group>authentication_failed,</group>
  </rule>
</group>
```

Penjelasan:

- `<group>` mengelompokkan rule.
- `name="local,authentication,"` memberi kategori rule.
- `<rule>` membuka definisi rule.
- `id="100100"` adalah ID rule lokal.
- `level="8"` adalah tingkat kepentingan alert.
- `<if_sid>5716</if_sid>` berarti rule ini bergantung pada rule sebelumnya.
- `<description>` adalah deskripsi alert.
- `<group>` tambahan memudahkan filter.

Restart Wazuh manager:

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

Penjelasan:

- restart diperlukan agar rule baru dibaca.
- cek error jika service gagal aktif.

Uji Rule

Gunakan simulasi login gagal di lab lokal. Jangan melakukan pengujian ke sistem publik.

Setelah muncul alert:

- buka dashboard;
- filter berdasarkan rule ID;
- cek agent;
- cek waktu;
- cek log asli;
- catat hasil di laporan.

3.16 Hands-on 6 — Menguji Rule dengan Wazuh Logtest

Wazuh menyediakan tool `wazuh-logtest` untuk menguji decoder dan rule terhadap contoh log. Dokumentasi resmi menjelaskan bahwa `/var/ossec/bin/wazuh-logtest` dapat dipakai melalui dashboard, command line, atau API untuk memasukkan event log, melihat decoder yang cocok, dan melihat alert yang cocok. (Dokumentasi Wazuh)

Jalankan:

```
sudo /var/ossec/bin/wazuh-logtest
```

Penjelasan:

- `sudo` menjalankan dengan hak admin.
- `/var/ossec/bin/wazuh-logtest` adalah tool pengujian rule dan decoder.

Masukkan contoh log autentikasi gagal dari endpoint Linux. Hasil yang perlu diperhatikan:

- decoder apa yang cocok;
- rule apa yang cocok;
- level alert;
- deskripsi alert;
- group alert.

Jangan membuat rule langsung di produksi tanpa logtest.
Rule yang buruk akan menambah noise.

3.17 Dasar Wazuh API

Wazuh API penting untuk integrasi. Dengan API, mahasiswa dapat mengambil data agent, alert, status manager, dan informasi lain secara otomatis.

Dokumentasi Wazuh menjelaskan bahwa Wazuh server API dapat dipakai untuk tugas seperti menambah agent, restart manager atau agent, mengambil event FIM, serta mengelola user, role, dan permission. (Dokumentasi Wazuh)

API berguna untuk:

- integrasi dengan Python;
- integrasi dengan SOAR;
- pengambilan alert;
- pembuatan laporan otomatis;
- otomasi triage;
- eksperimen AI untuk SOC.

Prinsip Keamanan API

- gunakan HTTPS;
- jangan menyimpan password sembarangan;
- gunakan token;
- batasi akses;
- jangan unggah token ke repository publik;
- gunakan user khusus lab;
- rotasi kredensial jika bocor.

Wazuh API membutuhkan autentikasi dan setiap call harus menyertakan JSON Web Token. (Dokumentasi Wazuh)

3.18 Hands-on 7 — Mengambil Token API dengan curl

Contoh alur umum:

```
TOKEN=$(curl -k -u wazuh-user:wazuh-password \
  -X GET
"https://127.0.0.1:55000/security/user/authenticate?raw=true"
)
```

Penjelasan:

- `TOKEN=$(...)` menyimpan hasil perintah ke variabel `TOKEN`.
- `curl` mengirim request HTTP.
- `-k` mengizinkan koneksi HTTPS dengan sertifikat lab/self-signed.
- `-u` mengirim username dan password.
- `-X GET` memakai metode GET.
- URL mengarah ke endpoint autentikasi API.
- `raw=true` meminta token dalam bentuk sederhana.

Catatan: ganti `wazuh-user` dan `wazuh-password` dengan kredensial lab. Jangan memakai kredensial default untuk jangka panjang.

Cek agent:

```
curl -k -X GET "https://127.0.0.1:55000/agents?pretty=true" \
  -H "Authorization: Bearer $TOKEN"
```

Penjelasan:

- `-H` menambahkan header.
- `Authorization: Bearer $TOKEN` mengirim token.
- `/agents` meminta daftar agent.
- `pretty=true` membuat output lebih mudah dibaca.

3.19 Hands-on 8 — Mengakses Wazuh API dengan Python

Python dipakai untuk mengambil data dari API. Ini fondasi untuk otomasi dan AI SOC.

Contoh script:

```
import requests
from requests.auth import HTTPBasicAuth

wazuh_url = "https://127.0.0.1:55000"
username = "wazuh-user"
password = "wazuh-password"

auth_response = requests.get(
    f"{wazuh_url}/security/user/authenticate?raw=true",
    auth=HTTPBasicAuth(username, password),
    verify=False
)

token = auth_response.text

agents_response = requests.get(
    f"{wazuh_url}/agents?pretty=true",
    headers={"Authorization": f"Bearer {token}"},
    verify=False
)

print(agents_response.text)
```

Penjelasan:

- `import requests` memanggil library untuk HTTP request.
- `HTTPBasicAuth` dipakai untuk autentikasi username dan password.
- `wazuh_url` menyimpan alamat API Wazuh.
- `username` menyimpan nama user API.
- `password` menyimpan password API.
- `requests.get()` mengirim request GET.
- `f"{wazuh_url}/..."` membuat URL lengkap.
- `auth=HTTPBasicAuth(...)` mengirim kredensial.
- `verify=False` dipakai untuk sertifikat lab/self-signed.
- `token = auth_response.text` menyimpan token.
- `headers` mengirim token dalam format Bearer.
- `print()` menampilkan hasil.

Catatan keamanan: `verify=False` hanya untuk lab. Untuk sistem serius, gunakan sertifikat yang benar.

Dokumentasi Wazuh juga menyediakan contoh autentikasi API menggunakan Python untuk memperoleh JWT dan mengambil informasi dasar Wazuh serta ringkasan status agent. (Dokumentasi Wazuh)

3.20 Membaca Alert untuk Analisis SOC

Ketika alert muncul, jangan langsung menyimpulkan terjadi serangan. Ikuti pola triage sederhana.

Checklist Triage Alert

- **Apa rule ID-nya?**
- **Berapa level alert?**
- **Agent mana yang mengirim alert?**
- **Kapan alert terjadi?**
- **Apa log aslinya?**
- **Apakah ada alert lain di waktu berdekatan?**
- **Apakah aktivitas ini normal?**
- **Apakah perlu dibuat case?**
- **Apakah perlu respons?**
- **Apakah perlu tuning rule?**

Rujukan SOC menekankan pentingnya *log and event analysis*, deteksi anomali, integrasi log dengan aktivitas SOC lain, serta rekonstruksi rantai serangan.

> **Triage yang baik lebih penting daripada alert yang banyak.**

3.21 Kesalahan Umum Saat Belajar Wazuh

Beberapa kesalahan yang sering terjadi:

- memasang Wazuh tanpa memahami arsitektur;
- tidak mencatat kredensial dengan aman;
- tidak membuat snapshot sebelum instalasi;
- agent aktif tetapi tidak mengirim log;
- dashboard dibuka ke jaringan yang tidak perlu;
- rule dibuat terlalu sensitif;
- tidak menguji rule dengan logtest;
- terlalu cepat memakai active response;
- tidak membaca JSON;
- tidak menulis laporan hasil eksperimen.

SOC yang baik tidak hanya “berhasil install”. SOC yang baik bisa menjawab:

alert ini artinya apa, buktinya apa, dampaknya apa, responsnya apa?

3.22 Mini Template Laporan Praktikum Bab 3

Gunakan template berikut.

```
Judul Praktikum:
Instalasi Wazuh dan Monitoring Endpoint Linux

Tujuan:
Membangun fondasi SIEM/XDR open source menggunakan Wazuh.

Komponen:
- SOC Server:
- Endpoint Linux:
- Jaringan Lab:
- Versi Wazuh:

Langkah:
1.
2.
3.

Hasil:
- Agent terdaftar:
- Alert muncul:
- FIM aktif:
- API berhasil diakses:

Bukti:
- Screenshot dashboard:
- Potongan alert JSON:
- Output API:

Analisis:
- Alert apa yang muncul?
- Apakah alert valid?
- Apa bukti teknisnya?

Kesimpulan:
- Apa yang berhasil?
- Apa kendala?
- Apa perbaikan berikutnya?
```

3.23 Rangkuman Bab

Bab ini menjelaskan Wazuh sebagai fondasi SOC open source. Wazuh menyatukan kemampuan SIEM dan XDR, mengumpulkan log dari endpoint, menganalisis event, menghasilkan alert, menyimpan data, dan menampilkannya melalui dashboard.

Mahasiswa belajar bahwa Wazuh terdiri dari agent, server, indexer, dan dashboard. Mahasiswa juga memahami konsep log, event, alert, decoder, rule, group, dan agent. Konsep ini penting sebelum masuk ke deteksi serangan yang lebih kompleks pada Bab 4.

Bagian praktik memperkenalkan instalasi Wazuh single-node, penambahan endpoint Linux, aktivasi FIM, custom rule sederhana, pengujian rule dengan logtest, serta akses API menggunakan `curl` dan Python.

3.24 Output Bab

Setelah menyelesaikan Bab 3, pembaca diharapkan mampu:

- menjelaskan arsitektur Wazuh;
- membedakan Wazuh server, indexer, dashboard, dan agent;
- menjelaskan konsep log, event, alert, decoder, rule, group, dan agent;
- memasang Wazuh single-node di lab;
- menambahkan endpoint Linux;
- membaca alert di dashboard;
- membaca alert dalam format JSON;
- mengaktifkan FIM untuk file penting;
- membuat custom rule sederhana;
- menguji rule dengan Wazuh logtest;
- mengambil data dari Wazuh API menggunakan `curl`;
- mengambil data dari Wazuh API menggunakan Python;
- memahami bahwa AI SOC harus memakai data Wazuh sebagai bukti, bukan menggantikan analisis.

Pesan akhir Bab 3:
Wazuh memberi SOC mata, telinga, dan memori; tetapi analis tetap harus memberi konteks, validasi, dan keputusan.

Bab 4 — Deteksi Serangan dengan Wazuh: Brute Force, Malware, FIM, dan Vulnerability

> **Deteksi yang baik bukan alert paling banyak, tetapi alert yang dapat dibuktikan, dijelaskan, dan ditindaklanjuti.**

4.1 Arah Bab

Bab ini mengubah Wazuh dari sekadar sistem yang sudah terpasang menjadi **mesin deteksi SOC**. Pada Bab 3, pembaca sudah memahami arsitektur Wazuh, agent, rule, decoder, alert, JSON, dan API. Pada Bab 4, kemampuan itu dipakai untuk mendeteksi aktivitas nyata di lab.

Arah buku dari dokumen utama menempatkan Bab 4 sebagai bab deteksi: *brute force*, perubahan file, deteksi malware, *vulnerability detection*, *active response*, custom rule, dan ekspor alert JSON.

Karena buku ini diarahkan untuk **open source SOC**, seluruh praktik difokuskan pada lingkungan Linux, Wazuh, YARA, FIM, JSON, API, dan lab tertutup. Komponen yang bersifat proprietary atau terkait produk tertentu tidak dipakai sebagai fondasi praktik.

4.2 Mengapa Deteksi Lebih Penting daripada Dashboard

Dashboard hanya menampilkan data. Deteksi menjawab pertanyaan yang lebih penting:

- **apakah aktivitas ini normal?**
- **apakah ini tanda serangan?**
- **apakah alert ini valid?**
- **apa buktinya?**
- **apa dampaknya?**
- **apa tindakan yang aman?**

Rujukan SOC menekankan pentingnya *log and event analysis*, deteksi anomali, pengelolaan log, teknik analisis, dan kemampuan melihat pola dalam data event.

SOC yang matang tidak mengejar jumlah alert. SOC yang matang mengejar **kualitas deteksi**.

Deteksi berkualitas memiliki ciri:

- alert jelas;
- rule dapat dijelaskan;
- bukti log tersedia;
- severity masuk akal;
- false positive rendah;
- tindakan lanjut jelas;
- dapat dipetakan ke MITRE ATT&CK;
- dapat ditulis dalam laporan.

4.3 Dari Event ke Alert ke Incident

Mahasiswa perlu membedakan tiga istilah dasar.

Event

Event adalah kejadian yang tercatat.

Contoh:

- user login;
- service restart;
- file berubah;
- koneksi masuk;
- paket diperbarui;
- proses berjalan.

Event belum tentu buruk. Banyak event adalah aktivitas normal.

Alert

Alert adalah event yang dianggap penting oleh rule.

Contoh:

- login gagal berulang;
- file konfigurasi penting berubah;
- file mencurigakan muncul;
- vulnerability ditemukan;
- aktivitas tidak biasa terdeteksi.

Alert perlu dianalisis. Alert belum tentu incident.

Incident

Incident adalah kejadian keamanan yang sudah divalidasi dan membutuhkan respons.

Contoh:

- akun benar-benar disalahgunakan;
- file penting diubah tanpa izin;
- endpoint menunjukkan indikasi kompromi;
- vulnerability kritikal aktif dieksploitasi;
- pola serangan berulang dari sumber yang sama.

Rujukan karier SOC menekankan bahwa SOC bertanggung jawab atas triage, investigasi, dan respons terhadap insiden keamanan; deteksi datang lebih awal, lalu alert di triage, diselidiki, dan direspons.

Tidak semua event menjadi alert. Tidak semua alert menjadi incident.

4.4 Cara Kerja Deteksi Berbasis Rule

Wazuh menggunakan rule untuk menentukan apakah sebuah event perlu menjadi alert. Rule membaca log yang sudah diproses oleh decoder, lalu mengecek apakah log tersebut cocok dengan pola tertentu.

Alur sederhananya:

```
Log mentah
   ↓
Decoder
   ↓
Field terstruktur
   ↓
Rule
   ↓
Alert
   ↓
Triage analis
```

Dalam dokumentasi resmi, Wazuh rule ditulis dalam XML dan digunakan oleh Wazuh manager untuk mendeteksi pola atau perilaku tertentu dalam log. Rule dapat menghasilkan alert ketika kondisi terpenuhi. (Dokumentasi Wazuh)

Deteksi berbasis rule cocok untuk:

- login gagal;
- perubahan file;
- service tidak wajar;
- akses ke path sensitif;
- pola scanning;
- aktivitas malware berbasis indikator;
- konfigurasi lemah;
- vulnerability tertentu.

Namun rule juga punya kelemahan:

- bisa terlalu sensitif;
- bisa terlalu longgar;
- bisa menghasilkan *noise*;
- bisa gagal membaca teknik baru;
- perlu tuning terus-menerus.

Karena itu, SOC membutuhkan **rule + konteks + analisis manusia + threat intelligence + AI open source sebagai asisten**.

4.5 File Integrity Monitoring sebagai Deteksi Dasar

File Integrity Monitoring atau **FIM** adalah kemampuan untuk memantau perubahan file dan direktori penting. Dokumentasi Wazuh menjelaskan bahwa FIM memonitor integritas file sistem dan aplikasi, membantu mendeteksi perubahan pada aset sensitif yang dapat mengindikasikan kompromi. (Dokumentasi Wazuh)

FIM penting karena banyak serangan meninggalkan jejak pada file:

- konfigurasi berubah;
- file baru muncul;
- permission berubah;
- file sensitif dimodifikasi;
- script tidak dikenal ditambahkan;
- binary diganti;
- file log dihapus.

Contoh file penting di Linux:

- `/etc/ssh/sshd_config`
- `/etc/passwd`
- `/etc/group`
- `/etc/sudoers`
- `/etc/crontab`
- `/etc/systemd/system/`
- `/var/www/`
- `/opt/`

FIM bukan hanya untuk melihat “file berubah”. FIM membantu analis bertanya:

- siapa yang mengubah?
- kapan berubah?
- file apa yang berubah?
- apakah perubahan ini sah?
- apakah perubahan terjadi dekat dengan alert lain?
- apakah perubahan ini bagian dari rantai serangan?

4.6 Deteksi Brute Force

Brute force adalah percobaan login berulang menggunakan banyak kombinasi username dan password. Dalam SOC, pola ini sering terlihat dari:

- login gagal berulang;
- sumber IP yang sama mencoba berkali-kali;
- banyak username dicoba;
- waktu percobaan sangat rapat;
- akses terjadi di luar jam normal.

Dokumentasi Wazuh memiliki *proof of concept* untuk mendeteksi serangan *brute force* dan menunjukkan bahwa Wazuh memiliki rule bawaan untuk mengidentifikasi pola tersebut. (Dokumentasi Wazuh)

Dalam buku ini, simulasi dilakukan **hanya di lab tertutup**. Tujuannya bukan menyerang, tetapi menghasilkan alert agar mahasiswa belajar membaca bukti.

Indikator yang Dicari

- banyak kegagalan login;
- username tidak dikenal;
- percobaan dari sumber yang sama;
- percobaan berulang pada layanan yang sama;
- alert muncul dalam waktu singkat.

Pertanyaan Analis

- Apakah sumber IP berasal dari lab?
- Apakah username valid?
- Apakah ada login berhasil setelah banyak gagal?
- Apakah host target kritikal?
- Apakah perlu dibuat case?
- Apakah perlu respons otomatis atau manual?

Bahaya terbesar brute force bukan login gagal. Bahaya terbesar adalah login berhasil setelah banyak gagal.

4.7 Deteksi Malware Sederhana dengan YARA dan FIM

Deteksi malware dalam lab tidak perlu menjalankan malware nyata. Untuk pembelajaran, mahasiswa dapat memakai file dummy, hash dummy, atau rule YARA sederhana yang mendeteksi pola teks tertentu.

YARA adalah tool open source untuk mengidentifikasi dan mengklasifikasikan artefak malware berbasis pola tekstual atau biner. Dokumentasi Wazuh menjelaskan bahwa FIM dapat dikombinasikan dengan YARA untuk mendeteksi file yang ditambahkan atau dimodifikasi pada endpoint. (Dokumentasi Wazuh)

Dalam pendekatan aman:

- FIM memantau folder tertentu;
- jika ada file baru atau berubah, Wazuh memicu pemeriksaan;
- YARA memeriksa isi file berdasarkan rule;
- alert muncul jika pola cocok;
- analis membaca alert dan memvalidasi.

Contoh Folder Latihan

Gunakan folder aman:

```
/home/soc-lab/suspicious-files/
```

Jangan memakai folder produksi. Jangan memakai sampel malware nyata untuk kelas dasar.

Prinsip Aman Deteksi Malware

- gunakan file dummy;
- gunakan rule YARA sederhana;
- jangan menjalankan file mencurigakan;
- jangan mengunduh malware dari internet;
- jangan memakai data sensitif;
- semua pengujian dilakukan di VM terisolasi.

4.8 Vulnerability Detection

Vulnerability detection membantu SOC melihat kelemahan pada endpoint. Wazuh dapat memanfaatkan inventaris endpoint untuk memeriksa paket, versi aplikasi, dan potensi CVE yang relevan. Rujukan Wazuh terlampir juga menempatkan *vulnerability detection* sebagai salah satu kemampuan penting Wazuh untuk security monitoring.

Tujuan vulnerability detection:

- mengetahui endpoint yang rentan;
- memprioritaskan perbaikan;
- menghubungkan alert dengan risiko;
- membantu laporan keamanan;
- mencegah serangan sebelum terjadi.

Vulnerability detection bukan pengganti patch management. Ia membantu SOC memberi visibilitas:

- host mana yang lemah;
- paket apa yang bermasalah;
- severity apa yang muncul;
- apakah kelemahan sudah lama;
- apakah kelemahan terjadi di aset penting.

Cara Membaca Hasil

Jangan hanya membaca jumlah vulnerability. Baca konteks:

- apakah severity tinggi?
- apakah endpoint kritikal?
- apakah layanan terbuka?
- apakah vulnerability dapat dieksploitasi dari jaringan?
- apakah sudah ada alert terkait host tersebut?
- apakah ada mitigasi sementara?

Vulnerability yang tidak diprioritaskan hanya menjadi daftar panjang yang tidak ditindaklanjuti.

4.9 Active Response: Potensi dan Risiko

Active response adalah kemampuan Wazuh untuk menjalankan tindakan ketika trigger tertentu terpenuhi. Dokumentasi Wazuh menjelaskan bahwa modul Active Response membantu tim keamanan mengotomasi tindakan respons berdasarkan trigger tertentu, termasuk mengurangi alert fatigue dan menyediakan script respons bawaan. (Dokumentasi Wazuh)

Contoh respons:

- mencatat event tambahan;
- memblokir sumber tertentu di lab;
- menjalankan script pemeriksaan;
- mengirim sinyal ke workflow automation;
- melakukan isolasi terbatas pada skenario terkendali.

Namun active response berisiko.

Risiko Active Response

- false positive menyebabkan blokir salah;
- layanan penting terganggu;
- bukti hilang karena tindakan terlalu cepat;
- respons otomatis sulit diaudit;
- rule yang buruk bisa memicu tindakan berulang;
- mahasiswa salah memahami otomasi sebagai “selalu benar”.

Karena itu, prinsipnya:

> **Untuk awal, active response harus berjalan dalam mode lab, terbatas, tercatat, dan mudah dibatalkan.**

Dokumentasi Wazuh juga menyediakan contoh active response untuk memblokir sumber *brute force* pada lab SSH. (Dokumentasi Wazuh)

4.10 Mapping Alert ke MITRE ATT&CK

MITRE ATT&CK membantu analis menghubungkan alert dengan taktik dan teknik adversary. Dokumentasi Wazuh menjelaskan bahwa integrasi MITRE ATT&CK memungkinkan alert Wazuh dipetakan ke taktik dan teknik tertentu, sehingga tim keamanan lebih mudah memahami sifat ancaman dan menyusun mitigasi. (Dokumentasi Wazuh)

Contoh mapping sederhana:

Aktivitas	Kemungkinan Taktik	Makna SOC
Login gagal berulang	*Credential Access*	Ada percobaan menebak kredensial
Login berhasil setelah banyak gagal	*Initial Access*	Perlu investigasi segera
File konfigurasi berubah	*Persistence* / *Defense Evasion*	Ada potensi perubahan untuk bertahan atau menyembunyikan aktivitas
File mencurigakan muncul	*Execution* / *Defense Evasion*	Perlu validasi artefak
Paket rentan ditemukan	*Resource Development* / risiko eksposur	Perlu patch atau mitigasi

MITRE bukan untuk gaya-gayaan. MITRE membantu SOC menjawab:

- teknik apa yang mungkin dipakai?
- tahap serangan apa yang sedang terjadi?
- apakah ada pola berulang?
- apa mitigasi yang relevan?
- apa prioritas investigasi?

Wazuh juga menjelaskan bahwa modul MITRE ATT&CK dapat membantu *threat hunting* dengan memetakan TTP ke event yang dihasilkan. (Dokumentasi Wazuh)

4.11 Peran AI Open Source dalam Deteksi Wazuh

AI dalam Bab 4 dipakai sebagai **asisten analisis**, bukan pengambil keputusan.

AI open source dapat membantu:

- merangkum alert JSON;
- menjelaskan arti rule ID;
- menyusun timeline dari beberapa alert;
- membantu klasifikasi awal;
- membuat draft laporan incident;
- menyarankan pertanyaan triage;
- membantu mapping MITRE;
- membantu mendeteksi pola berulang dari alert.

Namun AI tidak boleh:

- menggantikan log asli;
- memutuskan incident tanpa bukti;
- mengeksekusi active response otomatis;
- menerima data sensitif tanpa sanitasi;
- membuat kesimpulan tanpa validasi analisis.

Rujukan SOC analyst menekankan bahwa manual analysis tetap penting karena memberi kedalaman dan konteks yang sering tidak ditangkap automation.

> **AI mempercepat membaca. Analis tetap bertanggung jawab membuktikan.**

4.12 Hands-on 1 — Simulasi Login Gagal di Lab Tertutup

Tujuan

Menghasilkan alert login gagal secara aman untuk latihan triage.

Lingkungan

- 1 SOC server dengan Wazuh aktif.
- 1 endpoint Linux dengan Wazuh agent aktif.
- 1 mesin simulasi dalam jaringan lab.
- Tidak ada koneksi ke sistem publik.

Langkah Aman

1. Pastikan endpoint Linux sudah terdaftar di Wazuh.
2. Pastikan layanan login jarak jauh di endpoint lab hanya terbuka di jaringan lab.
3. Dari mesin simulasi, lakukan beberapa percobaan login gagal secara manual.
4. Tunggu alert muncul di Wazuh dashboard.
5. Catat waktu, host, user, sumber, dan rule ID.
6. Jangan melakukan pengujian ke alamat di luar lab.

Yang Dicatat

- waktu percobaan;
- IP sumber lab;
- IP endpoint;
- username yang dicoba;
- jumlah kegagalan;
- rule ID Wazuh;
- level alert;
- log asli.

Pertanyaan Analisis

- Apakah alert muncul?
- Apakah alert sesuai aktivitas?
- Apakah severity masuk akal?
- Apakah ada alert turunan?
- Apakah perlu tuning rule?

4.13 Hands-on 2 — Membaca Alert Brute Force di Wazuh

Setelah alert muncul, baca bukan hanya judulnya. Buka detail alert.

Field penting:

```
timestamp
agent.name
agent.ip
rule.id
rule.level
rule.description
data.srcip
data.user
location
full_log
```

Makna field:

- `timestamp` menunjukkan waktu.
- `agent.name` menunjukkan endpoint.
- `agent.ip` menunjukkan alamat endpoint.
- `rule.id` menunjukkan rule yang memicu alert.
- `rule.level` menunjukkan tingkat kepentingan.
- `rule.description` menjelaskan alert.
- `data.srcip` menunjukkan sumber.
- `data.user` menunjukkan username.
- `location` menunjukkan sumber log.
- `full_log` menunjukkan log mentah.

Jangan menganalisis alert tanpa membaca full_log.

4.14 Hands-on 3 — Custom Rule Login Gagal

Tujuan

Membuat rule lokal untuk memperjelas alert login gagal di lab.

Lokasi rule lokal:

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

Contoh rule:

```
<group name="local,authentication,">
  <rule id="100200" level="8">
    <if_sid>5716</if_sid>
    <description>Login gagal berulang pada endpoint
Linux lab</description>
    <group>authentication_failed,lab_detection,</group>
  </rule>
</group>
```

Penjelasan:

- `<group>` mengelompokkan rule.
- `local` menunjukkan rule lokal.
- `authentication` menunjukkan kategori autentikasi.
- `rule id="100200"` adalah ID rule lokal.
- `level="8"` memberi tingkat alert.
- `<if_sid>5716</if_sid>` membuat rule turunan dari rule autentikasi gagal.
- `<description>` memberi deskripsi yang mudah dibaca.
- `lab_detection` memudahkan filter dashboard.

Restart manager:

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

Uji dengan logtest:

```
sudo /var/ossec/bin/wazuh-logtest
```

Wazuh menyediakan `wazuh-logtest` untuk menguji decoder dan rule terhadap contoh log sebelum rule dipakai aktif. (Dokumentasi Wazuh)

4.15 Hands-on 4 — Mengaktifkan FIM untuk File Penting

Tujuan

Mendeteksi perubahan file konfigurasi penting.

File contoh:

```
/etc/ssh/sshd_config
```

Contoh konfigurasi FIM:

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

Penjelasan:

- `<syscheck>` adalah modul FIM.
- `<directories>` menentukan file atau direktori yang dipantau.
- `check_all="yes"` meminta pemeriksaan atribut penting.
- path file menunjukkan target monitoring.

Restart agent:

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

Uji perubahan aman:

```
sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak
sudo touch /etc/ssh/sshd_config
```

Penjelasan:

- `cp` membuat backup.
- `touch` mengubah timestamp file.
- perubahan ini aman untuk latihan dasar.

Cek alert FIM di dashboard. Dokumentasi Wazuh menjelaskan bahwa FIM dapat mendeteksi penambahan, perubahan, dan penghapusan file. (Dokumentasi Wazuh)

4.16 Hands-on 5 — Deteksi File Dummy dengan YARA

Tujuan

Menguji konsep malware detection tanpa memakai malware nyata.

Buat Folder Latihan

```
mkdir -p ~/soc-lab/suspicious-files\
```

Penjelasan:

- `mkdir` membuat folder.
- `-p` memastikan tidak error jika folder sudah ada.
- folder ini menjadi tempat file uji.

Buat File Dummy

```
echo "SOC_LAB_TEST_SIGNATURE" >
~/soc-lab/suspicious-files/sample.txt
```

Penjelasan:

- `echo` menulis teks.
- `>` menyimpan output ke file.
- `SOC_LAB_TEST_SIGNATURE` adalah pola dummy.
- `sample.txt` adalah file uji, bukan malware.

Buat Rule YARA Sederhana

rule SOC_Lab_Test_File

```
{
  strings:
    $a = "SOC_LAB_TEST_SIGNATURE"

  condition:
    $a
}
```

Penjelasan:

- `rule` memberi nama rule.
- `strings` berisi pola yang dicari.
- `$a` adalah variabel pola.
- `condition` menentukan kapan rule cocok.
- `$a` berarti cocok jika pola ditemukan.

Wazuh mendukung integrasi YARA untuk memindai file yang ditambahkan atau dimodifikasi pada endpoint. (Dokumentasi Wazuh)

Catatan Keamanan

- Jangan memakai malware nyata.
- Jangan menjalankan file mencurigakan.
- Jangan menguji di mesin produksi.
- Gunakan file dummy.
- Fokus pada alur deteksi.

4.17 Hands-on 6 — Mengaktifkan Vulnerability Detection

Tujuan

Melihat daftar kelemahan pada endpoint lab.

Langkah umum:

1. Pastikan Wazuh agent aktif.
2. Pastikan inventaris sistem terkirim.
3. Aktifkan modul vulnerability sesuai dokumentasi versi Wazuh yang digunakan.
4. Tunggu proses sinkronisasi.
5. Buka dashboard vulnerability.
6. Filter berdasarkan endpoint.
7. Catat severity tertinggi.
8. Buat rekomendasi mitigasi.

Yang dicatat:

```
hostname
package
installed_version
cve
severity
published_date
recommended_action
```

Pertanyaan analisis:

- vulnerability mana yang paling kritis?
- apakah endpoint adalah aset penting?
- apakah layanan terkait terbuka?
- apakah patch tersedia?
- apakah mitigasi sementara diperlukan?
- apakah vulnerability terkait alert lain?

Rujukan Wazuh terlampir menempatkan vulnerability detector dan vulnerability security configuration management sebagai bagian penting dari penggunaan Wazuh.

4.18 Hands-on 7 — Active Response Sederhana di Lab

Tujuan

Membuat respons terbatas untuk kasus login gagal berulang.

Peringatan: lakukan hanya di lab tertutup.

Prinsip active response:

- trigger jelas;
- rule jelas;
- tindakan terbatas;
- durasi blokir terbatas;
- log respons dicatat;
- mudah dibatalkan.

Contoh skenario:

```
Jika login gagal berulang dari satu sumber lab,
maka buat tindakan respons terbatas:
- catat sumber,
- beri label incident candidate,
- blokir sementara di endpoint lab,
- tulis log respons.
```

Jangan gunakan respons otomatis untuk sistem penting sebelum:

- rule diuji;
- false positive rendah;
- ada approval;
- ada dokumentasi;
- ada prosedur rollback.

Otomasi SOC harus bisa dijelaskan dan dibatalkan.

4.19 Hands-on 8 — Ekspor Alert JSON untuk Laporan

Tujuan

Mengambil alert sebagai bukti laporan.

Contoh struktur laporan JSON:

```
{
  "case_id": "SOC-LAB-004-001",
  "title": "Login gagal berulang pada endpoint Linux",
  "timestamp": "2026-01-01T10:00:00Z",
  "agent_name": "linux-endpoint-01",
  "source_ip": "10.10.10.50",
  "rule_id": "100200",
  "severity": 8,
  "event_type": "authentication_failed",
  "mitre_tactic": "Credential Access",
  "status": "under_review",
  "analyst_note": "Alert valid dihasilkan dari simulasi lab
tertutup."
}
```

Penjelasan:

- `case_id` adalah nomor kasus.
- `title` adalah judul singkat.
- `timestamp` adalah waktu alert.
- `agent_name` adalah endpoint.
- `source_ip` adalah sumber aktivitas.
- `rule_id` adalah rule yang memicu.
- `severity` adalah tingkat alert.
- `event_type` adalah jenis kejadian.
- `mitre_tactic` adalah mapping taktik.
- `status` adalah status analisis.
- `analyst_note` adalah catatan analis.

JSON ini dapat digunakan untuk:

- laporan praktikum;
- input ke AI lokal;
- integrasi SOAR;
- case management;
- dashboard;
- dokumentasi proyek akhir.

4.20 Template Analisis Alert

Gunakan template ringkas berikut.

```
Judul Alert:
Waktu:
Endpoint:
Sumber:
Rule ID:
Severity:
Jenis Event:

Log Mentah:
...

Analisis:
- Apa yang terjadi?
- Apakah alert valid?
- Apa bukti pendukung?
- Apakah ada alert lain?
- Apakah perlu dibuat case?

Mapping MITRE:
- Taktik:
- Teknik:

Tindakan:
- Monitor
- Eskalasi
- Blokir sementara di lab
- Tuning rule
- Buat case

Kesimpulan:
...
```

Template ini melatih mahasiswa agar tidak hanya melihat dashboard, tetapi menulis analisis.

4.21 Kesalahan Umum dalam Deteksi

Kesalahan yang sering terjadi:

- menganggap semua alert adalah incident;
- tidak membaca log mentah;
- tidak memeriksa konteks endpoint;
- tidak membedakan simulasi lab dan serangan nyata;
- membuat rule terlalu sensitif;
- mengaktifkan respons otomatis terlalu cepat;
- tidak menguji rule dengan logtest;
- tidak menulis bukti;
- tidak mapping MITRE;
- terlalu percaya AI tanpa validasi log.

Deteksi tanpa validasi hanya menghasilkan kepanikan.

4.22 Peran AI Lokal untuk Membantu Bab Ini

AI open source dapat dipakai setelah alert JSON diekspor.

Contoh prompt aman:

```
Anda adalah asisten SOC analyst.
Analisis alert JSON berikut.
Jelaskan:
1. ringkasan kejadian,
2. indikator penting,
3. kemungkinan false positive,
4. mapping MITRE,
5. rekomendasi langkah berikutnya.

Jangan membuat kesimpulan di luar data.
Jika data tidak cukup, tulis "data belum cukup".
```

Gunakan AI untuk:

- mempercepat ringkasan;
- membuat draft laporan;
- menyusun pertanyaan triage;
- membantu mahasiswa belajar membaca alert.

Jangan gunakan AI untuk:

- menjalankan respons otomatis;
- memutuskan blokir;
- menyimpan token;
- membaca data sensitif tanpa sanitasi.

AI adalah mentor tambahan, bukan pengganti analis.

4.23 Rangkuman Bab

Bab ini menjelaskan bagaimana Wazuh dipakai untuk mendeteksi aktivitas penting dalam SOC lab. Pembaca belajar membedakan event, alert, dan incident; memahami deteksi berbasis rule; mengaktifkan FIM; membaca alert *brute force*; membuat custom rule; memahami deteksi malware aman dengan YARA; mengaktifkan vulnerability detection; memahami risiko active response; dan mengekspor alert JSON untuk laporan.

Rujukan Wazuh terlampir menekankan bahwa Wazuh mendukung threat detection melalui log analysis, intrusion detection, FIM, malware detection, vulnerability detection, custom rule, decoder, API, active response, dan integrasi keamanan lain.

Inti Bab 4 adalah sederhana:

> **SOC yang baik bukan hanya menerima alert, tetapi mampu membuktikan alert.**

4.24 Output Bab

Setelah menyelesaikan Bab 4, pembaca diharapkan mampu:

- menjelaskan cara kerja deteksi berbasis rule;
- membedakan event, alert, dan incident;
- membaca alert login gagal di Wazuh;
- membuat custom rule sederhana;
- mengaktifkan FIM untuk file penting;
- memahami deteksi file mencurigakan dengan YARA dan FIM;
- membaca hasil vulnerability detection;
- memahami potensi dan risiko active response;
- mapping alert ke MITRE ATT&CK;
- mengekspor alert dalam format JSON;
- membuat laporan analisis alert singkat;
- memakai AI open source sebagai asisten analis, bukan pengganti bukti.

Pesan akhir Bab 4:
Deteksi adalah seni membaca bukti; semakin rapi log, rule, dan konteks, semakin cepat SOC membedakan noise dari ancaman nyata.

Bab 5 — Network Security Monitoring dengan Security Onion, Suricata, dan Zeek

> **Endpoint memberi tahu apa yang terjadi di mesin; jaringan memberi tahu bagaimana serangan bergerak. SOC yang kuat harus membaca keduanya.**

5.1 Arah Bab

Bab ini melengkapi Wazuh dengan **visibilitas jaringan**. Pada Bab 3 dan Bab 4, pembaca sudah belajar membaca endpoint, log, FIM, rule, alert, dan vulnerability. Namun SOC tidak boleh hanya melihat endpoint. Banyak serangan meninggalkan jejak penting di jaringan.

Outline buku menempatkan Bab 5 sebagai bab *network security monitoring* dengan **Security Onion, Suricata, dan Zeek**. Fokusnya adalah membaca trafik jaringan, alert Suricata, log Zeek, DNS, HTTP, PCAP, dan alur investigasi dari jaringan.

Security Onion secara resmi disebut sebagai platform gratis dan terbuka untuk *network visibility*, *host visibility*, *intrusion detection*, *honeypot*, *log management*, dan *case management*. Di sisi jaringan, Security Onion memakai Suricata untuk deteksi berbasis signature, Zeek atau Suricata untuk metadata protokol dan ekstraksi file, serta full packet capture untuk investigasi. (Dokumentasi Security Onion)

5.2 Mengapa Log Endpoint Saja Tidak Cukup?

Endpoint log penting, tetapi tidak selalu cukup. Ada serangan yang terlihat jelas dari endpoint, tetapi ada juga yang lebih mudah dibaca dari jaringan.

Contoh:

- port scanning;
- koneksi ke domain mencurigakan;
- transfer data besar;
- komunikasi ke IP asing;
- percobaan akses aplikasi web;
- DNS query tidak wajar;
- beaconing berkala;
- koneksi lateral antar host;
- trafik terenkripsi yang tetap terlihat dari metadata.

Endpoint menjawab:

- proses apa berjalan?
- file apa berubah?
- user apa login?
- paket apa terpasang?
- konfigurasi apa berubah?

Jaringan menjawab:

- siapa berbicara dengan siapa?
- lewat port apa?
- protokol apa?
- berapa lama koneksi?
- berapa banyak data dikirim?
- domain apa ditanyakan?
- pola koneksi normal atau aneh?

Rujukan SOC menempatkan *network traffic analysis* sebagai salah satu kompetensi penting SOC, termasuk *network traffic monitoring*, *packet analysis*, *protocol inspection*, NIDS/NIPS, dan network forensics.

> **SOC yang hanya melihat endpoint mudah buta terhadap gerakan di jaringan.**

5.3 Konsep Network Security Monitoring

Network Security Monitoring atau **NSM** adalah proses mengumpulkan, menganalisis, dan menginterpretasikan data jaringan untuk menemukan tanda ancaman.

NSM tidak selalu berarti menyimpan semua paket. NSM dapat memakai beberapa jenis data:

- **alert** — sinyal deteksi dari rule;
- **flow** — ringkasan koneksi;
- **metadata protokol** — DNS, HTTP, TLS, file, user agent;
- **PCAP** — paket jaringan lengkap;
- **log sensor** — catatan operasi sensor;
- **file extraction** — file yang terlihat dari trafik;
- **threat intelligence** — konteks tambahan dari IOC.

Tujuan NSM:

- melihat trafik yang mencurigakan;
- mendeteksi serangan jaringan;
- menghubungkan endpoint dan jaringan;
- memperkuat bukti insiden;
- membantu *threat hunting*;
- membangun timeline serangan.

NSM bukan sekadar “menangkap paket”. NSM adalah **cara membaca percakapan digital antar mesin**.

5.4 NIDS dan NIPS

Dalam jaringan, ada dua istilah penting: **NIDS** dan **NIPS**.

NIDS

Network Intrusion Detection System atau **NIDS** memantau trafik jaringan dan menghasilkan alert jika menemukan pola mencurigakan.

Contoh yang dicari:

- port scanning;
- eksploit aplikasi;
- koneksi ke domain mencurigakan;
- malware callback;
- trafik aneh pada protokol tertentu;
- pola serangan web.

NIDS bersifat mengamati dan memberi alert.

NIPS

Network Intrusion Prevention System atau **NIPS** tidak hanya mendeteksi, tetapi juga dapat memblokir atau mencegah trafik tertentu.

Untuk lab mahasiswa, pendekatan utama adalah **NIDS**, bukan NIPS. Deteksi lebih aman daripada pencegahan otomatis. Pencegahan yang salah dapat memutus layanan dan membuat analisis kacau.

Belajar SOC dimulai dari melihat, bukan langsung memblokir.

5.5 Peran Security Onion

Security Onion dipakai sebagai distribusi SOC jaringan. Ia menggabungkan beberapa komponen penting dalam satu platform.

Fungsi utama Security Onion:

- *network visibility*;
- *host visibility*;
- *intrusion detection*;
- *log management*;
- *case management*;
- *threat hunting*;
- dashboard;
- PCAP;
- alert;
- integrasi sensor.

Dokumentasi Security Onion menjelaskan bahwa platform ini menyediakan visibilitas jaringan melalui Suricata untuk deteksi berbasis signature, Zeek atau Suricata untuk metadata protokol dan ekstraksi file, serta full packet capture. (Dokumentasi Security Onion)

Dalam arsitektur terdistribusi, Security Onion dapat memiliki **manager node**, **sensor node**, dan **search node**. Pendekatan ini memberi skalabilitas karena sensor dan log source dapat ditambah sesuai kebutuhan. (Dokumentasi Security Onion)

Untuk mahasiswa, cukup mulai dari mode sederhana:

- 1 VM Security Onion;
- 1 interface management;
- 1 interface monitor;
- 1 jaringan lab;
- 1 mesin target;
- 1 mesin simulasi.

5.6 Peran Suricata

Suricata adalah engine deteksi jaringan open source. Dalam SOC, Suricata berperan sebagai sensor yang membaca trafik dan menghasilkan alert.

Suricata digunakan untuk:

- *signature-based detection*;
- membaca rule;
- membuat alert;
- menghasilkan log JSON;
- mendeteksi anomali;
- membaca metadata protokol;
- mendukung investigasi trafik.

Dokumentasi Suricata menjelaskan bahwa EVE JSON output menghasilkan alert, anomali, metadata, informasi file, dan catatan protokol spesifik dalam format JSON. (docs.suricata.io)

Konsep Rule Suricata

Rule Suricata menentukan pola apa yang dicari dalam trafik. Rule dapat melihat:

- protokol;
- IP sumber;
- IP tujuan;
- port sumber;
- port tujuan;
- arah trafik;
- payload;
- metadata;
- pesan alert;
- klasifikasi;
- severity.

Rule membantu SOC mengubah trafik mentah menjadi alert yang dapat dianalisis.

Eve JSON

Eve JSON adalah output penting dari Suricata. Format JSON membuat data mudah dikirim ke SIEM, OpenSearch, pipeline log, SOAR, atau AI lokal.

Contoh field yang umum:

```
{
  "timestamp": "2026-01-01T10:00:00.000000+0000",
  "event_type": "alert",
  "src_ip": "10.10.10.50",
  "src_port": 44512,
  "dest_ip": "10.10.10.30",
  "dest_port": 80,
  "proto": "TCP",
  "alert": {
    "signature": "Lab possible web attack",
    "category": "Attempted Information Leak",
    "severity": 2
  }
}
```

Maknanya:

- `timestamp` adalah waktu event;
- `event_type` menunjukkan jenis data;
- `src_ip` adalah IP sumber;
- `dest_ip` adalah IP tujuan;
- `proto` adalah protokol;
- `alert.signature` adalah nama alert;
- `alert.category` adalah kategori;
- `alert.severity` adalah tingkat prioritas.

Eve JSON adalah jembatan penting antara sensor jaringan dan SOC analytics.

5.7 Peran Zeek

Zeek bukan sekadar IDS berbasis signature. Zeek lebih kuat sebagai mesin **metadata jaringan** dan *protocol analysis*.

Zeek membantu menjawab:

- koneksi apa yang terjadi?
- protokol apa digunakan?
- DNS apa yang ditanyakan?
- HTTP request apa yang muncul?
- file apa terlihat?
- berapa lama koneksi?
- berapa byte dikirim?
- apakah pola koneksi normal?

Dokumentasi Zeek menunjukkan bahwa `conn.log` merekam percakapan antar sistem, termasuk protokol, port, durasi, dan informasi koneksi. (docs.zeek.org)

Dokumentasi Zeek juga menjelaskan bahwa `dns.log` menangkap aktivitas resolusi nama pada level aplikasi selama trafik DNS dapat terlihat, dan analis harus membaca log berdasarkan pertanyaan investigasi yang ingin dijawab. (docs.zeek.org)

Log Penting Zeek

Log yang penting untuk SOC:

- `conn.log` — ringkasan koneksi;
- `dns.log` — query DNS;
- `http.log` — aktivitas HTTP;
- `ssl.log` atau `tls.log` — metadata TLS;
- `files.log` — informasi file;
- `notice.log` — event penting dari Zeek;
- `weird.log` — anomali protokol.

Mengapa Zeek Penting?

Suricata bagus untuk alert. Zeek bagus untuk konteks.

Contoh:

> Suricata memberi alert:
> Possible web attack detected

Zeek membantu menjawab:

- koneksi dari IP mana?
- ke host mana?
- URL apa diakses?
- metode HTTP apa?
- user agent apa?
- berapa lama koneksi?
- DNS apa yang mendahului koneksi?
- apakah ada koneksi lain dari host yang sama?

Suricata berteriak ketika ada pola bahaya. Zeek menjelaskan percakapannya.

5.8 Sensor Interface dan Traffic Mirroring

Sensor jaringan membutuhkan akses ke trafik. Tanpa trafik, sensor tidak membaca apa-apa.

Biasanya ada dua interface:

Management Interface

Interface ini dipakai untuk:

- mengakses dashboard;
- update sistem;
- SSH admin lab;
- komunikasi manajemen;
- mengirim log.

Management interface memiliki IP.

Monitor Interface

Interface ini dipakai untuk:

- membaca trafik;
- menerima salinan paket;
- menjalankan Suricata;
- menjalankan Zeek;
- melihat komunikasi antar mesin.

Monitor interface sering tidak diberi IP. Ia hanya mendengar trafik.

Konsep penting:

```
Management interface = untuk mengelola sensor
Monitor interface    = untuk membaca trafik
```

Untuk lab VM, trafik dapat diarahkan melalui:

- jaringan internal;
- bridge lab;
- virtual switch;
- port mirroring;
- TAP virtual;
- arsitektur gateway sensor.

Kesalahan umum: sensor dipasang, tetapi monitor interface tidak melihat trafik. Akibatnya dashboard kosong, bukan karena tool gagal, tetapi karena arsitektur jaringan salah.

5.9 PCAP, Alert, Flow, dan Pivot Investigasi

Dalam NSM, analis membaca beberapa jenis data.

Alert

Alert adalah sinyal awal. Biasanya berasal dari Suricata.

Pertanyaan:

- apa signature-nya?
- severity berapa?
- IP sumber dan tujuan apa?
- port apa?
- rule apa?
- kapan terjadi?

Flow

Flow adalah ringkasan koneksi. Biasanya dibaca dari Zeek `conn.log`.

Pertanyaan:

- koneksi dari mana ke mana?
- protokol apa?
- durasi berapa?
- byte dikirim berapa?
- apakah koneksi berhasil?
- apakah banyak koneksi berulang?

PCAP

PCAP adalah paket jaringan lengkap. Ini bukti paling detail, tetapi berat.

Dipakai jika:

- alert penting;
- perlu bukti teknis;
- perlu melihat payload;
- perlu rekonstruksi komunikasi;
- metadata belum cukup.

Pivot Investigasi

Pivot berarti berpindah dari satu data ke data lain.

Contoh alur pivot:

```
Alert Suricata
   ↓
src_ip
   ↓
Zeek conn.log
   ↓
dns.log
   ↓
http.log
   ↓
PCAP
   ↓
Case note
```

Pivot membantu SOC tidak berhenti pada satu alert.

> **Alert memberi pintu masuk. Pivot memberi cerita lengkap.**

5.10 Hubungan Security Onion, Suricata, Zeek, Wazuh, dan OpenSearch

Dalam buku ini, tools tidak berdiri sendiri. Masing-masing punya peran.

Wazuh

Fokus:

- endpoint;
- log host;
- FIM;
- vulnerability;
- rule endpoint;
- agent;
- API.

Security Onion

Fokus:

- jaringan;
- sensor;
- Suricata;
- Zeek;
- PCAP;
- threat hunting jaringan.

Suricata

Fokus:

- alert jaringan;
- rule;
- signature;
- Eve JSON.

Zeek

Fokus:

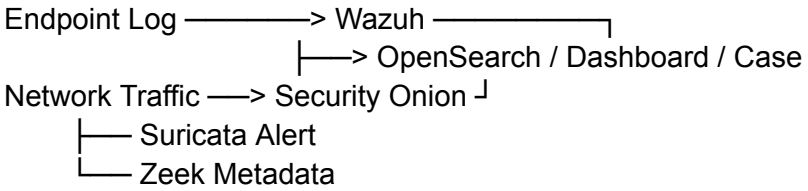
- metadata jaringan;
- koneksi;
- DNS;
- HTTP;
- protokol;
- konteks investigasi.

OpenSearch

Fokus:

- indexing;
- query;
- dashboard;
- korelasi log;
- analitik.

Arsitektur konseptual:



Tujuan akhirnya adalah korelasi:

- alert endpoint + alert jaringan;
- FIM + koneksi keluar;
- login gagal + port scan;
- vulnerability + eksploit trafik;
- DNS mencurigakan + proses endpoint;
- alert JSON + laporan AI lokal.

5.11 Peran AI Open Source dalam Network Monitoring

AI open source dapat membantu membaca banyak log jaringan. Namun AI harus diberi data yang sudah disaring, bukan semua trafik mentah.

AI dapat membantu:

- merangkum alert Suricata;
- menjelaskan field Eve JSON;
- menyusun timeline koneksi;
- membaca pola dari `conn.log`;
- mengelompokkan DNS query;
- membuat catatan investigasi;
- menyusun laporan singkat;
- menyarankan pertanyaan lanjutan.

AI tidak boleh:

- menerima PCAP berisi data sensitif tanpa sanitasi;
- membuat kesimpulan tanpa bukti;
- mengganti analisis Zeek/Suricata;
- menjalankan respons otomatis tanpa approval;
- menyimpan IP publik, kredensial, atau data pribadi tanpa kontrol.

Prinsipnya:

> **AI membantu membaca volume data. Analis tetap memutuskan berdasarkan bukti.**

5.12 Hands-on 1 — Desain VM Security Onion

Tujuan

Menyiapkan desain VM Security Onion untuk lab tertutup.

Spesifikasi Awal

Untuk latihan dasar:

- CPU: 4 core atau lebih.
- RAM: 12–16 GB atau lebih.
- Storage: 150 GB atau lebih.
- 2 interface jaringan:
    - management;
    - monitor.
- Jaringan lab terisolasi.

Topologi Sederhana

```
Host / Admin
    |
Management Network
    |
Security Onion
    |
Monitor Interface
    |
Lab Network
    ├── Endpoint Linux
    ├── Server Lab
    └── Mesin Simulasi
```

Catatan

- Management interface memiliki IP.
- Monitor interface membaca trafik lab.
- Dashboard hanya diakses dari jaringan admin.
- Jangan arahkan trafik publik ke lab.
- Buat snapshot setelah instalasi awal.

5.13 Hands-on 2 — Instalasi Security Onion di VM

Tujuan

Memasang Security Onion sebagai sensor dan platform monitoring jaringan.

Dokumentasi Security Onion menyediakan panduan instalasi berbasis ISO dan pengaturan melalui setup wizard. Security Onion didesain sebagai platform terbuka untuk monitoring dan pertahanan jaringan. (Dokumentasi Security Onion)

Langkah Ringkas

1. Unduh ISO Security Onion dari situs resmi.
2. Buat VM baru.
3. Tambahkan dua interface:
    - interface 1 untuk management;
    - interface 2 untuk monitor.
4. Pasang Security Onion.
5. Jalankan setup wizard.
6. Pilih mode lab/standalone.
7. Tentukan management interface.
8. Tentukan monitor interface.
9. Buat user admin lab.
10. Selesaikan setup.
11. Akses dashboard dari jaringan management.
12. Buat snapshot.

Validasi Awal

Cek:

- dashboard bisa dibuka;
- sensor aktif;
- Suricata berjalan;
- Zeek berjalan;
- interface monitor benar;
- log mulai masuk;
- tidak ada trafik keluar yang tidak perlu.

5.14 Hands-on 3 — Simulasi Port Scanning di Lab

Tujuan

Menghasilkan trafik jaringan yang dapat dibaca sensor.

Peringatan: hanya lakukan di lab milik sendiri.

Skenario

- Mesin simulasi melakukan scanning ke server lab.
- Security Onion membaca trafik.
- Suricata menghasilkan alert.
- Zeek mencatat koneksi.
- Mahasiswa membuat catatan investigasi.

Langkah Aman

1. Pastikan semua mesin berada di jaringan lab.
2. Pastikan sensor melihat trafik.
3. Jalankan scanning ringan ke server lab.
4. Buka dashboard Security Onion.
5. Cari alert terkait scanning.
6. Buka detail alert.
7. Catat IP sumber, IP tujuan, port, waktu, dan signature.
8. Cari koneksi terkait di log Zeek.

Pertanyaan Analisis

- IP mana yang melakukan scanning?
- Target mana yang discan?
- Port apa yang disentuh?
- Berapa lama aktivitas berlangsung?
- Apakah Suricata memberi alert?
- Apakah Zeek mencatat koneksi?
- Apakah perlu dibuat case?

5.15 Hands-on 4 — Membaca Alert Suricata

Alert Suricata harus dibaca dengan disiplin.

Field penting:

```
timestamp
event_type
src_ip
src_port
dest_ip
dest_port
proto
alert.signature
alert.category
alert.severity
flow_id
```

Makna:

- `timestamp` menunjukkan waktu alert;
- `event_type` biasanya `alert`;
- `src_ip` adalah sumber trafik;
- `dest_ip` adalah tujuan;
- `proto` menunjukkan protokol;
- `alert.signature` memberi nama deteksi;
- `alert.category` memberi kategori;
- `alert.severity` menunjukkan prioritas;
- `flow_id` membantu korelasi event dalam satu flow.

Dokumentasi Suricata menjelaskan bahwa field `flow_id` membantu mengorelasikan data EVE dari protokol, flow, alert, fileinfo, dan anomaly log pada satu sesi atau flow. (docs.suricata.io)

Pertanyaan Triage

- Apakah signature relevan?
- Apakah IP sumber berasal dari lab?
- Apakah port tujuan terbuka?
- Apakah alert tunggal atau berulang?
- Apakah ada koneksi terkait di Zeek?
- Apakah ada endpoint alert di Wazuh?
- Apakah ini false positive?

5.16 Hands-on 5 — Membaca Log Zeek

Zeek memberikan konteks. Mulai dari `conn.log`.

Field Penting `conn.log`

```
ts
uid
id.orig_h
id.orig_p
id.resp_h
id.resp_p
proto
service
duration
orig_bytes
resp_bytes
conn_state
```

Makna:

- `ts` adalah waktu;
- `uid` adalah ID koneksi;
- `id.orig_h` adalah host sumber;
- `id.orig_p` adalah port sumber;
- `id.resp_h` adalah host tujuan;
- `id.resp_p` adalah port tujuan;
- `proto` adalah protokol;
- `service` adalah layanan yang dikenali;
- `duration` adalah lama koneksi;
- `orig_bytes` adalah byte dari sumber;
- `resp_bytes` adalah byte dari tujuan;
- `conn_state` adalah status koneksi.

Gunakan `conn.log` untuk menjawab:

- siapa menghubungi siapa?
- port apa dipakai?
- koneksi berhasil atau gagal?
- berapa banyak koneksi terjadi?
- apakah ada pola scanning?

5.17 Hands-on 6 — Membaca DNS dan HTTP Log

DNS Log

DNS log membantu melihat domain yang diminta endpoint.

Pertanyaan:

- domain apa yang ditanyakan?
- endpoint mana yang bertanya?
- jawaban DNS apa?
- apakah query berulang?
- apakah domain terlihat acak?
- apakah domain baru atau mencurigakan?

Dokumentasi Zeek menjelaskan bahwa DNS log menangkap aktivitas resolusi nama, selama trafik DNS terlihat dan tidak disembunyikan oleh mekanisme terenkripsi tertentu. (docs.zeek.org)

HTTP Log

HTTP log membantu melihat aktivitas web yang tidak terenkripsi atau metadata tertentu.

Pertanyaan:

- metode apa digunakan?
- host apa diakses?
- URI apa diminta?
- status code apa?
- user agent apa?
- apakah pola mirip scanning?
- apakah ada akses ke path sensitif?

Contoh catatan investigasi:

Endpoint 10.10.10.21 melakukan koneksi HTTP ke 10.10.10.30.
Zeek mencatat beberapa request ke path berbeda dalam waktu singkat.
Suricata memberi alert scanning.
Aktivitas sesuai simulasi lab dan bukan incident nyata.

5.18 Hands-on 7 — Mengambil Contoh Eve JSON

Tujuan

Membaca output Suricata dalam format JSON.

Contoh Eve JSON:


```
{
  "timestamp": "2026-01-01T10:15:20.000000+0000",
  "flow_id": 123456789,
  "event_type": "alert",
  "src_ip": "10.10.10.50",
  "src_port": 43122,
  "dest_ip": "10.10.10.30",
  "dest_port": 80,
  "proto": "TCP",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 200001,
    "rev": 1,
    "signature": "LAB Possible Port Scan",
    "category": "Attempted Information Leak",
    "severity": 2
  }
}
```


Analisis Singkat

- Ada alert dari Suricata.
- Sumber adalah `10.10.10.50`.
- Tujuan adalah `10.10.10.30`.
- Port tujuan adalah `80`.
- Aksi masih `allowed`, berarti ini deteksi, bukan blokir.
- Signature menunjukkan kemungkinan scanning.
- Severity `2` perlu ditinjau, tetapi belum otomatis incident.

Alert allowed tetap penting. Ia berarti sensor melihat aktivitas mencurigakan, walau belum diblokir.

5.19 Hands-on 8 — Catatan Investigasi Sederhana

Gunakan template berikut.

```
Judul:
Deteksi scanning jaringan pada lab SOC

Waktu:
...

Sumber:
IP:
Host:

Tujuan:
IP:
Layanan/Port:

Data Suricata:
Signature:
Category:
Severity:
Flow ID:

Data Zeek:
conn.log:
dns.log:
http.log:

Analisis:
- Aktivitas apa yang terjadi?
- Apakah sesuai simulasi lab?
- Apakah ada koneksi berhasil?
- Apakah ada target lain?
- Apakah ada endpoint alert di Wazuh?

Kesimpulan:
...

Rekomendasi:
- Monitor
- Buat case
- Tuning rule
- Tambahkan segmentasi
- Lanjutkan pivot ke PCAP
```

Template ini melatih mahasiswa menulis investigasi berbasis bukti, bukan opini.

5.20 Kesalahan Umum dalam Network Monitoring

Kesalahan umum:

- sensor tidak melihat trafik;
- monitor interface diberi IP dan salah konfigurasi;
- hanya membaca alert tanpa membaca Zeek;
- tidak mencatat waktu;
- tidak menyamakan zona waktu;
- tidak menghubungkan alert jaringan dengan alert endpoint;
- terlalu cepat menyimpulkan incident;
- tidak menyimpan PCAP penting;
- tidak membuat catatan investigasi;
- terlalu banyak rule tanpa tuning.

  Network monitoring yang buruk menghasilkan alert ramai tetapi konteks kosong.

5.21 Rangkuman Bab

Bab ini memperluas SOC dari endpoint ke jaringan. Wazuh tetap penting untuk endpoint, tetapi Security Onion, Suricata, dan Zeek memberi visibilitas terhadap trafik. Suricata membantu mendeteksi pola serangan melalui rule dan Eve JSON. Zeek membantu membaca metadata koneksi, DNS, HTTP, dan protokol lain. Security Onion menyatukan kemampuan ini dalam platform monitoring jaringan yang terbuka.

Inti Bab 5 adalah **korelasi**. Alert jaringan harus dihubungkan dengan log endpoint, dashboard, case, dan laporan. Mahasiswa harus belajar melakukan pivot: dari alert Suricata ke Zeek `conn.log`, DNS, HTTP, PCAP, lalu ke catatan investigasi.

Dengan pendekatan ini, SOC tidak hanya melihat “apa yang terjadi di mesin”, tetapi juga “bagaimana aktivitas bergerak di jaringan”.

5.22 Output Bab

Setelah menyelesaikan Bab 5, pembaca diharapkan mampu:

- menjelaskan mengapa endpoint log saja tidak cukup;
- memahami konsep *network security monitoring*;
- membedakan NIDS dan NIPS;
- menjelaskan peran Security Onion dalam SOC open source;
- menjelaskan peran Suricata sebagai engine deteksi jaringan;
- membaca dasar Eve JSON;
- menjelaskan peran Zeek sebagai penghasil metadata jaringan;
- memahami perbedaan alert, flow, metadata, dan PCAP;
- memahami konsep management interface dan monitor interface;
- melakukan simulasi scanning di lab tertutup;
- membaca alert Suricata;
- membaca log Zeek;
- menganalisis DNS dan HTTP log;
- membuat catatan investigasi trafik;
- menghubungkan data jaringan dengan Wazuh dan OpenSearch.

Pesan akhir Bab 5:
Serangan tidak hanya meninggalkan jejak di endpoint; ia juga meninggalkan percakapan di jaringan. SOC yang matang harus mampu membaca keduanya.

Bab 6 — OpenSearch untuk Log Management, Query, Dashboard, dan Alert

> **Log yang tidak bisa dicari, disaring, dan dijelaskan adalah beban; log yang terindeks dengan baik adalah bahan bakar utama SOC.**

6.1 Arah Bab

Bab ini fokus pada **data SOC**. Setelah Bab 3–5 membahas Wazuh, deteksi endpoint, Suricata, Zeek, dan monitoring jaringan, Bab 6 mengajarkan bagaimana log disimpan, dicari, divisualisasikan, dan diubah menjadi alert yang bisa dipakai analis.

Dalam outline buku, Bab 6 diarahkan untuk membahas **OpenSearch sebagai pusat log management, query, dashboard, alert, retensi log, normalisasi field, dan JSON sebagai bahasa pertukaran data SOC**. Output akhirnya adalah pembaca mampu membuat dashboard SOC sederhana dan query investigasi.

OpenSearch dipilih karena bersifat **open source**, berlisensi Apache 2.0, dan dirancang untuk *ingest*, *search*, *visualize*, dan *analyze data*. (OpenSearch)

6.2 Mengapa Log Perlu Diindeks?

SOC menghasilkan banyak data:

- log endpoint dari Wazuh;
- alert dari rule;
- hasil FIM;
- vulnerability;
- alert Suricata;
- metadata Zeek;
- log akses aplikasi;
- log sistem;
- catatan case;
- hasil enrichment threat intelligence.

Jika semua data hanya disimpan sebagai file mentah, analis akan kesulitan menjawab pertanyaan sederhana:

- host mana paling banyak menghasilkan alert?
- IP mana paling sering menjadi sumber serangan?
- rule ID mana paling sering muncul?
- severity tertinggi terjadi kapan?
- alert apa yang terkait satu host?
- apakah ada pola berulang?
- apakah ada lonjakan alert setelah konfigurasi berubah?

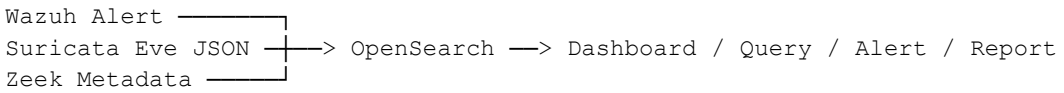
Indeks membuat log dapat dicari cepat. Dalam konteks SIEM, rujukan SOC menekankan bahwa konsolidasi log, normalisasi, query, korelasi, visualisasi, dan alert otomatis adalah kemampuan penting untuk memperkuat deteksi, investigasi, dan respons.

> **Tanpa indeks, SOC tenggelam dalam arsip. Dengan indeks, SOC bisa bertanya kepada data.**

6.3 OpenSearch dalam Arsitektur SOC

OpenSearch berperan sebagai mesin penyimpanan dan pencarian data. OpenSearch Dashboards menjadi antarmuka visual untuk eksplorasi data, pembuatan visualisasi, dashboard, dan observability. Dokumentasi OpenSearch Dashboards menyediakan fitur untuk membuat *index pattern*, mengelola data source, mengeksplorasi data, dan membangun visualisasi. (OpenSearch Documentation)

Dalam arsitektur SOC buku ini:



Peran OpenSearch:

- **menyimpan log terstruktur**;
- **membuat data mudah dicari**;
- **membantu korelasi event**;
- **membuat dashboard SOC**;
- **menjalankan alert berbasis query**;
- **menjadi sumber data untuk laporan**;
- **menjadi bahan input aman untuk AI lokal**.

Dalam Wazuh, beberapa indeks penting antara lain `wazuh-alerts`, `wazuh-archives`, `wazuh-monitoring`, dan `wazuh-statistics`. Rujukan Wazuh menjelaskan bahwa `wazuh-alerts` menyimpan alert dari Wazuh server, `wazuh-archives` menyimpan semua event termasuk yang tidak memicu alert, `wazuh-monitoring` menyimpan status agent, dan `wazuh-statistics` menyimpan data performa Wazuh server.

6.4 Struktur Data Log SOC

Log SOC harus punya struktur yang konsisten. Tanpa struktur, query menjadi sulit dan dashboard menjadi tidak stabil.

Field minimum yang dipakai dalam buku ini:

```
timestamp
host
source_ip
destination_ip
rule_id
severity
event_type
mitre_tactic
mitre_technique
```

1. timestamp

Menunjukkan waktu kejadian. Ini field paling penting untuk timeline.

Digunakan untuk:

- melihat urutan insiden;
- mencari lonjakan alert;
- menghubungkan endpoint dan jaringan;
- membuat grafik per jam/hari;
- menyusun laporan.

2. host

Menunjukkan mesin yang menghasilkan event.

Digunakan untuk:

- melihat endpoint paling berisiko;
- memfilter alert per host;
- menghubungkan alert dengan inventory aset.

3. source_ip

Menunjukkan sumber aktivitas.

Digunakan untuk:

- membaca sumber koneksi;
- mencari IP yang sering memicu alert;
- mendeteksi scanning;
- menghubungkan alert jaringan dan endpoint.

4. destination_ip

Menunjukkan tujuan aktivitas.

Digunakan untuk:

- melihat aset yang diserang;
- memantau layanan target;
- mengelompokkan alert per server.

5. rule_id

Menunjukkan rule yang memicu alert.

Digunakan untuk:

- tuning rule;
- melihat rule paling ramai;
- menilai kualitas deteksi;
- mengurangi *noise*.

6. severity

Menunjukkan tingkat prioritas.

Digunakan untuk:

- membuat triage;
- memilih alert kritikal;
- membuat dashboard prioritas;
- menentukan eskalasi.

7. event_type

Menunjukkan jenis event.

Contoh:

- `authentication_failed`
- `file_modified`
- `network_scan`
- `malware_indicator`
- `vulnerability_detected`
- `dns_query`
- `http_request`

8. mitre_tactic

Menunjukkan taktik MITRE.

Contoh:

- *Initial Access*;
- *Credential Access*;
- *Discovery*;
- *Persistence*;
- *Defense Evasion*.

9. mitre_technique

Menunjukkan teknik lebih spesifik.

Field ini membantu laporan menjadi lebih profesional karena analis tidak hanya menyebut “ada alert”, tetapi juga menjelaskan **taktik dan teknik adversary**.

6.5 Konsep Index, Mapping, Field, dan Query

Index

Index adalah tempat menyimpan dokumen log. Dalam SOC, satu index bisa berisi alert Wazuh, log Suricata, atau event lain.

Contoh:

```
soc-alerts-2026.01
wazuh-alerts-*
suricata-eve-*
zeek-conn-*
```

OpenSearch menyediakan *Create Index API* untuk membuat index dan dapat menentukan *settings*, *mappings*, serta *aliases*. (OpenSearch Documentation)

Mapping

Mapping adalah definisi tipe field. Misalnya:

- `timestamp` sebagai `date`;
- `host` sebagai `keyword`;
- `source_ip` sebagai `ip`;
- `severity` sebagai `integer`;
- `full_log` sebagai `text`.

OpenSearch dapat memakai *dynamic mapping* untuk mendeteksi field otomatis, tetapi *explicit mapping* lebih rapi karena tipe field ditentukan sejak awal. (OpenSearch Documentation)

Field

Field adalah elemen data dalam dokumen JSON.

Contoh:

```
{
  "host": "linux-endpoint-01",
  "severity": 8
}
```

`host` dan `severity` adalah field.

Untuk pencarian presisi, field seperti hostname, IP, event type, dan rule ID sebaiknya dipetakan sebagai `keyword` atau tipe yang sesuai. Dokumentasi OpenSearch menjelaskan bahwa tipe `keyword` cocok untuk string yang membutuhkan pencocokan persis dan bukan *full-text search*. (OpenSearch Documentation)

Query

Query adalah pertanyaan ke data.

Contoh pertanyaan SOC:

- tampilkan alert severity tinggi;
- cari login gagal;
- tampilkan top source IP;
- cari alert dari host tertentu;
- cari event dalam 1 jam terakhir;
- cari rule ID tertentu.

OpenSearch menyediakan *Query DSL* untuk melakukan pencarian dengan struktur JSON. (OpenSearch Documentation)

6.6 JSON sebagai Bahasa Pertukaran Data SOC

JSON menjadi penting karena banyak tools open source SOC menggunakannya:

- Wazuh alert;
- Suricata Eve JSON;
- API OpenSearch;
- API Wazuh;
- webhook SOAR;
- output enrichment;
- laporan otomatis;
- input AI lokal.

Contoh alert JSON sederhana:

```
{
  "timestamp": "2026-01-01T10:00:00Z",
  "host": "linux-endpoint-01",
  "source_ip": "10.10.10.50",
  "destination_ip": "10.10.10.21",
  "rule_id": "100200",
  "severity": 8,
  "event_type": "authentication_failed",
  "mitre_tactic": "Credential Access",
  "mitre_technique": "Brute Force"
}
```

JSON memudahkan integrasi karena formatnya:

- mudah dibaca;
- mudah dikirim lewat API;
- mudah diproses Python;
- mudah dimasukkan ke OpenSearch;
- mudah dipakai untuk dashboard;
- mudah disanitasi sebelum dipakai AI lokal.

SOC modern adalah ekosistem JSON.

6.7 Normalisasi Field

Normalisasi berarti menyamakan nama dan format field dari berbagai sumber.

Masalah umum:

- Wazuh memakai `agent.name`;
- Suricata memakai `src_ip`;
- Zeek memakai `id.orig_h`;
- log aplikasi memakai `client_ip`;
- script custom memakai `source`.

Jika tidak dinormalisasi, query menjadi rumit.

Contoh normalisasi:

Sumber	Field Asli	Field Normal
Wazuh	`agent.name`	`host`
Suricata	`src_ip`	`source_ip`
Zeek	`id.orig_h`	`source_ip`
Zeek	`id.resp_h`	`destination_ip`
Wazuh	`rule.id`	`rule_id`
Wazuh	`rule.level`	`severity`

Normalisasi membantu:

- membuat dashboard lintas sumber;
- menyederhanakan query;
- mengurangi kesalahan analisis;
- memudahkan korelasi;
- memudahkan AI lokal membaca data.

Rujukan SOC menekankan bahwa normalisasi menerjemahkan format log yang beragam ke format event umum, sehingga atribut dapat dibandingkan dan query korelasi dapat dibangun lebih mudah.

6.8 Retensi Log dan Kapasitas Storage

SOC sering gagal bukan karena deteksi buruk, tetapi karena storage habis. Log tumbuh cepat.

Faktor yang mempengaruhi storage:

- jumlah endpoint;
- jumlah sensor;
- volume trafik;
- jumlah rule aktif;
- banyaknya event non-alert;
- masa retensi;
- kompresi;
- replikasi;
- format log;
- jumlah field.

Prinsip retensi:

- simpan alert penting lebih lama;
- simpan raw event sesuai kebutuhan;
- pisahkan hot data dan archive data;
- hapus data yang tidak diperlukan;
- dokumentasikan kebijakan retensi;
- jangan menyimpan data sensitif tanpa alasan.

Rujukan SOC menekankan bahwa retensi dan storage harus menyeimbangkan kebutuhan audit, pencarian cepat, biaya, dan volume log yang terus bertambah.

Untuk lab mahasiswa:

- simpan data 7–14 hari;
- hapus index lama;
- jangan aktifkan semua log tanpa batas;
- gunakan dataset kecil;
- fokus pada pola analisis, bukan volume besar.

Retensi log adalah strategi, bukan sekadar kapasitas disk.

6.9 OpenSearch Dashboard untuk SOC

Dashboard bukan hiasan. Dashboard harus membantu analis menjawab pertanyaan cepat.

Dashboard SOC minimum:

- alert per host;
- alert per severity;
- top source IP;
- top rule ID;
- alert per MITRE tactic;
- tren alert per waktu;
- host dengan vulnerability tinggi;
- event login gagal;
- perubahan file penting.

OpenSearch Dashboards menyediakan alat visualisasi untuk menerjemahkan data besar menjadi grafik, diagram, atau representasi visual yang lebih mudah dipahami. (OpenSearch Documentation)

Dashboard yang baik harus:

- sederhana;
- tidak terlalu ramai;
- fokus pada keputusan;
- memakai filter waktu;
- bisa drill-down;
- bisa dipakai untuk laporan;
- mudah dijelaskan mahasiswa saat presentasi.

Dashboard SOC harus mengurangi kebingungan, bukan menambah dekorasi.

6.10 Alert Rule Berbasis Query

OpenSearch dapat membuat alert berbasis query. Dokumentasi OpenSearch Alerting menjelaskan bahwa alert dibuat dengan monitor yang berjalan terjadwal dan melakukan query ke index; trigger menentukan kondisi alert, sedangkan action menentukan apa yang dilakukan setelah alert aktif. (OpenSearch Documentation)

Komponen alerting:

- **monitor** — query yang berjalan berkala;
- **trigger** — kondisi pemicu;
- **action** — tindakan setelah alert;
- **schedule** — frekuensi pemeriksaan.

Contoh ide alert:

- jika login gagal lebih dari 10 kali dalam 5 menit;
- jika severity lebih dari 10 muncul;
- jika satu source IP menyerang banyak host;
- jika file penting berubah;
- jika alert MITRE *Credential Access* meningkat;
- jika endpoint kritikal tidak mengirim log.

Dokumentasi OpenSearch juga menjelaskan bahwa monitor dapat dipakai bersama alerting dan anomaly detection untuk memberi notifikasi ketika data melewati threshold tertentu. (OpenSearch Documentation)

Untuk lab mahasiswa, action cukup sederhana:

- tulis alert;
- tampilkan di dashboard;
- ekspor JSON;
- kirim ke workflow lokal pada bab SOAR;
- jangan otomatis melakukan blokir pada tahap awal.

6.11 Peran AI Open Source dalam Bab Ini

AI open source paling berguna ketika data sudah terstruktur.

AI dapat membantu:

- menjelaskan query;
- merangkum hasil pencarian;
- membuat laporan insiden dari JSON;
- mengelompokkan alert;
- menyarankan visualisasi;
- membantu membaca tren;
- membuat ringkasan untuk dosen atau pimpinan lab;
- membantu membuat *query template*.

AI tidak boleh:

- menerima kredensial;
- menerima log sensitif tanpa sanitasi;
- membuat keputusan incident tanpa bukti;
- menggantikan dashboard dan query;
- menyimpan data yang tidak perlu.

Prinsipnya:

> **OpenSearch mencari bukti. AI membantu menjelaskan bukti. Keputusan tetap milik analis.**

6.12 Hands-on 1 — Instalasi OpenSearch dan OpenSearch Dashboards

Tujuan

Menyiapkan OpenSearch dan OpenSearch Dashboards untuk menerima log SOC.

Pilihan Instalasi

Untuk lab mahasiswa, ada dua pendekatan:

- instalasi langsung pada VM;
- instalasi menggunakan Docker Compose.

Untuk buku ini, Docker Compose lebih cocok karena:

- mudah diulang;
- mudah dibersihkan;
- cocok untuk lab;
- konfigurasi bisa disimpan di folder proyek;
- mendukung portofolio Git.

Struktur Folder

```
mkdir -p ~/soc-lab/docker/opensearch
cd ~/soc-lab/docker/opensearch
```

Penjelasan:

- `mkdir` membuat folder;
- `-p` membuat folder bertingkat;
- `cd` masuk ke folder kerja.

Catatan Keamanan

- jangan buka dashboard ke internet;
- gunakan password kuat;
- batasi akses ke jaringan lab;
- jangan gunakan kredensial di repository publik;
- buat snapshot sebelum instalasi.

6.13 Hands-on 2 — Mengirim Sample Log JSON ke OpenSearch

Tujuan

Memasukkan sample alert SOC ke OpenSearch agar bisa dicari dan divisualisasikan.

Contoh Log JSON

Simpan sebagai `sample-alert.json`:

```
{
  "timestamp": "2026-01-01T10:00:00Z",
  "host": "linux-endpoint-01",
  "source_ip": "10.10.10.50",
  "destination_ip": "10.10.10.21",
  "rule_id": "100200",
  "severity": 8,
  "event_type": "authentication_failed",
  "mitre_tactic": "Credential Access",
  "mitre_technique": "Brute Force"
}
```

Kirim ke Index

```
curl -k -X POST
"https://localhost:9200/soc-alerts-2026/_doc" \
  -H "Content-Type: application/json" \
  -u admin:'password-lab-kuat' \
  -d @sample-alert.json
```

Penjelasan:

- `curl` mengirim request HTTP;
- `-k` dipakai untuk sertifikat lab;
- `-X POST` mengirim data baru;
- URL menunjuk index `soc-alerts-2026`;
- `-H` menentukan tipe konten JSON;
- `-u` mengirim username dan password lab;
- `-d @sample-alert.json` mengirim isi file JSON.

Catatan: ganti password sesuai konfigurasi lab.

6.14 Hands-on 3 — Membuat Index Pattern

Agar data dapat dibaca di dashboard, buat *index pattern*. Dokumentasi OpenSearch menjelaskan bahwa *index pattern* dibuat melalui menu Management → Dashboards Management → Index patterns, lalu memasukkan nama index dan memilih time field. (OpenSearch Documentation)

Langkah

1. Buka OpenSearch Dashboards dari jaringan lab.
2. Masuk ke menu Management.
3. Pilih Dashboards Management.
4. Pilih Index patterns.
5. Buat index pattern:

```
soc-alerts-*
```

6. Pilih field waktu:

```
timestamp
```

7. Simpan.
8. Buka Discover.
9. Pastikan dokumen sample muncul.

Validasi

Cek:

- data terlihat;
- field terbaca;
- timestamp benar;
- severity bertipe angka;
- IP terbaca sebagai field;
- event type dapat difilter.

6.15 Hands-on 4 — Query Dasar untuk SOC

Query membantu analis mencari bukti.

Cari Alert Severity Tinggi

```
{
  "query": {
    "range": {
      "severity": {
        "gte": 8
      }
    }
  }
}
```

Makna:

- `range` mencari nilai dalam rentang;
- `severity` adalah field tingkat alert;
- `gte` berarti lebih besar atau sama dengan;
- `8` adalah batas minimum.

Cari Login Gagal

```
{
  "query": {
    "term": {
      "event_type": "authentication_failed"
    }
  }
}
```

Makna:

- `term` mencari kecocokan persis;
- `event_type` adalah jenis event;
- `authentication_failed` adalah nilai yang dicari.

Cari Alert dari Host Tertentu

```
{
  "query": {
    "term": {
      "host": "linux-endpoint-01"
    }
  }
}
```

Cari Alert dari Source IP Tertentu

```
{
  "query": {
    "term": {
      "source_ip": "10.10.10.50"
    }
  }
}
```

Cari Alert MITRE Tactic Tertentu

```
{
  "query": {
    "term": {
      "mitre_tactic": "Credential Access"
    }
  }
}
```

Query adalah cara analisis bertanya kepada log.

6.16 Hands-on 5 — Membuat Dashboard SOC Sederhana

Visualisasi 1 — Alert per Host

Tujuan:

- melihat host paling ramai alert;
- menemukan endpoint bermasalah;
- memprioritaskan investigasi.

Field:

```
host
```

Jenis visualisasi:

- bar chart;
- data table.

Visualisasi 2 — Alert per Severity

Tujuan:

- melihat distribusi tingkat alert;
- memisahkan low, medium, high;
- memprioritaskan triage.

Field:

```
severity
```

Visualisasi 3 — Top Source IP

Tujuan:

- mencari sumber yang paling sering memicu alert;
- membantu deteksi scanning;
- membantu laporan insiden.

Field:

```
source_ip
```

Visualisasi 4 — Top Rule ID

Tujuan:

- melihat rule paling bising;
- menilai tuning rule;
- mengurangi false positive.

Field:

```
rule_id
```

Visualisasi 5 — Alert per MITRE Tactic

Tujuan:

- melihat pola taktik adversary;
- menghubungkan alert dengan kerangka analisis;
- membantu laporan strategis.

Field:

```
mitre_tactic
```

Prinsip Dashboard

- jangan terlalu banyak panel;
- gunakan filter waktu;
- fokus pada pertanyaan analis;
- semua panel harus bisa dijelaskan;
- jangan membuat visualisasi hanya karena terlihat menarik.

6.17 Hands-on 6 — Membuat Alert Sederhana

Skenario

Buat alert jika ada lebih dari 5 event `authentication_failed` dalam 5 menit.

Konsep

- monitor berjalan setiap 1 menit;
- query mencari event login gagal;
- trigger aktif jika jumlah melewati threshold;
- action menulis atau mengirim notifikasi lokal.

Query Konseptual

```
{
  "query": {
    "bool": {
      "filter": [
        {
          "term": {
            "event_type": "authentication_failed"
          }
        },
        {
          "range": {
            "timestamp": {
              "gte": "now-5m"
            }
          }
        }
      ]
    }
  }
}
```

Penjelasan:

- `bool` menggabungkan beberapa kondisi;
- `filter` membuat pencarian lebih spesifik;
- `term` memilih event login gagal;
- `range` membatasi waktu;
- `now-5m` berarti lima menit terakhir.

Catatan

Alert berbasis query harus diuji. Jangan langsung percaya.

Uji:

- apakah alert muncul saat simulasi?
- apakah tidak muncul saat tidak ada simulasi?
- apakah threshold terlalu rendah?
- apakah query terlalu luas?
- apakah field sudah normal?

6.18 Hands-on 7 — Ekspor Hasil Query untuk Laporan Insiden

Tujuan

Mengubah hasil pencarian menjadi bukti laporan.

Field yang diekspor:

```
timestamp
host
source_ip
destination_ip
rule_id
severity
event_type
mitre_tactic
mitre_technique
```

Contoh output laporan ringkas:

```
{
  "case_id": "SOC-LAB-006-001",
  "summary": "Terdapat 12 event login gagal dalam 5 menit
dari satu source IP ke endpoint Linux.",
  "evidence": {
    "source_ip": "10.10.10.50",
    "destination_ip": "10.10.10.21",
    "event_type": "authentication_failed",
    "count": 12,
    "time_window": "5 minutes",
    "mitre_tactic": "Credential Access"
  },
  "status": "needs_triage"
}
```

Gunakan hasil ini untuk:

- laporan praktikum;
- case management;
- input AI lokal;
- dokumentasi proyek akhir;
- bahan presentasi.

6.19 Kesalahan Umum dalam Log Management

Kesalahan yang sering terjadi:

- semua log dimasukkan tanpa strategi;
- field tidak normal;
- timestamp salah;
- timezone tidak konsisten;
- severity dianggap mutlak tanpa konteks;
- dashboard terlalu ramai;
- query tidak diuji;
- alert threshold terlalu rendah;
- storage cepat penuh;
- log sensitif tidak disanitasi;
- hasil query tidak ditulis sebagai laporan.

Log management yang buruk membuat SOC lambat, mahal, dan membingungkan.

6.20 Checklist Bab 6

Sebelum lanjut Bab 7, pastikan:

- OpenSearch aktif.
- OpenSearch Dashboards aktif.
- Sample log JSON berhasil masuk.
- Index pattern dibuat.
- Field timestamp benar.
- Query severity tinggi berhasil.
- Query login gagal berhasil.
- Dashboard alert per host dibuat.
- Dashboard alert per severity dibuat.
- Top source IP dibuat.
- Top rule ID dibuat.
- Alert sederhana dibuat.
- Hasil query dapat diekspor untuk laporan.
- Data sensitif tidak dimasukkan ke AI eksternal.

6.21 Rangkuman Bab

Bab ini menjelaskan OpenSearch sebagai fondasi *log management*, query, dashboard, dan alert dalam SOC open source. OpenSearch membantu analis mencari bukti, menyaring noise, melihat tren, membuat dashboard, dan menyusun laporan dari data terstruktur.

Pembaca belajar bahwa log harus diindeks, field harus dinormalisasi, mapping harus dipahami, query harus diuji, dan retensi log harus direncanakan. Dashboard bukan hiasan, tetapi alat bantu keputusan. Alert berbasis query bukan pengganti analis, tetapi mekanisme awal untuk mempercepat triage.

Dengan OpenSearch, SOC tidak hanya menerima alert. SOC bisa **bertanya kepada data**.

6.22 Output Bab

Setelah menyelesaikan Bab 6, pembaca diharapkan mampu:

- menjelaskan mengapa log perlu diindeks;
- memahami struktur data log SOC;
- membedakan *index*, *mapping*, *field*, dan *query*;
- memahami fungsi OpenSearch dan OpenSearch Dashboards;
- mengirim sample log JSON ke OpenSearch;
- membuat *index pattern*;
- membuat query dasar untuk SOC;
- membuat dashboard alert per host, severity, source IP, rule ID, dan MITRE tactic;
- membuat alert sederhana berbasis query;
- memahami retensi log dan kapasitas storage;
- melakukan normalisasi field lintas sumber;
- mengekspor hasil query untuk laporan insiden;
- memakai AI open source sebagai asisten penjelas data, bukan pengganti bukti.

Pesan akhir Bab 6:
SOC yang matang bukan hanya menyimpan log; SOC yang matang mampu menemukan bukti yang tepat pada waktu yang tepat.

Bab 7 — Incident Response dan Case Management dengan DFIR-IRIS atau TheHive/Cortex

> **Alert belum tentu incident; incident harus dibuktikan, di triage, didokumentasikan, ditangani, dan ditutup dengan pelajaran yang jelas.**

7.1 Arah Bab

Bab ini mengajarkan disiplin penting dalam SOC: **mengubah alert menjadi case investigasi yang rapi**. Pada bab sebelumnya, pembaca sudah belajar Wazuh, deteksi, FIM, vulnerability, Suricata, Zeek, OpenSearch, query, dan dashboard. Semua itu menghasilkan data dan alert. Namun alert saja belum cukup.

Outline buku menempatkan Bab 7 sebagai bab *incident response* dan *case management*: membedakan event, alert, incident, case; memahami tahapan respons insiden; melakukan triage; mengelola observables, task, timeline, evidence, analyst notes, dan final report; lalu menutup case dengan rekomendasi.

Untuk buku yang tegas memakai **open source tools**, **DFIR-IRIS** dapat dijadikan pilihan utama karena secara resmi menyebut dirinya sebagai platform kolaboratif *free and open-source* untuk *incident response*. (DFIR-IRIS) TheHive/Cortex tetap dibahas sebagai referensi penting karena banyak dipakai dalam integrasi SOC, Wazuh, MISP, analyzer, dan workflow incident response.

7.2 Mengapa Alert Tidak Sama dengan Incident?

SOC menghasilkan banyak alert. Namun tidak semua alert harus menjadi incident.

> **Alert adalah sinyal. Incident adalah masalah keamanan yang sudah divalidasi dan perlu respons.**

Contoh:

- login gagal satu kali = event;
- login gagal berulang = alert;
- login berhasil setelah banyak percobaan gagal dari sumber mencurigakan = kandidat incident;
- terbukti ada akses tidak sah dan perubahan file penting = incident.

Rujukan SOC analyst membedakan security logs, security event, incident, dan breach. Security event dapat dieskalasi menjadi incident jika perlu, sedangkan incident memicu proses *incident response* dan biasanya membutuhkan penanganan lebih formal.

Prinsip penting:

- **event** belum tentu penting;
- **alert** perlu triage;
- **incident** perlu response;
- **case** adalah wadah dokumentasi investigasi;
- **breach** adalah istilah sensitif dan tidak boleh dipakai sembarangan.

Untuk mahasiswa, gunakan istilah dengan hati-hati. Jangan menulis “breach” jika belum ada bukti kuat dan otoritas yang menyatakan.

7.3 Perbedaan Event, Alert, Incident, dan Case

Event

Event adalah kejadian yang tercatat.

Contoh:

- user login;
- file berubah;
- koneksi jaringan;
- DNS query;
- service restart;
- vulnerability ditemukan.

Event adalah bahan mentah untuk analisis.

Alert

Alert adalah event yang dianggap penting oleh rule, query, atau sistem deteksi.

Contoh:

- login gagal berulang;
- file konfigurasi penting berubah;
- port scan terdeteksi;
- hash mencurigakan;
- vulnerability severity tinggi.

Alert harus di triage.

Incident

Incident adalah kejadian keamanan yang sudah divalidasi, memiliki dampak atau potensi dampak, dan membutuhkan respons.

Contoh:

- akun disalahgunakan;
- host menunjukkan indikasi kompromi;
- file penting diubah tanpa izin;
- endpoint berkomunikasi dengan infrastruktur mencurigakan;
- ada eksfiltrasi data.

Incident membutuhkan tindakan.

Case

Case adalah wadah kerja investigasi.

Case berisi:

- ringkasan;
- observables;
- evidence;
- timeline;
- tasks;
- analyst notes;
- status;
- keputusan;
- rekomendasi;
- laporan akhir.

Case adalah memori kerja SOC. Tanpa case, investigasi mudah hilang, bias, dan sulit dipertanggungjawabkan.

7.4 Tahapan Incident Response

Kerangka NIST untuk *incident response* membagi proses menjadi empat fase besar: **Preparation**, **Detection and Analysis**, **Containment, Eradication and Recovery**, dan **Post-Incident Activity**. (NIST Pubs)

Dalam buku ini, fase tersebut dijelaskan dalam bentuk yang lebih mudah untuk mahasiswa:

1. Preparation

Tahap persiapan.

Isi:

- membuat lab;
- menyiapkan tools;
- menyiapkan kontak;
- membuat template laporan;
- membuat rule deteksi;
- membuat SOP;
- membuat backup;
- membuat daftar aset;
- menyiapkan akses yang aman.

Preparation adalah fase paling sering diremehkan. Padahal respons yang baik dimulai sebelum insiden terjadi.

2. Identification

Tahap identifikasi.

Pertanyaan utama:

- alert apa yang muncul?
- sumbernya dari mana?
- aset apa terdampak?
- apakah alert valid?
- apakah ini incident?
- bukti apa yang mendukung?

Di tahap ini, analis membaca Wazuh, OpenSearch, Suricata, Zeek, FIM, vulnerability, dan threat intelligence.

3. Containment

Tahap pembatasan dampak.

Tujuannya bukan langsung “membersihkan”, tetapi **mencegah kerusakan meluas**.

Contoh containment di lab:

- memutus koneksi host uji;
- memblokir IP sumber di jaringan lab;
- menonaktifkan akun uji;
- menghentikan service uji;
- membatasi akses sementara.

Containment harus hati-hati. Jangan sampai tindakan respons menghapus bukti.

4. Eradication

Tahap penghapusan akar masalah.

Contoh:

- menghapus file berbahaya di lab;
- menutup konfigurasi lemah;
- menghapus akun tidak sah;
- memperbaiki rule;
- menghapus persistence;
- memperbaiki service rentan.

Eradication dilakukan setelah bukti cukup.

5. Recovery

Tahap pemulihan.

Contoh:

- mengembalikan service;
- memulihkan file dari backup;
- memastikan host bersih;
- memantau ulang alert;
- validasi bahwa serangan tidak muncul lagi.

Recovery bukan sekadar “nyalakan lagi”. Recovery harus memastikan sistem kembali aman.

6. Lessons Learned

Tahap pembelajaran.

Isi:

- apa yang terjadi?
- mengapa terjadi?
- apa yang berhasil?
- apa yang gagal?
- rule apa perlu diperbaiki?
- dashboard apa perlu ditambah?
- SOP apa perlu direvisi?
- apa rekomendasi untuk mencegah kejadian ulang?

NIST SP 800-61r3 juga menekankan bahwa organisasi perlu belajar dari insiden untuk mengurangi dampak dan memperbaiki praktik manajemen risiko serta respons siber. (NIST Pubs)

Incident response yang tidak menghasilkan lesson learned hanya memadamkan api, bukan memperbaiki sistem.

7.5 Triage: Memilih Alert yang Layak Menjadi Case

Triage adalah proses menentukan prioritas dan arah penanganan alert. Triage menjawab: **apakah alert ini perlu diabaikan, di monitor, eskalasi, atau dijadikan case?**

Empat faktor utama:

1. Severity

Severity menunjukkan tingkat teknis alert.

Contoh:

- low;
- medium;
- high;
- critical.

Namun severity tidak boleh dibaca sendirian. Alert severity medium pada aset penting bisa lebih serius daripada alert high pada host lab biasa.

2. Confidence

Confidence menunjukkan seberapa yakin alert tersebut benar.

Pertanyaan:

- apakah rule akurat?
- apakah ada bukti pendukung?
- apakah alert sering false positive?
- apakah indikator cocok dengan threat intelligence?
- apakah ada korelasi endpoint dan jaringan?

3. Asset Criticality

Asset criticality menunjukkan nilai aset.

Contoh aset kritikal:

- server autentikasi;
- server database;
- repository kode;
- sistem akademik;
- server layanan publik;
- endpoint admin.

Rujukan SOC modern menekankan pentingnya konteks aset, termasuk criticality, departemen, lokasi, dan informasi identity untuk memperkaya alert agar korelasi lebih bermakna.

4. Impact

Impact menunjukkan dampak potensial.

Pertanyaan:

- apakah data sensitif terlibat?
- apakah layanan terganggu?
- apakah ada akses tidak sah?
- apakah ada indikasi eskalasi?
- apakah ada pergerakan lateral?
- apakah ada potensi eksfiltrasi?

Formula sederhana:

Prioritas Case = Severity + Confidence + Asset Criticality + Impact

Triage yang baik menghemat waktu analis dan mencegah SOC tenggelam dalam noise.

7.6 Case Management

Case management adalah proses mengelola investigasi dari awal sampai selesai.

Case yang baik harus menjawab:

- apa yang terjadi?
- kapan terjadi?
- siapa atau apa yang terdampak?
- bukti apa yang tersedia?
- analisis apa yang sudah dilakukan?
- task apa yang belum selesai?
- siapa yang bertanggung jawab?
- apa keputusan akhirnya?
- apa rekomendasinya?

Komponen Case

1. Observables

Observables adalah objek teknis yang diamati.

Contoh:

- IP address;
- domain;
- hash;
- URL;
- username;
- hostname;
- file path;
- process name;
- rule ID;
- email address jika relevan dan sah.

2. Tasks

Tasks adalah pekerjaan yang harus dilakukan analis.

Contoh:

- cek alert Wazuh;
- cek Zeek `conn.log`;
- cek DNS log;
- cek FIM;
- cek vulnerability;
- cek MISP;
- validasi hash;
- buat timeline;
- buat rekomendasi.

3. Timeline

Timeline adalah urutan kejadian.

Contoh:

10:00 - Login gagal pertama
10:02 - Login gagal berulang
10:04 - Alert Wazuh muncul
10:05 - Koneksi jaringan terdeteksi
10:07 - File konfigurasi berubah
10:10 - Case dibuat

Timeline membantu analis melihat cerita, bukan sekadar potongan alert.

4. Evidence

Evidence adalah bukti.

Contoh:

- alert JSON;
- potongan log;
- screenshot dashboard;
- hasil query OpenSearch;
- hash file;
- PCAP;
- output Zeek;
- catatan FIM;
- laporan vulnerability.

Evidence harus disimpan rapi. Jangan mengubah bukti asli.

5. Analyst Notes

Catatan analis berisi interpretasi dan keputusan sementara.

Contoh:

- alert valid;
- belum cukup bukti incident;
- perlu cek DNS;
- perlu cek host lain;
- kemungkinan false positive;
- butuh konfirmasi pemilik aset.

6. Final Report

Final report adalah ringkasan akhir.

Isi:

- ringkasan insiden;
- timeline;
- aset terdampak;
- bukti;
- analisis;
- tindakan;
- dampak;
- status akhir;
- rekomendasi;
- lesson learned.

7.7 DFIR-IRIS sebagai Pilihan Utama Open Source

DFIR-IRIS adalah platform kolaboratif untuk *incident response*. Situs resminya menyebut DFIR-IRIS sebagai platform kolaboratif *free and open-source* untuk menyederhanakan investigasi, berbagi detail teknis, dan kolaborasi real-time. (DFIR-IRIS)

Dokumentasi DFIR-IRIS menjelaskan bahwa IRIS adalah platform digital kolaboratif yang dirancang agar analis incident response dapat berbagi investigasi kompleks pada level teknis. Ia dapat dipasang pada server khusus atau sebagai aplikasi portabel untuk investigasi bergerak. (docs.dfir-iris.org)

Kelebihan DFIR-IRIS untuk buku ini:

- **open source**;
- cocok untuk case management;
- cocok untuk dokumentasi investigasi;
- mendukung kolaborasi;
- cocok untuk mahasiswa;
- fokus pada DFIR dan incident response;
- bisa dipakai untuk laporan praktikum.

Fungsi yang relevan:

- membuat case;
- menambahkan observables;
- membuat timeline;
- mengelola evidence;
- menulis notes;
- mengatur task;
- membuat laporan.

Untuk arah buku “tools gratis dan open source”, DFIR-IRIS sangat cocok dijadikan jalur utama.

7.8 TheHive/Cortex sebagai Referensi Integrasi SOC

TheHive banyak dipakai sebagai *case management platform* untuk SOC, CERT, CSIRT, dan tim incident response. Dokumentasi TheHive 5 dan Cortex tersedia dari StrangeBee untuk instalasi, konfigurasi, maintenance, penggunaan TheHive, dan Cortex. (StrangeBee Docs)

Namun status versi dan lisensi perlu dicek sesuai kebutuhan implementasi. Repository TheHive saat ini menyatakan TheHive didistribusikan sebagai versi komersial. (GitHub) Karena buku ini fokus pada open source, TheHive/Cortex dibahas sebagai referensi arsitektur dan integrasi, sementara praktik utama dapat diarahkan ke DFIR-IRIS.

Rujukan Wazuh terlampir menjelaskan bahwa TheHive dapat menerima alert dari Wazuh, menjalankan analisis observable melalui Cortex, membandingkan observable dengan MISP, membuat task untuk analis, dan membantu menghasilkan laporan.

Cortex sendiri dijelaskan sebagai engine open-source pendamping TheHive yang dapat menganalisis banyak observables sekaligus dan memicu respons aktif dengan beberapa klik. (StrangeBee)

Kesimpulan praktis:

- **DFIR-IRIS**: direkomendasikan sebagai jalur utama open source untuk mahasiswa.
- **TheHive/Cortex**: dibahas sebagai model integrasi SOC yang banyak digunakan, terutama untuk Wazuh–MISP–analyzer workflow.

7.9 Alur Kerja Case dari Alert Wazuh

Alur sederhana:

```
Alert Wazuh
    ↓
Triage
    ↓
Validasi log / OpenSearch / Zeek
    ↓
Buat Case
    ↓
Tambah Observables
    ↓
Tambah Evidence
    ↓
Buat Timeline
    ↓
Buat Tasks
    ↓
Analyst Notes
    ↓
Final Report
    ↓
Close Case
```

Contoh skenario:

Alert:
Login gagal berulang pada endpoint Linux.

Triage:
Severity medium-high, confidence cukup, aset medium, impact belum terbukti.

Validasi:
Cek Wazuh alert, OpenSearch query, Zeek conn.log.

Case:
Dibuat sebagai "SOC-LAB-007-001 - Login Gagal Berulang".

Status:
Under investigation.

Case bukan tempat menyalin alert. Case adalah tempat membangun cerita berbasis bukti.

7.10 Peran AI Open Source dalam Case Management

AI open source dapat membantu incident response, tetapi tetap harus dikontrol.

AI dapat membantu:

- merangkum alert;
- menyusun timeline awal;
- membuat draft laporan;
- merapikan analyst notes;
- membantu klasifikasi severity;
- menyarankan task investigasi;
- membantu mapping MITRE;
- membuat ringkasan untuk dosen atau pimpinan lab.

AI tidak boleh:

- mengambil keputusan incident sendirian;
- menghapus atau mengubah evidence;
- menerima data sensitif tanpa sanitasi;
- menjalankan containment otomatis tanpa approval;
- menyimpulkan breach tanpa dasar kuat.

Rujukan open-source SOC menjelaskan bahwa analitik dan AI dapat membantu konteks insiden, scoring severity, laporan otomatis, dan prioritas respons, tetapi otomasi tetap membutuhkan governance, audit log, dan kontrol etis agar tidak menimbulkan pelanggaran atau kesalahan operasional.

Prinsipnya:

> **AI membantu menulis dan merangkum; analis tetap membuktikan dan memutuskan.**

7.11 Hands-on 1 — Instalasi DFIR-IRIS dengan Docker Compose

Tujuan

Menyiapkan platform case management open source untuk lab SOC.

Prasyarat

- Docker aktif.
- Docker Compose aktif.
- Folder `soc-lab/` tersedia.
- Jaringan lab terisolasi.
- Snapshot sudah dibuat.
- Password kuat sudah disiapkan.

Struktur Folder

```
mkdir -p ~/soc-lab/docker/dfir-iris
cd ~/soc-lab/docker/dfir-iris
```

Penjelasan:

- `mkdir -p` membuat folder bertingkat.
- `~/soc-lab/docker/dfir-iris` adalah lokasi konfigurasi IRIS.
- `cd` masuk ke folder kerja.

Instalasi

Gunakan panduan resmi DFIR-IRIS dari dokumentasi atau repository GitHub. Repository DFIR-IRIS menjelaskan bahwa IRIS adalah platform kolaboratif untuk incident responders dan menggunakan lisensi LGPL-3.0. (GitHub)

Prinsip instalasi:

- gunakan file compose resmi;
- jangan membuka port ke internet;
- ganti semua password default;
- batasi akses hanya dari jaringan lab;
- catat versi;
- buat snapshot setelah berhasil login.

Validasi

Cek:

- halaman login terbuka dari jaringan lab;
- user admin bisa masuk;
- case baru bisa dibuat;
- observable bisa ditambahkan;
- timeline bisa dibuat;
- report bisa diekspor atau ditulis.

7.12 Hands-on 2 — Membuat Case dari Alert Wazuh

Tujuan

Mengubah alert Wazuh menjadi case investigasi.

Sumber Alert

Gunakan alert dari Bab 4, misalnya:

Login gagal berulang pada endpoint Linux

Data Minimum Case

Isi:

Judul Case:
SOC-LAB-007-001 - Login Gagal Berulang pada Endpoint Linux

Severity:
Medium

Status:
Under investigation

Sumber Alert:
Wazuh

Waktu Deteksi:
2026-01-01 10:04

Endpoint:
linux-endpoint-01

Ringkasan:
Terdapat login gagal berulang dari satu source IP ke endpoint Linux dalam jaringan lab.

Prinsip Penulisan

- tulis singkat;
- tulis berbasis fakta;
- jangan menuduh tanpa bukti;
- bedakan simulasi lab dan incident nyata;
- catat asumsi;
- catat data yang belum lengkap.

7.13 Hands-on 3 — Menambahkan Observables

Tambahkan observables berikut:

IP address:
10.10.10.50

Hostname:
linux-endpoint-01

Username:
admin-test

Rule ID:
100200

Event type:
authentication_failed

Jika ada hash file dari skenario lain, tambahkan:

Hash:
e3b0c44298fc1c149afbf4c8996fb924...

Jika ada domain dari log DNS, tambahkan:

Domain:
example-lab.local

Pertanyaan Analis

- observable mana yang paling penting?
- observable mana yang perlu enrichment?
- apakah IP berasal dari lab?
- apakah username valid?
- apakah hash perlu dicek?
- apakah domain muncul di threat intelligence?

Rujukan Wazuh menjelaskan bahwa TheHive dapat membantu menganalisis observables dari alert Wazuh sebelum analis memutuskan apakah alert diabaikan atau dikonversi menjadi case.

7.14 Hands-on 4 — Membuat Timeline Insiden

Timeline harus sederhana dan kronologis.

Contoh:

10:00 - Percobaan login gagal pertama tercatat.
10:01 - Login gagal berulang dari source IP yang sama.
10:04 - Wazuh menghasilkan alert rule 100200.
10:05 - Analis membuka alert di dashboard.
10:07 - Query OpenSearch menunjukkan 12 event dalam 5 menit.
10:10 - Case dibuat di DFIR-IRIS.
10:15 - Tidak ditemukan login berhasil setelah percobaan gagal.
10:20 - Case diklasifikasikan sebagai simulasi valid di lab.

Timeline membantu melihat:

- urutan kejadian;
- jeda deteksi;
- waktu respons;
- bukti pendukung;
- langkah analis.

Rujukan Wazuh terkait TheHive menyebut case timeline menggambarkan siklus hidup case, termasuk alert awal, task, IOC, dan aktivitas yang berlangsung selama investigasi.

7.15 Hands-on 5 — Membuat Task untuk Analis

Task membuat investigasi terstruktur.

Contoh task:

> Task 1:
> Validasi alert Wazuh.
> Status: Done.
>
> Task 2:
> Cek apakah ada login berhasil setelah percobaan gagal.
> Status: Done.
>
> Task 3:
> Cek source IP pada Zeek conn.log.
> Status: In progress.
>
> Task 4:
> Cek apakah endpoint mengalami perubahan file penting.
> Status: Pending.
>
> Task 5:
> Buat kesimpulan dan rekomendasi.
> Status: Pending.

Task membantu membagi pekerjaan jika investigasi dilakukan kelompok.

Pembagian Peran Mahasiswa

- analis 1: Wazuh alert;
- analis 2: OpenSearch query;
- analis 3: Zeek/Suricata;
- analis 4: laporan;
- analis 5: rekomendasi dan presentasi.

Case management melatih kerja tim, bukan hanya kemampuan individu.

7.16 Hands-on 6 — Menambahkan Evidence

Evidence harus jelas dan dapat ditelusuri.

Contoh evidence:

Evidence 1:
Potongan alert JSON dari Wazuh.

Evidence 2:
Hasil query OpenSearch untuk authentication_failed.

Evidence 3:
Zeek conn.log terkait IP sumber.

Evidence 4:
Screenshot dashboard alert per host.

Evidence 5:
Catatan FIM bahwa tidak ada perubahan file penting.

Format Evidence Ringkas

```
{
  "evidence_id": "EV-001",
  "source": "Wazuh",
  "type": "alert_json",
  "timestamp": "2026-01-01T10:04:00Z",
  "summary": "Login gagal berulang dari 10.10.10.50 ke
linux-endpoint-01",
  "stored_at":
"reports/SOC-LAB-007-001/evidence/wazuh-alert.json"
}
```

Prinsip evidence:

- jangan ubah data asli;
- simpan salinan;
- beri nama jelas;
- catat sumber;
- catat waktu;
- catat analis yang mengambil;
- jangan menyimpan data sensitif tanpa perlu.

7.17 Hands-on 7 — Membuat Laporan Insiden Satu Halaman

Gunakan format singkat.

> Laporan Incident Response — SOC-LAB-007-001
>
> Judul:
> Login gagal berulang pada endpoint Linux.
>
> Ringkasan:
> Wazuh mendeteksi login gagal berulang dari satu source IP ke endpoint Linux dalam jaringan lab. Setelah validasi dengan OpenSearch dan log jaringan, tidak ditemukan login berhasil setelah percobaan gagal. Aktivitas diklasifikasikan sebagai simulasi valid di lab.
>
> Severity:
> Medium.
>
> Aset Terdampak:
> linux-endpoint-01.
>
> Observable:
> - Source IP: 10.10.10.50
> - Hostname: linux-endpoint-01
> - Username: admin-test
> - Rule ID: 100200
>
> Timeline:
> - 10:00 login gagal pertama.
> - 10:04 alert Wazuh muncul.
> - 10:07 query OpenSearch dilakukan.
> - 10:10 case dibuat.
> - 10:20 case siap ditutup.
>
> Evidence:
> - Alert JSON Wazuh.
> - Query OpenSearch.
> - Log koneksi jaringan.
> - Catatan analis.
>
> Analisis:
> Alert valid sebagai simulasi lab. Belum ada bukti akses berhasil atau perubahan file penting.
>
> Tindakan:
> - Monitoring.
> - Dokumentasi case.
> - Rekomendasi penguatan password dan pembatasan akses.
>
> Status:
> Closed — Lab simulation validated.
>
> Lesson Learned:
> Alert perlu dikorelasikan dengan bukti tambahan sebelum dinyatakan incident.

Rujukan SOC analyst method menekankan pentingnya menulis alasan alert, bukti pendukung, analisis, tindakan, kesimpulan, dan langkah berikutnya.

7.18 Hands-on 8 — Menutup Case dengan Status dan Rekomendasi

Case tidak boleh dibiarkan menggantung.

Status sederhana:

- **New** — case baru dibuat.
- **Under investigation** — sedang dianalisis.
- **Waiting evidence** — menunggu data tambahan.
- **Resolved** — sudah ditangani.
- **Closed false positive** — tidak terbukti ancaman.
- **Closed lab simulation** — simulasi valid.
- **Escalated** — butuh tim lanjutan.

Rekomendasi akhir:

- perbaiki rule;
- tambahkan dashboard;
- tambahkan alert threshold;
- perbaiki konfigurasi endpoint;
- batasi akses layanan;
- tambahkan monitoring FIM;
- tambahkan threat intelligence;
- buat playbook SOAR;
- latih analis membaca timeline.

Rujukan SOC metrics menekankan pentingnya MTTD, MTTR, false positive rate, incident closure rate, impact assessment, dan post-incident review untuk memperbaiki efektivitas SOC.

7.19 Template Case Incident

Gunakan template berikut sebagai standar buku.

Case ID:
Judul:
Tanggal/Waktu Deteksi:
Sumber Alert:
Analis:
Status:
Severity:
Confidence:
Asset Criticality:
Impact:

Ringkasan:
...

Observable:
- IP address:
- Domain:
- Hash:
- Username:
- Hostname:
- Rule ID:

Evidence:
- Alert JSON:
- Query OpenSearch:
- Log Zeek/Suricata:
- FIM:
- Screenshot:
- Catatan lain:

Timeline:
1.
2.
3.

Tasks:
1.
2.
3.

Analyst Notes:
...

Analisis:

...

Tindakan:
...

Kesimpulan:
...

Rekomendasi:
...

Lesson Learned:
...

Template ini dapat dipakai untuk laporan praktikum, proyek akhir, atau simulasi kelas.

7.20 Kesalahan Umum dalam Incident Response

Kesalahan yang sering terjadi:

- menganggap semua alert sebagai incident;
- tidak melakukan triage;
- tidak mencatat evidence;
- tidak membuat timeline;
- tidak membedakan simulasi lab dan insiden nyata;
- langsung melakukan containment tanpa bukti;
- menghapus bukti saat “membersihkan” sistem;
- memakai istilah breach sembarangan;
- tidak menulis lesson learned;
- menutup case tanpa rekomendasi;
- terlalu percaya ringkasan AI tanpa memeriksa log.

Incident response yang buruk bisa memperburuk insiden.

7.21 Checklist Bab 7

Sebelum lanjut ke Bab 8, pastikan:

- Paham beda event, alert, incident, dan case.
- Paham lifecycle incident response.
- Paham triage: severity, confidence, asset criticality, impact.
- DFIR-IRIS atau tool case management terbuka sudah tersedia.
- Case dari alert Wazuh berhasil dibuat.
- Observable berhasil ditambahkan.
- Timeline dibuat.
- Task analis dibuat.
- Evidence dicatat.
- Laporan satu halaman dibuat.
- Case ditutup dengan status dan rekomendasi.
- AI hanya dipakai untuk membantu ringkasan, bukan memutuskan.

7.22 Rangkuman Bab

Bab ini menjelaskan bahwa SOC tidak berhenti pada alert. Alert harus di triage, divalidasi, diberi konteks, lalu dikelola sebagai case jika layak. Incident response membutuhkan proses yang jelas: preparation, identification, containment, eradication, recovery, dan lessons learned.

DFIR-IRIS cocok sebagai platform utama untuk buku ini karena mendukung pendekatan open source dan kolaboratif dalam incident response. TheHive/Cortex tetap penting sebagai referensi integrasi SOC, terutama untuk alur Wazuh, MISP, observables, analyzer, dan workflow investigasi.

Inti Bab 7 adalah **akuntabilitas**. Setiap keputusan harus punya bukti, setiap tindakan harus tercatat, dan setiap case harus menghasilkan pelajaran.

7.23 Output Bab

Setelah menyelesaikan Bab 7, pembaca diharapkan mampu:

- membedakan event, alert, incident, dan case;
- menjelaskan tahapan incident response;
- melakukan triage berbasis severity, confidence, asset criticality, dan impact;
- memahami fungsi case management;
- membuat case dari alert Wazuh;
- menambahkan observable seperti IP, hash, domain, username, dan hostname;
- membuat timeline insiden;
- membuat task untuk analis;
- menyimpan evidence;
- menulis analyst notes;
- membuat laporan insiden satu halaman;
- menutup case dengan status dan rekomendasi;
- memakai AI open source sebagai asisten dokumentasi, bukan pengganti keputusan.

Pesan akhir Bab 7:
SOC yang profesional bukan hanya cepat mendeteksi, tetapi mampu membuktikan, mencatat, merespons, dan belajar dari setiap insiden.

Bab 8 — *Threat Intelligence* dengan MISP dan Mapping MITRE ATT&CK

> **Alert memberi tahu ada sinyal; *threat intelligence* memberi konteks agar SOC tahu sinyal itu penting, relevan, atau hanya noise.**

8.1 Arah Bab

Bab ini membahas bagaimana SOC tidak bekerja sendirian. Wazuh, Security Onion, Suricata, Zeek, OpenSearch, dan DFIR-IRIS memberi visibilitas internal. Namun SOC juga membutuhkan informasi dari luar: **IOC, TTP, malware family, campaign, threat actor, dan pola serangan global**.

Dalam outline buku, Bab 8 diarahkan untuk membahas MISP, IOC, TTP, MITRE ATT&CK, Cyber Kill Chain, feed, JSON, etika berbagi IOC, serta integrasi alert Wazuh dengan konteks threat intelligence.

MISP dipilih karena merupakan platform *open source threat intelligence sharing*. MISP menyediakan model data seperti **event**, **object**, dan **attribute**, serta API untuk query, membuat, dan memodifikasi data intelijen ancaman. (MISP Threat Intelligence Platform) MISP core format juga menggunakan JSON sederhana untuk pertukaran event dan attribute, sehingga cocok untuk ekosistem SOC berbasis API dan automation. (MISP Threat Intelligence Platform)

8.2 Apa Itu Threat Intelligence?

Threat intelligence adalah pengetahuan terstruktur tentang ancaman siber. Bukan sekadar daftar IP berbahaya. Threat intelligence menjelaskan **siapa menyerang, bagaimana caranya, indikator apa yang ditinggalkan, target apa yang disasar, dan tindakan pertahanan apa yang relevan**.

Rujukan Wazuh menjelaskan bahwa threat intelligence membantu SOC membuat keputusan keamanan yang lebih cepat, berbasis data, dan lebih proaktif. Threat intelligence juga membantu memahami threat actor, motif, kapabilitas, IOC, serta TTP.

Threat intelligence membantu SOC menjawab:

- **Apakah IP ini pernah dipakai dalam serangan?**
- **Apakah domain ini terkait kampanye tertentu?**
- **Apakah hash file ini pernah dilaporkan sebagai malware?**
- **Apakah teknik serangan ini cocok dengan MITRE ATT&CK?**
- **Apakah alert ini relevan untuk organisasi kita?**
- **Apakah incident perlu dinaikkan prioritasnya?**

Tanpa threat intelligence, SOC hanya melihat sinyal internal. Dengan threat intelligence, SOC bisa melihat **konteks eksternal**.

8.3 Data, Informasi, dan Intelligence

Banyak pemula mencampuradukkan data, informasi, dan intelligence. Padahal ketiganya berbeda.

Data

Data adalah fakta mentah.

Contoh:

```
10.10.10.50
example-lab.local
b1946ac92492d2347c6235b4d2611184
```

Data belum menjelaskan makna.

Informasi

Informasi adalah data yang sudah diberi konteks.

Contoh:

```
IP 10.10.10.50 melakukan 20 percobaan login gagal ke
endpoint Linux dalam 5 menit.
```

Informasi mulai menjawab apa yang terjadi.

Intelligence

Intelligence adalah informasi yang sudah dianalisis untuk mendukung keputusan.

Contoh:

```
IP 10.10.10.50 dalam lab memicu pola brute force. Tidak
ditemukan login berhasil. Case diklasifikasikan sebagai
simulasi valid. Rule tetap dipertahankan, tetapi
threshold perlu diuji ulang agar tidak menimbulkan
noise.
```

Intelligence membantu menentukan tindakan.

Data menjawab “apa”. Informasi menjawab “apa artinya”. Intelligence menjawab “apa yang harus dilakukan”.

8.4 IOC vs TTP

Dalam threat intelligence, dua istilah yang wajib dipahami adalah **IOC** dan **TTP**.

IOC — Indicator of Compromise

IOC adalah indikator teknis yang menunjukkan kemungkinan kompromi atau aktivitas berbahaya.

Contoh IOC:

- IP address;
- domain;
- URL;
- file hash;
- file path;
- email address;
- nama file;
- user agent;
- registry key jika relevan pada lingkungan tertentu;
- pola komunikasi jaringan.

Rujukan Wazuh mendefinisikan IOC sebagai data teknis yang dikumpulkan dari investigasi, threat hunting, atau malware analysis, misalnya IP address, domain, dan file hash.

Kelebihan IOC:

- mudah dipakai untuk matching;
- mudah dimasukkan ke SIEM;
- mudah dibagikan dalam JSON;
- cocok untuk deteksi cepat.

Kelemahan IOC:

- cepat basi;
- mudah diganti penyerang;
- bisa menghasilkan false positive;
- tanpa konteks bisa menyesatkan.

TTP — Tactics, Techniques, and Procedures

TTP menjelaskan cara kerja adversary.

- **Tactic** = tujuan umum penyerang.
- **Technique** = cara teknis mencapai tujuan.
- **Procedure** = implementasi spesifik di lapangan.

Rujukan Wazuh menjelaskan bahwa MITRE ATT&CK digunakan untuk mengkategorikan dan memahami TTP adversary dalam operasi keamanan. Tactics adalah “apa” yang ingin dicapai, techniques adalah “bagaimana” dilakukan, dan procedures adalah langkah lebih spesifik.

Contoh:

- Tactic: *Credential Access*.
- Technique: *Brute Force*.
- Procedure: mencoba banyak password ke layanan login dalam waktu singkat.

IOC mudah berubah. TTP lebih tahan lama. SOC yang matang membaca keduanya.

8.5 Jenis Threat Intelligence

Threat intelligence dapat dibagi menjadi beberapa jenis. Dalam buku ini, digunakan empat kategori agar mudah dipahami mahasiswa.

1. Strategic Intelligence

Strategic intelligence ditujukan untuk pengambil keputusan. Isinya bukan detail teknis panjang, tetapi gambaran besar.

Contoh:

- tren serangan terhadap sektor pendidikan;
- risiko ransomware terhadap kampus;
- ancaman terhadap infrastruktur publik;
- prioritas investasi keamanan;
- risiko geopolitik digital;
- arah kebijakan pertahanan siber.

Cocok untuk:

- pimpinan kampus;
- kepala unit TI;
- manajer keamanan;
- pembuat kebijakan.

2. Operational Intelligence

Operational intelligence menjelaskan operasi adversary.

Pertanyaan:

- siapa threat actor?
- apa motivasinya?
- campaign apa yang berjalan?
- targetnya siapa?
- bagaimana pola operasinya?
- TTP apa yang sering dipakai?

Rujukan Wazuh menjelaskan bahwa operational intelligence menjawab “who”, “why”, dan “how”; bagian “how” terkait TTP adversary.

3. Tactical Intelligence

Tactical intelligence berisi indikator teknis yang langsung bisa dipakai SOC.

Contoh:

- IP berbahaya;
- domain C2;
- hash file;
- URL phishing;
- signature;
- rule deteksi;
- IOC list.

Rujukan Wazuh menjelaskan tactical intelligence bersifat teknis, berfokus pada IOC, dan sering memiliki masa pakai pendek karena IP atau domain dapat cepat berubah.

4. Technical Intelligence

Technical intelligence lebih detail pada bukti teknis.

Contoh:

- struktur malware;
- pola komunikasi;
- protokol C2;
- YARA rule;
- Suricata rule;
- Zeek script;
- sample konfigurasi;
- eksploitasi vulnerability tertentu.

Untuk mahasiswa, technical intelligence paling cocok dipakai dalam lab karena dapat langsung dihubungkan dengan log, IOC, dan rule.

8.6 Mengapa MISP?

MISP adalah platform open source untuk mengumpulkan, menyimpan, mendistribusikan, dan berbagi indikator ancaman. MISP juga menyediakan API fleksibel dan PyMISP untuk mengambil, menambah, memperbarui event dan atribut, serta mencari indikator. (GitHub)

MISP cocok untuk SOC open source karena:

- **open source**;
- mendukung sharing IOC;
- mendukung event dan attribute;
- mendukung tag dan taxonomy;
- mendukung galaxy untuk konteks lebih kaya;
- mendukung feed;
- memiliki API;
- dapat diekspor dalam JSON;
- bisa dihubungkan ke Wazuh, case management, dan SOAR.

Dalam konteks Wazuh, integrasi MISP dapat dipakai untuk korelasi threat intelligence. Contoh resmi dari komunitas MISP menunjukkan integrasi Wazuh–MISP untuk mencocokkan hash file dari endpoint dengan indikator di MISP, lalu menghasilkan alert jika ada match. (MISP Threat Intelligence Platform)

8.7 Konsep MISP: Event, Attribute, Tag, Galaxy, Feed

Event

Event adalah wadah utama di MISP. Event mewakili satu kejadian, kampanye, laporan, atau kumpulan indikator.

Contoh event:

```
SOC-LAB-008-001 - Simulasi Brute Force Lab
```

Isi event dapat mencakup:

- deskripsi;
- organisasi pembuat;
- tanggal;
- tingkat distribusi;
- threat level;
- analysis status;
- attribute;
- tag;
- galaxy;
- reference.

Attribute

Attribute adalah indikator di dalam event.

Contoh attribute:

- IP address;
- domain;
- URL;
- hash;
- filename;
- email;
- user-agent;
- comment.

MISP API mendukung model data seperti event, object, dan attribute, sehingga SOC dapat membuat dan memperbarui data intelijen secara otomatis. (MISP Threat Intelligence Platform)

Tag

Tag memberi label pada event atau attribute.

Contoh tag:

```
tlp:clear
type:osint
workflow:state="complete"
```

Tag membantu klasifikasi, filtering, dan etika berbagi.

Galaxy

Galaxy memberi konteks lebih kaya, misalnya threat actor, malware family, tool, campaign, atau framework tertentu.

Rujukan Wazuh terlampir juga mencatat MISP memiliki feed layer dan galaxies layer sebagai bagian dari ekosistemnya.

Feed

Feed adalah sumber data threat intelligence yang dapat diaktifkan untuk mengambil indikator dari sumber tertentu.

Feed dapat membantu SOC mendapatkan:

- IOC baru;
- domain berbahaya;
- IP berisiko;
- hash malware;
- referensi OSINT;
- data komunitas.

Namun feed harus dikurasi. Terlalu banyak feed dapat meningkatkan noise.

8.8 IOC dalam Format JSON

JSON penting karena menjadi bahasa pertukaran antar tools SOC.

Contoh IOC sederhana:

```
{
  "case_id": "SOC-LAB-008-001",
  "source": "MISP",
  "ioc": [
    {
      "type": "ip-src",
      "value": "10.10.10.50",
      "comment": "Source IP simulasi brute force lab",
      "tlp": "clear"
    },
    {
      "type": "domain",
      "value": "example-lab.local",
      "comment": "Domain lab untuk latihan DNS analysis",
      "tlp": "clear"
    },
    {
      "type": "sha256",
      "value":
"e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852
b855",
      "comment": "Hash file dummy untuk latihan",
      "tlp": "clear"
    },
    {
      "type": "url",
      "value": "http://example-lab.local/test",
      "comment": "URL lab untuk latihan",
      "tlp": "clear"
    }
  ]
}
```

Catatan penting:

- gunakan data lab;
- jangan pakai data pribadi;
- jangan bagikan indikator yang tidak tervalidasi;
- beri komentar yang jelas;
- gunakan tag distribusi yang sesuai;
- pisahkan IOC simulasi dan IOC nyata.

JSON membuat IOC mudah dipakai oleh Wazuh, OpenSearch, SOAR, dan AI lokal.

8.9 MITRE ATT&CK untuk Membaca Taktik dan Teknik

MITRE ATT&CK adalah basis pengetahuan global tentang taktik dan teknik adversary berdasarkan observasi dunia nyata. ATT&CK dipakai sebagai fondasi untuk threat modeling dan metodologi keamanan di sektor privat, pemerintah, dan komunitas keamanan. (attack.mitre.org)

Dalam SOC, MITRE membantu analis:

- memberi bahasa umum pada alert;
- mapping teknik serangan;
- menyusun deteksi;
- memprioritaskan hunting;
- menulis laporan;
- mengukur coverage deteksi;
- menjelaskan incident kepada tim.

Rujukan SOC analyst menyebut MITRE ATT&CK banyak dipakai untuk threat intelligence, red team, blue team, detection engineering, dan incident response.

Contoh mapping:

Alert	Tactic	Technique	Makna SOC
Login gagal berulang	*Credential Access*	*Brute Force*	Percobaan memperoleh kredensial
Port scan	*Discovery*	*Network Service Discovery*	Pencarian layanan terbuka
File konfigurasi berubah	*Persistence* / *Defense Evasion*	tergantung konteks	Potensi bertahan atau menyembunyikan jejak
Koneksi ke domain C2	*Command and Control*	tergantung pola	Potensi komunikasi keluar
Eksfiltrasi data	*Exfiltration*	tergantung metode	Potensi keluarnya data

MITRE juga menyediakan ATT&CK Navigator untuk mengeksplorasi dan memvisualisasikan teknik, membantu perencanaan red/blue team, serta melihat coverage deteksi. (attack.mitre.org)

8.10 Cyber Kill Chain sebagai Model Alur Serangan

Cyber Kill Chain adalah model untuk memahami tahapan serangan. Rujukan SOC analyst menjelaskan bahwa Cyber Kill Chain digunakan untuk mapping adversary dan pengembangan countermeasure, serta membantu memahami fase serangan agar organisasi dapat menempatkan pertahanan di setiap tahap.

Tahapan umum Cyber Kill Chain:

1. **Reconnaissance**
   Penyerang mengumpulkan informasi target.
2. **Weaponization**
   Penyerang menyiapkan alat atau payload.
3. **Delivery**
   Payload dikirim ke target.
4. **Exploitation**
   Kelemahan dieksploitasi.
5. **Installation**
   Payload atau mekanisme bertahan dipasang.
6. **Command and Control**
   Komunikasi dengan infrastruktur kendali.
7. **Actions on Objectives**
   Penyerang mencapai tujuan, misalnya pencurian data atau sabotase.

Untuk mahasiswa, Cyber Kill Chain berguna sebagai model sederhana. MITRE ATT&CK lebih detail untuk mapping teknik. Keduanya saling melengkapi.

8.11 Etika Berbagi IOC

Berbagi IOC harus hati-hati. Tidak semua indikator boleh dibagikan secara bebas.

Prinsip etika:

- **jangan bagikan data pribadi**;
- **jangan bagikan log mentah yang sensitif**;
- **jangan menuduh organisasi tanpa bukti**;
- **jangan membagikan IP internal yang bisa membuka struktur jaringan**;
- **gunakan tag distribusi yang benar**;
- **bedakan IOC lab dan IOC nyata**;
- **beri konteks, bukan hanya daftar indikator**;
- **validasi sebelum berbagi**;
- **hormati aturan hukum dan kebijakan organisasi**.

Gunakan prinsip sederhana:

> **IOC tanpa konteks bisa menyesatkan; IOC tanpa etika bisa membahayakan.**

8.12 Peran AI Open Source dalam Threat Intelligence

AI open source dapat membantu threat intelligence, tetapi harus dikontrol.

AI dapat membantu:

- merangkum laporan threat intelligence;
- mengekstrak IOC dari teks;
- mengubah IOC menjadi JSON;
- membantu mapping ke MITRE ATT&CK;
- menyusun ringkasan case;
- membuat tabel IOC;
- membantu menjelaskan TTP;
- membantu membuat prompt hunting.

AI tidak boleh:

- menerima data sensitif tanpa sanitasi;
- mengarang atribusi threat actor;
- menyimpulkan campaign tanpa bukti;
- membagikan IOC tanpa validasi;
- menggantikan analis.

Rujukan open-source SOC menekankan bahwa threat intelligence memberi konteks untuk memprioritaskan alert, mengarahkan threat hunting, dan mempercepat incident response; tetapi kualitas sumber, signal-to-noise, validasi analitis, dan relevansi terhadap risiko organisasi tetap menjadi tantangan utama.

> **AI membantu merapikan intelligence. Analis tetap bertanggung jawab atas kebenaran dan konteksnya.**

8.13 Hands-on 1 — Instalasi MISP

Tujuan

Menyiapkan MISP sebagai platform threat intelligence di lab SOC.

Prasyarat

- Linux server lab.
- Docker dan Docker Compose tersedia.
- Jaringan lab terisolasi.
- Password kuat.
- Snapshot sebelum instalasi.
- Akses hanya dari jaringan lab.

Langkah Ringkas

1. Siapkan folder kerja:

```
mkdir -p ~/soc-lab/docker/misp
cd ~/soc-lab/docker/misp
```

Penjelasan:

- `mkdir -p` membuat folder bertingkat.
- `~/soc-lab/docker/misp` adalah lokasi konfigurasi MISP.
- `cd` masuk ke folder kerja.

2. Gunakan panduan resmi MISP atau repository Docker MISP yang dipercaya.
3. Jalankan layanan dengan Docker Compose.
4. Akses antarmuka MISP dari jaringan lab.
5. Ganti password default.
6. Buat user lab.
7. Catat versi.
8. Buat snapshot setelah berhasil login.

Catatan Keamanan

- jangan buka MISP ke internet;
- jangan masukkan IOC nyata tanpa izin;
- pisahkan data lab dan data nyata;
- gunakan tag yang jelas;
- aktifkan akses minimum.

8.14 Hands-on 2 — Mengaktifkan Feed Threat Intelligence

Tujuan

Mengaktifkan feed untuk latihan membaca IOC.

Langkah umum:

1. Masuk ke MISP.
2. Buka menu feed.
3. Pilih feed yang relevan.
4. Aktifkan satu atau dua feed saja.
5. Jalankan fetch.
6. Cek event yang masuk.
7. Catat sumber feed.
8. Evaluasi kualitas indikator.

Prinsip Kurasi Feed

Jangan aktifkan semua feed sekaligus.

Pilih feed berdasarkan:

- relevansi lab;
- kualitas sumber;
- frekuensi update;
- jenis IOC;
- tingkat false positive;
- kejelasan lisensi dan distribusi;
- kemampuan mahasiswa menjelaskan isinya.

Threat feed bukan kebenaran absolut. Feed adalah bahan analisis.

8.15 Hands-on 3 — Membuat Event MISP Sederhana

Tujuan

> Membuat event untuk simulasi SOC lab.

Contoh event:

```
SOC-LAB-008-001 - Simulasi Brute Force Endpoint Linux
```

Isi event:

- Date: tanggal simulasi.
- Threat level: low atau medium untuk lab.
- Analysis: ongoing atau completed.
- Distribution: internal lab.
- Info: ringkasan kasus.

Contoh deskripsi:

> Event ini dibuat untuk latihan threat intelligence pada SOC lab.
> Data di dalam event adalah data simulasi, bukan indikator publik.

Prinsip

- beri nama jelas;
- tandai sebagai lab;
- jangan gunakan data sensitif;
- jangan menyebarkan event simulasi sebagai IOC publik;
- tulis konteks agar tidak disalahpahami.

8.16 Hands-on 4 — Menambahkan IOC

Tambahkan attribute berikut ke event.

IP Address

Type: ip-src
Value: 10.10.10.50
Comment: Source IP simulasi login gagal berulang

Domain

Type: domain
Value: example-lab.local
Comment: Domain lab untuk latihan DNS analysis

Hash

Type: sha256
Value: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Comment: Hash file dummy untuk latihan

URL

Type: url
Value: http://example-lab.local/test
Comment: URL lab untuk simulasi analisis

Catatan

- gunakan IOC lab;
- jangan gunakan IOC yang belum dipahami;
- tulis comment;
- beri tag `lab`;
- jangan publish ke komunitas eksternal.

8.17 Hands-on 5 — Ekspor IOC dalam JSON

Tujuan

> Mengambil IOC dari MISP dalam format yang bisa dipakai Wazuh, OpenSearch, SOAR, atau AI lokal.
>
> MISP menyediakan API dan PyMISP untuk automation. PyMISP dapat mengambil event, menambah atau memperbarui event/attribute, serta mencari attribute melalui REST API MISP. (CIRCL)

Contoh struktur ekspor:

```
{
  "Event": {
    "info": "SOC-LAB-008-001 - Simulasi Brute Force Endpoint Linux",
    "Attribute": [
      {
        "type": "ip-src",
        "value": "10.10.10.50",
        "comment": "Source IP simulasi login gagal"
      },
      {
        "type": "domain",
        "value": "example-lab.local",
        "comment": "Domain lab"
      },
      {
        "type": "sha256",
        "value":
"e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
        "comment": "Hash file dummy"
      }
    ]
  }
}
```

Simpan hasil ekspor:

```
mkdir -p ~/soc-lab/reports/SOC-LAB-008-001
nano ~/soc-lab/reports/SOC-LAB-008-001/ioc.json
```

Penjelasan:

- folder case dibuat agar evidence rapi;
- `ioc.json` menyimpan indikator dalam format terstruktur;
- file ini dapat dilampirkan pada case incident.

8.18 Hands-on 6 — Menghubungkan IOC ke Case Incident

Gunakan case dari Bab 7.

Tambahkan ke case:

> Case ID:
> SOC-LAB-007-001
>
> Related MISP Event:
> SOC-LAB-008-001 - Simulasi Brute Force Endpoint Linux
>
> Observables:
> - 10.10.10.50
> - example-lab.local
> - hash dummy
> - URL lab
>
> Analyst Note:
> IOC berasal dari event MISP lab. Data digunakan untuk latihan korelasi dan tidak dibagikan sebagai indikator publik.

Tujuan:

- case punya konteks threat intelligence;
- IOC tidak berdiri sendiri;
- investigasi lebih rapi;
- laporan lebih profesional;
- mahasiswa belajar hubungan alert → IOC → case → rekomendasi.

8.19 Hands-on 7 — Mapping Alert Wazuh ke MITRE ATT&CK

Ambil contoh alert Wazuh:

> Login gagal berulang pada endpoint Linux.

Mapping sederhana:

> Tactic:
> Credential Access
>
> Technique:
> Brute Force
>
> Evidence:
> Wazuh alert rule 100200, 12 login gagal dalam 5 menit.
>
> Status:
> Valid lab simulation.
>
> Recommendation:
> Perkuat policy autentikasi, batasi akses layanan, monitor source IP, buat alert threshold.

Contoh format JSON:

```
{
  "alert": "Login gagal berulang pada endpoint Linux",
  "source": "Wazuh",
  "rule_id": "100200",
  "mitre": {
    "tactic": "Credential Access",
    "technique": "Brute Force"
  },
  "evidence": [
    "Wazuh alert JSON",
    "OpenSearch query result",
    "case timeline"
  ],
  "recommendation": [
    "monitor source IP",
    "review authentication policy",
    "create threshold alert"
  ]
}
```

8.20 Hands-on 8 — Membuat Tabel IOC untuk Laporan

Gunakan tabel berikut untuk laporan praktikum.

No	IOC Type	Value	Source	Confidence	Context	Action
1	IP	10.10.10.50	MISP Lab	Medium	Source login gagal	Monitor
2	Domain	example-lab.local	MISP Lab	Low	Domain latihan	No external sharing
3	SHA256	e3b0c442...	MISP Lab	Low	Hash dummy	Training only
4	URL	http://example-lab.local/test	MISP Lab	Low	URL latihan	Training only

Tambahkan kolom jika perlu:

- first seen;
- last seen;
- related case;
- MITRE tactic;
- MITRE technique;
- analyst note.

Tabel ini membantu laporan menjadi singkat, jelas, dan dapat diaudit.

8.21 Template Prompt AI Lokal untuk IOC

Gunakan AI lokal hanya setelah data disanitasi.

> Anda adalah asisten SOC analyst.
> Analisis IOC berikut dalam konteks lab.
> Tugas:
> 1. ringkas indikator,
> 2. bedakan IOC dan TTP,
> 3. sarankan mapping MITRE ATT&CK,
> 4. tulis rekomendasi triage,
> 5. jangan membuat klaim di luar data.
>
> Data IOC:
> [masukkan JSON IOC yang sudah disanitasi]

Aturan:

- jangan masukkan kredensial;
- jangan masukkan data pribadi;
- jangan masukkan token API;
- jangan masukkan log mentah sensitif;
- semua output AI harus divalidasi analis.

8.22 Kesalahan Umum dalam Threat Intelligence

Kesalahan yang sering terjadi:

- menganggap semua feed pasti benar;
- mengejar jumlah IOC, bukan kualitas;
- tidak memberi konteks;
- membagikan IOC lab ke luar;
- tidak memakai tag;
- tidak membedakan IOC dan TTP;
- tidak mapping ke MITRE;
- tidak mencatat sumber;
- tidak menilai confidence;
- menyimpulkan threat actor tanpa bukti;
- terlalu percaya AI.

Threat intelligence yang buruk hanya memindahkan noise dari luar ke dalam SOC.

8.23 Checklist Bab 8

Sebelum lanjut ke Bab 9, pastikan:

- Paham beda data, informasi, dan intelligence.
- Paham beda IOC dan TTP.
- Paham jenis threat intelligence.
- MISP terpasang di lab.
- Feed dipilih secara hati-hati.
- Event MISP sederhana dibuat.
- Attribute IOC ditambahkan.
- IOC diekspor dalam JSON.
- IOC dihubungkan ke case incident.
- Alert Wazuh dipetakan ke MITRE ATT&CK.
- Cyber Kill Chain dipahami sebagai model alur serangan.
- Tabel IOC dibuat untuk laporan.
- AI lokal hanya dipakai sebagai asisten, bukan sumber kebenaran.

8.24 Rangkuman Bab

Bab ini menjelaskan bahwa SOC membutuhkan konteks eksternal agar alert tidak dibaca secara sempit. Threat intelligence membantu SOC memprioritaskan alert, memperkaya case, mengarahkan threat hunting, dan mempercepat incident response.

MISP menjadi platform utama karena mendukung event, attribute, tag, galaxy, feed, API, dan JSON. MITRE ATT&CK membantu membaca taktik dan teknik adversary, sedangkan Cyber Kill Chain memberi model sederhana untuk memahami alur serangan.

Inti Bab 8 adalah **konteks**. IOC memberi indikator teknis. TTP memberi pola perilaku. MITRE memberi bahasa bersama. MISP memberi wadah pengelolaan. AI lokal membantu merangkum, tetapi analis tetap wajib memvalidasi.

8.25 Output Bab

Setelah menyelesaikan Bab 8, pembaca diharapkan mampu:

- menjelaskan apa itu *threat intelligence*;
- membedakan data, informasi, dan intelligence;
- membedakan IOC dan TTP;
- menjelaskan strategic, operational, tactical, dan technical intelligence;
- memahami konsep MISP: event, attribute, tag, galaxy, dan feed;
- membuat event MISP sederhana;
- menambahkan IOC berupa IP, domain, hash, dan URL;
- mengekspor IOC dalam JSON;
- menghubungkan IOC ke case incident;
- mapping alert Wazuh ke MITRE ATT&CK;
- memakai Cyber Kill Chain untuk membaca alur serangan;
- membuat tabel IOC untuk laporan;
- menerapkan etika berbagi IOC;
- memakai AI open source sebagai asisten analis, bukan pengganti bukti.

Pesan akhir Bab 8:
Threat intelligence membuat SOC melihat lebih jauh: dari alert lokal menjadi konteks ancaman, dari IOC menjadi TTP, dan dari data menjadi keputusan pertahanan yang lebih cerdas.

Bab 9 — SOAR dan Automation: Dari Alert ke Respons Terukur

> **Otomasi SOC yang baik bukan yang paling cepat mengeksekusi, tetapi yang paling aman, dapat diaudit, dan tetap berada di bawah kendali analis.**

9.1 Arah Bab

Bab ini membahas **SOAR** dan otomasi SOC. Setelah Bab 7 membahas *case management* dan Bab 8 membahas *threat intelligence*, Bab 9 mengajarkan bagaimana alert, IOC, case, dan tindakan respons dapat dihubungkan dalam *workflow* yang terukur.

Dalam outline buku, Bab 9 diarahkan untuk membahas **SOAR, script, workflow, playbook, webhook, JSON, enrichment IOC, pembuatan case, notifikasi, approval, blokir IP di lab, dan audit trail otomasi**.

Namun pesan utamanya tegas:

SOAR bukan berarti semua respons harus otomatis.
SOAR berarti SOC memiliki alur kerja yang jelas, berulang, terdokumentasi, dan dapat dikontrol.

Shuffle dipilih karena merupakan platform otomasi open source yang dirancang untuk kebutuhan keamanan. Dokumentasi Shuffle menyebut Shuffle sebagai platform otomasi open source untuk industri keamanan yang dapat dijalankan secara *on-premises*. (GitHub) n8n dapat dipakai sebagai alternatif workflow automation umum karena mendukung workflow visual, kode, deployment mandiri, dan integrasi API. (GitHub)

9.2 Apa Itu SOAR?

SOAR adalah singkatan dari *Security Orchestration, Automation, and Response*.

Secara sederhana:

- **Security Orchestration** berarti menghubungkan banyak tools keamanan.
- **Automation** berarti menjalankan langkah berulang secara otomatis.
- **Response** berarti membantu SOC merespons alert atau incident secara terstruktur.

SOAR membantu SOC menghubungkan:

- Wazuh;
- OpenSearch;
- MISP;
- DFIR-IRIS;
- Suricata;
- Zeek;
- webhook;
- API;
- script Python;
- AI lokal;
- notifikasi;
- laporan.

Rujukan SOC menekankan bahwa *incident response automation and orchestration* berhubungan dengan playbook, workflow, threat intelligence gathering, integrasi sistem, efektivitas respons, dan governance manusia.

Tujuan SOAR bukan menggantikan analis. Tujuannya adalah:

- mengurangi pekerjaan berulang;
- mempercepat triage;
- memperkaya alert;
- membuat case lebih konsisten;
- mencatat tindakan;
- mengurangi lupa langkah;
- memperbaiki MTTD dan MTTR;
- memastikan respons dapat diaudit.

SOAR adalah disiplin workflow, bukan tombol ajaib.

9.3 Mengapa SOC Membutuhkan Otomasi?

SOC menerima banyak alert. Tidak semuanya penting, tetapi semuanya butuh perhatian. Tanpa otomasi, analis mudah kelelahan.

Otomasi membantu pada bagian yang berulang:

- menerima alert JSON;
- parsing field penting;
- mengambil source IP;
- cek IOC di MISP;
- membuat case;
- menambahkan observable;
- mengirim notifikasi;
- membuat draft laporan;
- mencatat audit log.

Namun bagian yang membutuhkan penilaian tetap harus dilakukan manusia:

- menentukan incident atau bukan;
- memutuskan containment;
- menilai dampak bisnis;
- memutuskan eskalasi;
- menutup case;
- menyatakan breach;
- membuat rekomendasi final.

Rujukan SOC analyst menekankan bahwa otomasi membantu efisiensi, tetapi SOC tetap membutuhkan penilaian analis, konteks, dan prioritas agar respons tidak salah arah.

> **Otomasi mempercepat proses. Analis menjaga akurasi.**

9.4 Script, Workflow, dan Playbook

Mahasiswa perlu membedakan tiga istilah ini.

Script

Script adalah potongan kode untuk menjalankan tugas tertentu.

Contoh:

- mengambil alert dari API;
- mengekstrak IP dari JSON;
- mengecek hash ke MISP;
- menulis hasil ke file;
- mengirim notifikasi lokal.

Script biasanya kecil dan spesifik.

Workflow

Workflow adalah rangkaian langkah.

Contoh:

```
Terima alert
→ parsing JSON
→ cek IOC
→ buat case
→ kirim notifikasi
→ tulis audit log
```

Workflow mengatur urutan kerja.

Playbook

Playbook adalah workflow yang sudah diberi konteks keamanan, aturan, keputusan, dan prosedur.

Playbook berisi:

- trigger;
- input;
- kondisi;
- langkah analisis;
- enrichment;
- approval;
- tindakan;
- logging;
- rollback;
- output;
- kriteria penutupan.

Contoh playbook:

```
SSH brute force detected
```

Isi playbook:

- jika alert brute force muncul;
- ambil source IP;
- cek apakah IP internal atau eksternal lab;
- cek jumlah percobaan;
- cek apakah ada login berhasil;
- cek reputasi IOC;
- buat case jika memenuhi threshold;
- minta approval sebelum blokir;
- catat tindakan.

Script menjalankan tugas. Workflow menghubungkan tugas. Playbook memberi konteks SOC.

9.5 Konsep Dasar SOAR: Trigger, Condition, Action, Approval, Logging

1. Trigger

Trigger adalah pemicu workflow.

Contoh:

- alert Wazuh masuk;
- query OpenSearch melewati threshold;
- IOC baru masuk dari MISP;
- case baru dibuat;
- webhook menerima JSON;
- file laporan diunggah.

OpenSearch Alerting memakai monitor yang berjalan sesuai jadwal, trigger untuk kondisi pemicu, alert sebagai hasil kondisi, dan action sebagai tugas setelah alert aktif. (OpenSearch Documentation)

2. Condition

Condition adalah syarat.

Contoh:

```
Jika severity >= 8
dan event_type = authentication_failed
dan count > 10 dalam 5 menit
maka lanjutkan workflow.
```

Condition mencegah otomasi berjalan sembarangan.

3. Action

Action adalah tindakan.

Contoh:

- ambil detail alert;
- cek IOC;
- buat case;
- kirim notifikasi;
- tulis audit log;
- blokir IP di lab;
- buat draft laporan.

OpenSearch action dapat dikonfigurasi setelah trigger aktif, misalnya untuk mengirim pesan atau notifikasi berdasarkan hasil monitor. (OpenSearch Documentation)

4. Approval

Approval adalah persetujuan analis sebelum tindakan berisiko.

Wajib dipakai untuk:

- blokir IP;
- menonaktifkan akun;
- isolasi host;
- menghapus file;
- menghentikan service;
- menutup incident.

5. Logging

Logging adalah catatan semua tindakan otomasi.

Audit log harus mencatat:

- workflow apa berjalan;
- kapan berjalan;
- alert apa memicu;
- data apa dipakai;
- tindakan apa dilakukan;
- siapa memberi approval;
- hasil tindakan;
- error yang muncul.

Tanpa audit log, otomasi tidak bisa dipertanggungjawabkan.

9.6 Kapan Respons Boleh Otomatis?

Respons boleh otomatis jika risikonya rendah dan mudah dibatalkan.

Contoh yang relatif aman:

- membuat case;
- menambahkan observable;
- menulis audit log;
- mengirim notifikasi;
- membuat draft laporan;
- mengambil enrichment IOC;
- memberi label severity;
- membuat task analis;
- menyimpan alert JSON.

Respons seperti ini tidak mengubah sistem target secara langsung.

Prinsip Respons Otomatis Aman

- hanya berjalan di lab;
- trigger jelas;
- threshold jelas;
- tidak menghapus bukti;
- tidak memutus layanan penting;
- hasil dapat diaudit;
- mudah diulang;
- mudah dihentikan.

9.7 Kapan Harus Menunggu Persetujuan Analis?

Respons harus menunggu approval jika tindakan dapat berdampak pada layanan, bukti, atau pengguna.

Contoh:

- blokir IP;
- isolasi host;
- menghentikan service;
- menghapus file;
- mengubah konfigurasi firewall;
- menonaktifkan user;
- menghapus IOC;
- menutup case;
- menyatakan incident selesai.

Wazuh memiliki fitur *Active Response* untuk menjalankan script berdasarkan trigger tertentu, termasuk contoh pemblokiran IP pada skenario brute force. Namun dokumentasi Wazuh menunjukkan konteks ini sebagai use case yang harus dikonfigurasi hati-hati karena tindakan dilakukan otomatis pada endpoint yang dipantau. (documentation.wazuh.com)

> **Tindakan yang mengubah sistem harus punya kontrol manusia.**

9.8 Risiko Otomasi SOC

1. False Positive

Jika alert salah, otomasi juga salah. Ini bisa membuat SOC melakukan tindakan yang tidak perlu.

Contoh:

- aktivitas admin lab dianggap serangan;
- scanning legal dianggap malicious;
- IP internal diblokir;
- user sah dicurigai.

2. Blokir Aset Penting

Otomasi yang terlalu agresif bisa memutus akses ke server penting.

Untuk lab, ini hanya mengganggu praktikum. Untuk organisasi nyata, ini bisa menghentikan layanan.

3. Kehilangan Bukti

Tindakan terlalu cepat dapat menghapus bukti.

Contoh:

- file dihapus sebelum di-hash;
- service dihentikan sebelum log diambil;
- host dimatikan sebelum memory/artifact dicatat;
- rule active response mengubah kondisi sistem.

4. Respons Berlebihan

Tidak semua alert butuh respons keras.

Contoh:

- login gagal satu kali tidak perlu blokir;
- port scan lab tidak perlu emergency;
- IOC low confidence tidak perlu containment;
- alert severity medium perlu triage, bukan panic.

5. Ketergantungan pada Tools

Jika analis hanya percaya workflow, kemampuan investigasi melemah.

SOC harus tetap melatih:

- membaca log;
- memahami rule;
- memvalidasi IOC;
- membaca timeline;
- membuat keputusan manual.

Otomasi yang buruk mempercepat kesalahan.

9.9 Shuffle SOAR dan Webhook

Shuffle dipakai sebagai platform SOAR open source untuk praktik. Shuffle menyediakan workflow automation yang cocok untuk keamanan dan dapat dipasang di infrastruktur sendiri. (GitHub)

Dalam Shuffle, konsep penting:

- trigger;
- app;
- action;
- workflow;
- webhook;
- variable;
- execution log;
- condition;
- approval/manual step.

Webhook adalah pintu masuk data. Alert dari Wazuh atau OpenSearch dapat dikirim sebagai JSON ke webhook, lalu Shuffle menjalankan workflow.

Alur sederhana:

```
Wazuh / OpenSearch
      ↓
Webhook Shuffle
      ↓
Parse JSON
      ↓
Cek IOC ke MISP
      ↓
Buat Case
      ↓
Kirim Notifikasi
      ↓
Audit Log
```

Dokumentasi Shuffle menjelaskan trigger sebagai operator untuk menjalankan workflow secara otomatis dan biasanya menjadi node awal workflow. (shuffle.empowerists.com)

9.10 n8n sebagai Alternatif Workflow Automation Umum

n8n dapat dipakai sebagai alternatif workflow automation umum. n8n mendukung workflow visual, integrasi API, dan deployment di infrastruktur sendiri. Repository resminya menjelaskan n8n sebagai platform workflow automation untuk tim teknis yang menggabungkan fleksibilitas kode dan kecepatan *no-code*, dengan kontrol atas data dan deployment. (GitHub)

n8n cocok untuk:

- workflow non-SOC;
- integrasi API sederhana;
- eksperimen AI lokal;
- data transformation;
- webhook;
- parsing JSON;
- laporan otomatis;
- integrasi dengan script.

Namun untuk buku ini, **Shuffle lebih tepat sebagai jalur utama SOAR**, karena memang dirancang untuk otomasi keamanan. n8n dapat dijadikan alternatif untuk latihan workflow umum.

Catatan keamanan:

- jangan membuka endpoint webhook ke internet tanpa kontrol;
- gunakan autentikasi;
- batasi akses jaringan;
- simpan credential dengan aman;
- update berkala;
- jangan menyimpan token di workflow publik.

9.11 Format JSON untuk Integrasi Tools SOC

JSON adalah bahasa utama integrasi SOC.

Contoh alert JSON:

```
{
  "alert_id": "SOC-ALERT-009-001",
  "timestamp": "2026-01-01T10:00:00Z",
  "source": "Wazuh",
  "event_type": "authentication_failed",
  "rule_id": "100200",
  "severity": 8,
  "host": "linux-endpoint-01",
  "source_ip": "10.10.10.50",
  "destination_ip": "10.10.10.21",
  "mitre_tactic": "Credential Access",
  "mitre_technique": "Brute Force"
}
```

Field minimum:

- `alert_id`;
- `timestamp`;
- `source`;
- `event_type`;
- `rule_id`;
- `severity`;
- `host`;
- `source_ip`;
- `destination_ip`;
- `mitre_tactic`;
- `mitre_technique`.

JSON harus:

- ringkas;
- terstruktur;
- tidak membawa data sensitif berlebihan;
- mudah diparse;
- mudah disimpan;
- mudah dikirim ke case management;
- mudah dibaca AI lokal setelah disanitasi.

9.12 Playbook Dasar SOC

Playbook 1 — Enrich IP

Tujuan:

- mengambil `source_ip`;
- cek apakah IP internal atau eksternal lab;
- cek apakah ada di MISP;
- cek apakah muncul di alert lain;
- tulis hasil enrichment.

Output:

```
{
  "source_ip": "10.10.10.50",
  "scope": "lab_internal",
  "misp_match": false,
  "recommendation": "monitor and validate with logs"
}
```

Playbook 2 — Cek Reputasi Domain

Tujuan:

- ambil domain dari DNS log;
- cek MISP;
- cek feed internal;
- catat hasil;
- buat task jika mencurigakan.

Output:

```
{
  "domain": "example-lab.local",
  "source": "zeek_dns",
  "misp_match": false,
  "classification": "lab_domain"
}
```

Playbook 3 — Buat Case

Tujuan:

- jika severity tinggi;
- jika confidence cukup;
- jika aset penting;
- buat case di DFIR-IRIS;
- tambahkan observable;
- tambahkan timeline awal;
- tambahkan task.

Output:

```
{
  "case_id": "SOC-LAB-009-001",
  "status": "created",
  "owner": "soc-lab-team",
  "tasks_created": 4
}
```

Playbook 4 — Kirim Notifikasi

Tujuan:

- memberi tahu analis;
- mengirim ringkasan;
- menyertakan link case;
- menyertakan severity;
- mencatat waktu.

Notifikasi harus singkat:

```
Alert SOC Lab:
Login gagal berulang pada linux-endpoint-01.
Severity: 8.
Source IP: 10.10.10.50.
Case: SOC-LAB-009-001.
Status: menunggu triage analis.
```

Playbook 5 — Blok IP di Lab

Tujuan:

- hanya untuk lab;
- hanya setelah approval;
- hanya dengan durasi terbatas;
- semua tindakan dicatat.

Wazuh menyediakan contoh *Active Response* untuk memblokir IP pada skenario SSH brute force ketika rule brute force terpicu. (documentation.wazuh.com) Namun dalam buku ini, blok IP harus memakai **approval mode** agar mahasiswa memahami risiko.

9.13 Audit Trail Otomasi

Audit trail adalah catatan tindakan workflow.

Audit trail harus menjawab:

- workflow apa yang berjalan?
- siapa/apa yang memicu?
- kapan berjalan?
- input JSON apa?
- enrichment apa dilakukan?
- case apa dibuat?
- action apa dijalankan?
- siapa memberi approval?
- apa hasilnya?
- apakah ada error?
- apakah rollback dilakukan?

Contoh audit log:

```
{
  "workflow_id": "SOAR-PLAYBOOK-SSH-BRUTE-FORCE",
  "execution_id": "EXEC-009-001",
  "timestamp": "2026-01-01T10:05:00Z",
  "trigger": "wazuh_alert_webhook",
  "alert_id": "SOC-ALERT-009-001",
  "action": [
    "parse_json",
    "enrich_ip",
    "create_case",
    "send_notification"
  ],
  "approval_required": true,
  "approved_by": "analyst-lab-01",
  "final_action": "block_ip_lab",
  "status": "success"
}
```

Audit trail membuat otomasi dapat dipercaya.

9.14 Peran AI Open Source dalam SOAR

AI open source dapat membantu workflow SOC, tetapi harus ditempatkan dengan aman.

AI dapat dipakai untuk:

- membuat ringkasan alert;
- membuat draft case note;
- menulis laporan awal;
- menyarankan task investigasi;
- menjelaskan hasil enrichment;
- membuat summary dari banyak alert;
- membantu membuat rekomendasi triage.

AI tidak boleh dipakai untuk:

- memutuskan blokir otomatis;
- menutup case otomatis;
- menyatakan breach;
- menghapus data;
- menjalankan script respons;
- menerima token API;
- menerima data sensitif mentah.

Format aman:

```
Input AI:
- alert_id
- event_type
- severity
- source_ip sudah disamarkan jika perlu
- ringkasan log
- hasil enrichment

Output AI:
- ringkasan
- pertanyaan triage
- draft laporan
- rekomendasi non-eksekusi
```

AI boleh membuat draft. Analis yang memberi keputusan.

9.15 Hands-on 1 — Instalasi Shuffle

Tujuan

Menyiapkan platform SOAR open source untuk lab.

Prasyarat

- Docker aktif.
- Docker Compose aktif.
- Folder `soc-lab/` tersedia.
- Jaringan lab terisolasi.
- Password kuat.
- Snapshot sudah dibuat.

Struktur Folder

```
mkdir -p ~/soc-lab/docker/shuffle
cd ~/soc-lab/docker/shuffle
```

Penjelasan:

- `mkdir -p` membuat folder bertingkat.
- `~/soc-lab/docker/shuffle` menjadi lokasi konfigurasi Shuffle.
- `cd` masuk ke folder kerja.

Prinsip Instalasi

Gunakan dokumentasi resmi Shuffle atau repository resminya.

Setelah instalasi:

- jangan buka dashboard ke internet;
- buat user lab;
- ganti password default;
- batasi akses dari jaringan lab;
- catat versi;
- buat snapshot.

9.16 Hands-on 2 — Membuat Webhook Penerima Alert JSON

Tujuan

Membuat pintu masuk alert dari Wazuh/OpenSearch ke Shuffle.

Langkah Konseptual

1. Buat workflow baru.
2. Tambahkan trigger webhook.
3. Salin URL webhook.
4. Siapkan sample alert JSON.
5. Kirim sample alert ke webhook.
6. Cek execution log.

Contoh sample JSON:

```
{
  "alert_id": "SOC-ALERT-009-001",
  "timestamp": "2026-01-01T10:00:00Z",
  "source": "Wazuh",
  "event_type": "authentication_failed",
  "rule_id": "100200",
  "severity": 8,
  "host": "linux-endpoint-01",
  "source_ip": "10.10.10.50"
}
```

Contoh pengiriman:

```
curl -X POST
"http://shuffle-lab.local/webhook/isi-url-webhook" \
  -H "Content-Type: application/json" \
  -d @alert-dummy.json
```

Penjelasan:

- `curl` mengirim request HTTP.
- `-X POST` memakai metode POST.
- URL adalah webhook Shuffle lab.
- `-H` menyatakan data JSON.
- `-d @alert-dummy.json` mengirim isi file JSON.

9.17 Hands-on 3 — Parsing JSON

Tujuan

Mengambil field penting dari alert.

Field yang diambil:

```
alert_id
timestamp
source
event_type
rule_id
severity
host
source_ip
```

Hasil parsing:

```
{
  "alert_id": "SOC-ALERT-009-001",
  "event_type": "authentication_failed",
  "severity": 8,
  "host": "linux-endpoint-01",
  "source_ip": "10.10.10.50"
}
```

Pertanyaan validasi:

- apakah `source_ip` ada?
- apakah `severity` angka?
- apakah `event_type` sesuai?
- apakah `alert_id` unik?
- apakah timestamp valid?

Jika field tidak lengkap, workflow jangan lanjut ke action berisiko.

9.18 Hands-on 4 — Enrich IOC

Tujuan

Mengecek apakah `source_ip` ada dalam MISP atau daftar IOC lab.

Alur:

```
Ambil source_ip
    ↓
Cek format IP
    ↓
Cek daftar IOC lab / MISP
    ↓
Tambahkan hasil enrichment
    ↓
Tulis ke audit log
```

Output enrichment:

```
{
  "source_ip": "10.10.10.50",
  "ioc_match": false,
  "confidence": "medium",
  "note": "IP berasal dari jaringan lab dan sesuai skenario
simulasi."
}
```

Jika `ioc_match = true`, workflow dapat menaikkan prioritas case, tetapi tetap harus menunggu analis untuk tindakan berisiko.

9.19 Hands-on 5 — Membuat Case Otomatis

Tujuan

Membuat case jika kondisi terpenuhi.

Contoh condition:

```
event_type = authentication_failed
severity >= 8
source_ip tersedia
```

Case yang dibuat:

```
Title:
SOC-LAB-009-001 - SSH Brute Force Detected

Summary:
Wazuh mendeteksi login gagal berulang pada endpoint Linux
lab.
Alert diterima melalui webhook SOAR dan membutuhkan triage
analis.

Severity:
Medium

Status:
Under investigation
```

Task otomatis:

- validasi alert Wazuh;
- cek OpenSearch query;
- cek Zeek connection log;
- cek apakah ada login berhasil;
- putuskan apakah perlu blokir IP di lab.

9.20 Hands-on 6 — Membuat Playbook “SSH Brute Force Detected”

Struktur Playbook

Trigger:
Webhook menerima alert Wazuh.

Condition:
event_type = authentication_failed
severity >= 8

Action 1:
Parse JSON.

Action 2:
Enrich source_ip.

Action 3:
Buat case.

Action 4:
Tambahkan observable.

Action 5:
Kirim notifikasi ke analis.

Approval:
Analis memutuskan apakah IP diblokir di lab.

Action 6:
Jika approved, blok IP sementara di lab.

Action 7:
Tulis audit log.

Output:
Case dibuat, alert tercatat, tindakan terdokumentasi.

Prinsip

- jangan blokir otomatis tanpa approval;
- jangan hapus data;
- jangan mengubah endpoint tanpa catatan;
- semua langkah harus terlihat di execution log;
- semua output disimpan.

9.21 Hands-on 7 — Approval Sebelum Blok IP

Tujuan

Memastikan tindakan berisiko tidak berjalan tanpa persetujuan manusia.

Approval harus menjawab:

- siapa analis yang menyetujui?
- kapan disetujui?
- alert apa yang menjadi dasar?
- berapa lama blokir dilakukan?
- bagaimana rollback?
- apa risiko tindakan?

Format approval:

```
{
  "approval_id": "APPROVAL-009-001",
  "case_id": "SOC-LAB-009-001",
  "approved_by": "analyst-lab-01",
  "approved_at": "2026-01-01T10:15:00Z",
  "approved_action": "block_ip_lab",
  "duration_minutes": 15,
  "rollback": "remove temporary block rule",
  "reason": "Repeated failed login simulation confirmed in
lab."
}
```

Approval bukan formalitas. Approval adalah kontrol risiko.

9.22 Hands-on 8 — Membuat Log Hasil Automation

Simpan log eksekusi di:

```
~/soc-lab/reports/SOC-LAB-009-001/automation-log.json
```

Contoh:

```
{
  "case_id": "SOC-LAB-009-001",
  "workflow": "SSH Brute Force Detected",
  "trigger_time": "2026-01-01T10:05:00Z",
  "steps": [
    {
      "name": "receive_alert",
      "status": "success"
    },
    {
      "name": "parse_json",
      "status": "success"
    },
    {
      "name": "enrich_ip",
      "status": "success"
    },
    {
      "name": "create_case",
      "status": "success"
    },
    {
      "name": "send_notification",
      "status": "success"
    },
    {
      "name": "approval_required",
      "status": "waiting"
    }
  ],
  "final_status": "waiting_analyst_approval"
}
```

Audit log ini menjadi bukti bahwa otomasi berjalan dengan kontrol.

9.23 Kesalahan Umum dalam SOAR

Kesalahan yang sering terjadi:

- semua alert langsung dibuat incident;
- blokir otomatis tanpa approval;
- workflow tidak punya audit log;
- credential disimpan sembarangan;
- webhook terbuka ke publik;
- field JSON tidak divalidasi;
- tidak ada rollback;
- tidak membedakan lab dan produksi;
- AI diberi hak eksekusi;
- tidak menguji false positive;
- tidak menulis dokumentasi playbook.

SOAR yang tidak diaudit adalah risiko baru bagi SOC.

9.24 Checklist Bab 9

Sebelum lanjut ke Bab 10, pastikan:

- Paham konsep SOAR.
- Paham beda script, workflow, dan playbook.
- Paham trigger, condition, action, approval, logging.
- Shuffle tersedia di lab.
- Webhook penerima alert JSON dibuat.
- Alert dummy dari Wazuh/OpenSearch berhasil diterima.
- JSON berhasil diparse.
- IOC enrichment dibuat.
- Case otomatis dibuat.
- Notifikasi analis dibuat.
- Playbook “SSH Brute Force Detected” dibuat.
- Approval sebelum blok IP diterapkan.
- Audit log automation tersimpan.
- AI lokal hanya membuat ringkasan, bukan menjalankan action.

9.25 Rangkuman Bab

Bab ini menjelaskan SOAR dan otomasi SOC dengan pendekatan aman. Shuffle dipakai sebagai platform utama karena dirancang untuk otomasi keamanan open source. n8n dapat dipakai sebagai alternatif workflow umum, terutama untuk eksperimen API dan data transformation.

Mahasiswa belajar bahwa otomasi tidak boleh menggantikan analis. Workflow harus memiliki trigger, condition, action, approval, dan audit log. Playbook yang baik tidak hanya menjalankan action, tetapi juga mengontrol risiko, mencatat evidence, dan memastikan tindakan dapat dijelaskan.

Inti Bab 9 adalah **kontrol**. SOC boleh otomatis, tetapi tidak boleh liar.

9.26 Output Bab

Setelah menyelesaikan Bab 9, pembaca diharapkan mampu:

- menjelaskan apa itu SOAR;
- membedakan script, workflow, dan playbook;
- menjelaskan trigger, condition, action, approval, dan logging;
- memahami kapan respons boleh otomatis;
- memahami kapan respons harus menunggu approval analis;
- memahami risiko false positive, blokir aset penting, kehilangan bukti, dan respons berlebihan;
- memasang Shuffle di lab;
- membuat webhook penerima alert JSON;
- mengirim alert dummy dari Wazuh/OpenSearch;
- membuat workflow parsing JSON;
- melakukan enrichment IOC sederhana;
- membuat case otomatis;
- membuat notifikasi analis;
- membuat playbook “SSH Brute Force Detected”;
- membuat approval sebelum blok IP;
- membuat audit log automation;
- memakai AI open source sebagai asisten ringkasan dan dokumentasi, bukan eksekutor respons.

Pesan akhir Bab 9:
SOC yang kuat tidak hanya cepat merespons; SOC yang kuat merespons dengan bukti, kontrol, persetujuan, dan catatan yang dapat diaudit.

Bab 10 — Proyek Akhir: Mini SOC Berbasis Open Source

> **Portofolio SOC terbaik bukan yang paling banyak memasang tools, tetapi yang paling jelas menunjukkan alur data, deteksi, investigasi, respons, dan pelajaran yang bisa dibuktikan.**

10.1 Arah Bab

Bab ini adalah puncak buku. Semua kompetensi dari Bab 1 sampai Bab 9 digabung menjadi satu proyek akhir: **Mini SOC berbasis open source**.

Dalam outline buku, Bab 10 diarahkan sebagai proyek akhir yang menggabungkan Wazuh, Security Onion atau Suricata/Zeek, OpenSearch, MISP, DFIR-IRIS/TheHive, Shuffle/n8n, dataset, skenario uji, KPI SOC, dokumentasi, presentasi, publikasi aman, etika, dan roadmap karier SOC analyst.

Target proyek ini bukan sekadar “berhasil install”. Targetnya adalah mahasiswa mampu membangun sistem kecil yang dapat:

- menerima log;
- menghasilkan alert;
- membaca trafik jaringan;
- melakukan query;
- membuat dashboard;
- memperkaya alert dengan *threat intelligence*;
- membuat case;
- menjalankan playbook otomasi aman;
- menulis laporan;
- mempresentasikan hasil.

Proyek akhir ini harus menjadi portofolio akademik dan awal portofolio karier.

10.2 Tujuan Proyek Akhir

Tujuan utama proyek:

> **membangun Mini SOC berbasis open source yang mampu mendeteksi, menganalisis, mendokumentasikan, dan merespons skenario serangan sederhana di lab tertutup.**

Tujuan teknis:

- memasang minimal 3 komponen SOC open source;
- mengumpulkan log dari minimal 2 sumber;
- membuat minimal 3 rule atau query deteksi;
- membuat minimal 1 dashboard;
- membuat minimal 1 case incident;
- membuat minimal 1 playbook automation;
- menulis laporan akhir 5–10 halaman;
- mempresentasikan hasil dalam 10 menit.

Tujuan pembelajaran:

- memahami alur SOC end-to-end;
- membedakan event, alert, incident, dan case;
- membaca bukti teknis;
- menggunakan AI open source sebagai asisten analisis;
- menulis laporan yang dapat dipertanggungjawabkan;
- membangun portofolio yang aman untuk dibagikan.

10.3 Prinsip Proyek

Proyek harus mengikuti prinsip berikut.

1. Open Source First

Gunakan tools terbuka:

- Wazuh;
- Security Onion;
- Suricata;
- Zeek;
- OpenSearch;
- MISP;
- DFIR-IRIS;
- Shuffle;
- n8n bila dibutuhkan;
- Linux;
- Docker;
- Python;
- osquery;
- YARA.

Wazuh secara resmi adalah platform keamanan *free and open source* yang menyatukan kemampuan SIEM dan XDR. (Dokumentasi Wazuh) Security Onion adalah platform gratis dan terbuka untuk *network visibility*, *host visibility*, *intrusion detection*, *log management*, dan *case management*. (Dokumentasi Security Onion) OpenSearch adalah suite open source berlisensi Apache 2.0 untuk *ingest*, *search*, *visualize*, dan *analyze data*. (OpenSearch)

2. Lab Tertutup

Semua eksperimen dilakukan di jaringan lokal atau virtual.

Tidak boleh:

- menyerang sistem publik;
- memindai alamat di luar lab;
- memakai target tanpa izin;
- membagikan data sensitif;
- membuka dashboard SOC ke internet.

3. Bukti Lebih Penting daripada Klaim

Setiap klaim harus didukung bukti:

- alert JSON;
- log Wazuh;
- log Suricata;
- log Zeek;
- query OpenSearch;
- IOC MISP;
- timeline case;
- screenshot dashboard;
- audit log SOAR.

4. AI sebagai Asisten, Bukan Pengganti Analis

AI open source boleh membantu:

- merangkum alert;
- menyusun laporan;
- membuat daftar pertanyaan triage;
- membantu mapping MITRE;
- membuat ringkasan presentasi.

AI tidak boleh:

- memutuskan incident sendirian;
- mengeksekusi respons otomatis;
- menerima token API;
- menerima log sensitif tanpa sanitasi;
- membuat klaim tanpa bukti.

10.4 Spesifikasi Minimum Proyek

Komponen Minimal

Mahasiswa wajib memakai minimal 3 komponen dari daftar berikut:

- **Wazuh** untuk SIEM/XDR endpoint.
- **Security Onion** atau kombinasi **Suricata + Zeek** untuk monitoring jaringan.
- **OpenSearch** untuk query dan dashboard.
- **MISP** untuk *threat intelligence*.
- **DFIR-IRIS** untuk *case management*.
- **Shuffle** untuk SOAR.
- **AI lokal open source** untuk ringkasan alert dan laporan.

MISP adalah solusi open source untuk mengumpulkan, menyimpan, mendistribusikan, dan berbagi indikator serta ancaman siber. (MISP Threat Intelligence) DFIR-IRIS secara resmi menyebut dirinya sebagai platform *free and open-source* untuk incident response kolaboratif. (DFIR-IRIS)

Komponen Wajib Minimum

Agar proyek tetap realistis, gunakan konfigurasi minimum:

- Wazuh;
- OpenSearch dashboard;
- DFIR-IRIS;
- satu sensor jaringan atau log jaringan sederhana;
- satu workflow automation.

Sumber Data Minimum

Minimal 2 sumber log:

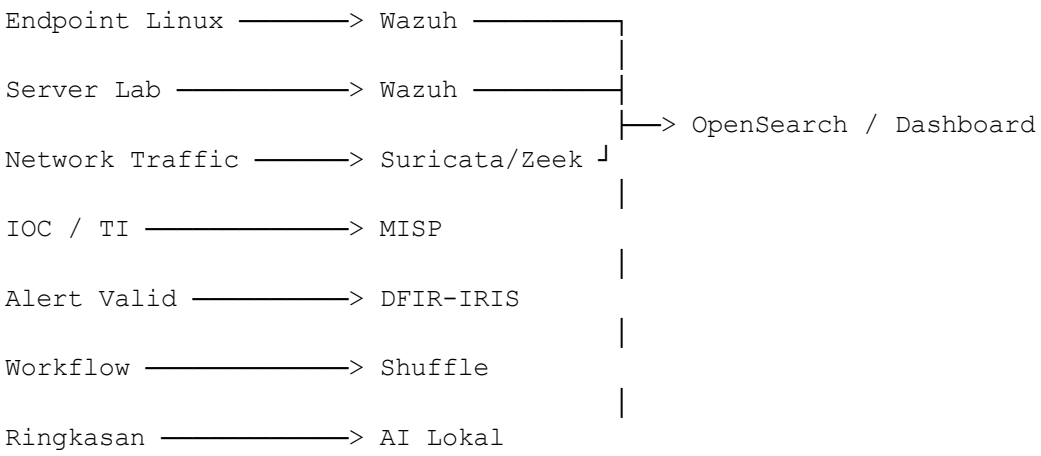
- endpoint Linux;
- server lab;
- sensor jaringan;
- log aplikasi lab;
- alert Suricata;
- metadata Zeek;
- alert Wazuh.

Output Minimum

- 3 deteksi;
- 1 dashboard;
- 1 case;
- 1 playbook;
- 1 laporan;
- 1 presentasi 10 menit.

10.5 Arsitektur Mini SOC

Arsitektur konseptual:



Peran setiap komponen:

- **Wazuh**: endpoint log, FIM, alert, rule, vulnerability.
- **Suricata**: alert jaringan berbasis signature.
- **Zeek**: metadata jaringan, koneksi, DNS, HTTP.
- **OpenSearch**: pencarian, dashboard, visualisasi, query.
- **MISP**: IOC dan konteks threat intelligence.
- **DFIR-IRIS**: case, observable, evidence, timeline, report.
- **Shuffle**: workflow automation, webhook, enrichment, audit log.
- **AI lokal**: ringkasan, draft laporan, bantuan analisis.

Security Onion dapat dipakai sebagai platform terpadu untuk visibilitas jaringan karena mencakup Suricata, Zeek, PCAP, alerting, hunting, dan case management. (GitHub)

10.6 Dataset dan Skenario Uji

Dataset proyek tidak perlu besar. Yang penting jelas, terstruktur, dan dapat diuji ulang.

Dataset Minimum

Gunakan data dari:

- alert Wazuh;
- FIM;
- log login;
- alert Suricata;
- Zeek `conn.log`;
- Zeek `dns.log`;
- sample IOC MISP;
- audit log Shuffle;
- case DFIR-IRIS.

Format Dataset

Gunakan JSON atau CSV.

Contoh field:

```
timestamp
source
host
source_ip
destination_ip
event_type
rule_id
severity
mitre_tactic
mitre_technique
case_id
status
```

Prinsip Dataset

- gunakan data lab;
- jangan gunakan data pribadi;
- jangan gunakan data sensitif;
- beri timestamp konsisten;
- simpan raw dan clean data;
- catat sumber data;
- pisahkan data simulasi dan data nyata.

10.7 Skenario Uji Proyek

Skenario tidak perlu banyak. Lebih baik sedikit tetapi bisa dijelaskan dengan bukti.

Skenario 1 — SSH Brute Force

Tujuan:

- memicu alert login gagal;
- membaca alert Wazuh;
- membuat query OpenSearch;
- membuat case;
- menjalankan playbook.

Bukti:

- alert Wazuh;
- source IP;
- rule ID;
- timestamp;
- jumlah percobaan;
- query hasil login gagal;
- case DFIR-IRIS;
- audit log Shuffle.

Skenario 2 — Port Scan

Tujuan:

- melihat aktivitas scanning di jaringan lab;
- membaca alert Suricata;
- membaca Zeek `conn.log`;
- membuat catatan investigasi.

Bukti:

- alert Suricata;
- source IP;
- destination IP;
- port;
- Zeek connection log;
- timeline investigasi.

Skenario 3 — Perubahan File Konfigurasi

Tujuan:

- menguji FIM;
- memonitor file penting;
- membaca alert perubahan file;
- membuat rekomendasi hardening.

Bukti:

- alert FIM;
- path file;
- waktu perubahan;
- host;
- user jika tersedia;
- analyst note.

Skenario 4 — Web Attack Sederhana di Lab

Tujuan:

- menguji deteksi trafik aplikasi web;
- membaca alert jaringan;
- menghubungkan alert jaringan dengan log server.

Gunakan aplikasi lab yang sah, lokal, dan tidak terhubung ke internet.

Bukti:

- alert Suricata;
- log aplikasi lab;
- Zeek HTTP log;
- query OpenSearch;
- case note.

Skenario 5 — Aktivitas Script Mencurigakan di Endpoint Linux

Karena proyek ini fokus open source dan tidak memakai ekosistem proprietary, skenario endpoint diarahkan ke Linux.

Tujuan:

- mendeteksi eksekusi script tidak biasa;
- membaca log proses atau osquery;
- menghubungkan aktivitas dengan Wazuh alert.

Bukti:

- alert endpoint;
- nama proses;
- command line yang disanitasi;
- path file;
- timestamp;
- rekomendasi hardening.

10.8 Deteksi Minimal yang Harus Dibuat

Mahasiswa harus membuat minimal 3 deteksi.

Deteksi 1 — Login Gagal Berulang

Kriteria:

```
event_type = authentication_failed
count > 10
time_window = 5 menit
```

Output:

- alert;
- dashboard;
- case jika valid;
- playbook triage.

Deteksi 2 — Perubahan File Penting

Kriteria:

```
file_path = /etc/ssh/sshd_config
event_type = file_modified
```

Output:

- alert FIM;
- evidence;
- rekomendasi validasi konfigurasi.

Deteksi 3 — Port Scan

Kriteria:

```
source_ip sama
destination_ip sama
banyak destination_port
time_window pendek
```

Output:

- alert Suricata;
- Zeek conn.log;
- catatan investigasi.

Deteksi Tambahan Opsional

- domain mencurigakan;
- akses web tidak wajar;
- vulnerability severity tinggi;
- file dummy cocok dengan YARA;
- IOC match dari MISP.

10.9 Dashboard Minimum

Dashboard bukan dekorasi. Dashboard harus menjawab pertanyaan SOC.

Dashboard minimum:

- alert per waktu;
- alert per host;
- alert per severity;
- top source IP;
- top rule ID;
- alert per MITRE tactic;
- case open vs closed;
- MTTD dan MTTR sederhana jika tersedia.

OpenSearch Dashboards dapat dipakai untuk membuat visualisasi dan dashboard berbasis data OpenSearch. (OpenSearch Documentation)

Contoh Panel

Panel 1 — Alert per Host

Menjawab:

- host mana paling banyak alert?

Panel 2 — Alert per Severity

Menjawab:

- berapa banyak alert high dan medium?

Panel 3 — Top Source IP

Menjawab:

- IP mana paling sering muncul?

Panel 4 — Top Rule ID

Menjawab:

- rule mana paling bising?

Panel 5 — Alert per MITRE Tactic

Menjawab:

- taktik adversary apa yang paling sering terlihat?

10.10 Case Incident Minimum

Mahasiswa harus membuat minimal 1 case incident.

Isi Case

- judul;
- waktu deteksi;
- sumber alert;
- severity;
- confidence;
- aset terdampak;
- observables;
- evidence;
- timeline;
- tasks;
- analyst notes;
- tindakan;
- rekomendasi;
- status akhir.

Contoh Case

Case ID:
SOC-FINAL-001

Judul:
Login Gagal Berulang pada Endpoint Linux

Sumber:
Wazuh

Severity:
Medium

Status:
Closed — Lab Simulation Validated

Ringkasan:
Wazuh mendeteksi login gagal berulang dari satu source IP ke endpoint Linux dalam jaringan lab. Hasil query OpenSearch menunjukkan lebih dari 10 event dalam 5 menit. Tidak ditemukan login berhasil setelah percobaan gagal.

Rekomendasi:
Batasi akses layanan, perkuat autentikasi, aktifkan monitoring FIM, dan gunakan alert threshold.

DFIR-IRIS cocok untuk bagian ini karena mendukung investigasi kolaboratif, dokumentasi teknis, dan manajemen case incident response. (DFIR-IRIS)

10.11 Playbook Automation Minimum

Mahasiswa harus membuat minimal 1 playbook.

Playbook: “SSH Brute Force Detected”

Alur:

```
Webhook menerima alert
    ↓
Parsing JSON
    ↓
Cek severity
    ↓
Ambil source_ip
    ↓
Cek IOC di MISP
    ↓
Buat case
    ↓
Tambahkan observable
    ↓
Kirim notifikasi
    ↓
Tunggu approval
    ↓
Jika approved, blokir sementara di lab
    ↓
Tulis audit log
```

Shuffle dapat dipakai untuk membuat workflow otomasi keamanan berbasis webhook dan integrasi API. (Shuffler)

Syarat Aman

- tidak ada blokir otomatis tanpa approval;
- semua input divalidasi;
- semua action dicatat;
- token tidak disimpan di repository;
- workflow diuji dengan data dummy;
- hasil automation disimpan sebagai evidence.

10.12 KPI SOC Sederhana

KPI tidak perlu rumit. Untuk proyek mahasiswa, cukup gunakan metrik sederhana.

1. MTTD — Mean Time To Detect

Waktu dari simulasi dimulai sampai alert muncul.

Contoh:

Simulasi dimulai: 10:00
Alert muncul: 10:03
MTTD: 3 menit

2. MTTR — Mean Time To Respond

Waktu dari alert muncul sampai respons awal dilakukan.

Contoh:

Alert muncul: 10:03
Case dibuat: 10:10
MTTR awal: 7 menit

3. Jumlah Alert

Catat total alert per skenario.

Contoh:

brute force: 12 alert;
port scan: 5 alert;
FIM: 2 alert.

4. False Positive

Catat alert yang tidak terbukti relevan.

Contoh:

Total alert: 20
False positive: 4
False positive rate: 20%

5. Severity

Catat distribusi:

- low;
- medium;
- high;
- critical.

6. Incident Closed

Catat case yang berhasil ditutup.

Contoh:

> Case dibuat: 1
> Case closed: 1
> Closure rate: 100%

Rujukan SOC menekankan pentingnya metrik seperti efektivitas incident response, beban kerja analis, false positive, dan pengukuran performa SOC untuk perbaikan berkelanjutan.

10.13 Dokumentasi Proyek

Dokumentasi harus rapi. Ini bagian penting dari portofolio.

Struktur Folder

```
mini-soc-final-project/
├── README.md
├── architecture/
├── configs/
├── dashboards/
├── detections/
├── sample-logs/
├── cases/
├── playbooks/
├── reports/
├── screenshots/
└── sanitized-data/
```

Isi Minimal README

Judul Proyek:
Mini SOC Berbasis Open Source

Tujuan:
Membangun SOC lab untuk deteksi, investigasi, case management, dan automation.

Tools:
Wazuh, Suricata/Zeek, OpenSearch, MISP, DFIR-IRIS, Shuffle, Linux, Docker, Python.

Arsitektur:
Jelaskan diagram singkat.

Skenario:
1. SSH brute force.
2. Port scan.
3. Perubahan file konfigurasi.

Output:
- 3 deteksi.
- 1 dashboard.
- 1 case.
- 1 playbook.
- 1 laporan.

Dokumentasi yang Wajib Ada

- diagram arsitektur;
- daftar aset;
- daftar IP lab;
- daftar tools dan versi;
- daftar rule;
- sample alert JSON;
- screenshot dashboard;
- case report;
- playbook flow;
- KPI;
- lesson learned.

10.14 Publikasi Aman ke GitHub

Publikasi boleh dilakukan, tetapi harus aman.

Yang Boleh Dipublikasikan

- diagram arsitektur umum;
- konfigurasi dummy;
- sample log dummy;
- rule lab;
- template case;
- template report;
- screenshot yang sudah disanitasi;
- README;
- playbook tanpa credential;
- dataset simulasi.

Yang Tidak Boleh Dipublikasikan

- password;
- token API;
- private key;
- log asli yang sensitif;
- IP publik organisasi;
- username nyata;
- domain internal nyata;
- data pribadi;
- screenshot dashboard berisi data sensitif;
- file konfigurasi dengan secret.

Checklist Sebelum Publish

- Semua credential dihapus.
- Semua token dihapus.
- IP sensitif disamarkan.
- Username nyata disamarkan.
- Screenshot diperiksa.
- Data lab diberi label simulasi.
- README menjelaskan bahwa proyek ini lab tertutup.
- Tidak ada instruksi menyerang sistem publik.

Portofolio yang baik menunjukkan kemampuan tanpa membocorkan rahasia.

10.15 Etika dan Legalitas

Proyek SOC harus legal dan etis.

Aturan utama:

- gunakan lab sendiri;
- jangan menyerang sistem publik;
- jangan memindai jaringan tanpa izin;
- jangan memakai data pribadi;
- jangan membagikan IOC yang belum tervalidasi;
- jangan membuat klaim breach tanpa bukti;
- jangan mempublikasikan log sensitif;
- jangan menjalankan malware nyata;
- jangan membuka dashboard SOC ke internet.

Etika SOC:

> **mendeteksi untuk melindungi, bukan mengeksploitasi untuk pamer.**

10.16 Roadmap Karier SOC Analyst

Proyek akhir ini dapat menjadi awal karier. Mahasiswa bisa memakai proyek ini untuk menunjukkan kemampuan nyata.

Kompetensi Teknis

- Linux;
- networking;
- log analysis;
- Wazuh;
- Suricata;
- Zeek;
- OpenSearch;
- MISP;
- DFIR-IRIS;
- Shuffle;
- Python;
- JSON;
- Docker;
- MITRE ATT&CK.

Kompetensi Analitis

- triage alert;
- validasi evidence;
- incident response;
- timeline;
- case management;
- threat intelligence;
- false positive analysis;
- risk-based prioritization.

Kompetensi Komunikasi

- menulis laporan;
- membuat ringkasan eksekutif;
- presentasi 10 menit;
- menjelaskan teknis secara sederhana;
- menulis lesson learned;
- menyampaikan rekomendasi.

Rujukan karier SOC menekankan bahwa analis SOC membutuhkan dasar networking, endpoint security, SIEM, IDS/IPS, incident response plan, MITRE ATT&CK, Cyber Kill Chain, dan kemampuan menjelaskan alert dengan bukti.

10.17 Hands-on Proyek — Checklist Implementasi

Gunakan checklist berikut sebagai panduan eksekusi.

Tahap 1 — Infrastruktur

- Siapkan lab Linux.
- Siapkan jaringan virtual/lokal.
- Buat snapshot awal.
- Buat struktur folder proyek.
- Buat inventaris aset.

Tahap 2 — Deploy Komponen SOC

- Deploy Wazuh.
- Deploy OpenSearch/Dashboard.
- Deploy Suricata/Zeek atau Security Onion.
- Deploy MISP.
- Deploy DFIR-IRIS.
- Deploy Shuffle.

Minimal wajib: **3 komponen berjalan baik**.

Tahap 3 — Log Collection

- Endpoint Linux mengirim log.
- Server lab mengirim log.
- Sensor jaringan membaca trafik.
- Sample alert masuk ke OpenSearch.
- Timestamp konsisten.

Minimal wajib: **2 sumber log aktif**.

Tahap 4 — Deteksi

- Rule login gagal dibuat.
- Rule FIM aktif.
- Deteksi port scan tersedia.
- Alert muncul.
- Alert bisa diekspor JSON.

Minimal wajib: **3 deteksi terbukti berjalan**.

Tahap 5 — Dashboard

- Alert per host.
- Alert per severity.
- Top source IP.
- Top rule ID.
- Alert per MITRE tactic.

Minimal wajib: **1 dashboard SOC sederhana**.

Tahap 6 — Case Management

- Case dibuat.
- Observable ditambahkan.
- Evidence dicatat.
- Timeline dibuat.
- Task dibuat.
- Case ditutup.

Minimal wajib: **1 case incident lengkap**.

Tahap 7 — Automation

- Webhook dibuat.
- Alert dummy diterima.
- JSON diparse.
- IOC enrichment dilakukan.
- Case dibuat otomatis atau semi-otomatis.
- Approval sebelum respons berisiko.
- Audit log tersimpan.

Minimal wajib: **1 playbook automation aman**.

Tahap 8 — Laporan dan Presentasi

- Laporan 5–10 halaman selesai.
- Screenshot disanitasi.
- KPI dihitung.
- Lesson learned ditulis.
- Slide/paparan 10 menit dibuat.
- Repository disanitasi.

10.18 Format Laporan Akhir 5–10 Halaman

Gunakan format berikut.

Judul:
Mini SOC Berbasis Open Source untuk Deteksi dan Respons Insiden Lab

1. Ringkasan Eksekutif
Tulis 1 paragraf: tujuan, tools, skenario, hasil utama.

2. Arsitektur
Jelaskan komponen, alur log, dan diagram.

3. Tools yang Digunakan
Wazuh, OpenSearch, Suricata/Zeek, MISP, DFIR-IRIS, Shuffle, AI lokal.

4. Dataset dan Sumber Log
Jelaskan endpoint, sensor, alert JSON, dan data lab.

5. Skenario Uji
Brute force, port scan, perubahan file, web attack lab, script mencurigakan Linux.

6. Deteksi dan Dashboard
Jelaskan rule, query, dan visualisasi.

7. Incident Response dan Case
Jelaskan case, observables, evidence, timeline, dan status.

8. Automation
Jelaskan playbook, webhook, approval, dan audit log.

9. KPI SOC
MTTD, MTTR, jumlah alert, false positive, severity, incident closed.

10. Lesson Learned dan Rekomendasi
Apa yang berhasil, gagal, dan perlu diperbaiki.

10.19 Format Presentasi 10 Menit

Struktur presentasi:

Menit 1 — Masalah

- Mengapa SOC diperlukan?
- Apa tujuan proyek?

Menit 2 — Arsitektur

- Tampilkan diagram Mini SOC.
- Jelaskan alur log.

Menit 3 — Data

- Sumber log.
- Format JSON.
- Normalisasi field.

Menit 4 — Deteksi

- Tunjukkan 3 rule/query.
- Jelaskan alert yang muncul.

Menit 5 — Dashboard

- Tampilkan dashboard.
- Jelaskan insight utama.

Menit 6 — Investigasi

- Tunjukkan case.
- Jelaskan observables dan evidence.

Menit 7 — Respons

- Tunjukkan playbook automation.
- Jelaskan approval dan audit log.

Menit 8 — KPI

- MTTD.
- MTTR.
- jumlah alert.
- false positive.
- case closed.

Menit 9 — AI Open Source

- Tunjukkan ringkasan alert atau draft laporan.
- Jelaskan batasan AI.

Menit 10 — Lesson Learned

- Apa yang dipelajari?
- Apa pengembangan berikutnya?

10.20 Rubrik Penilaian Proyek

Komponen	Bobot
Arsitektur Mini SOC jelas	15%
Minimal 3 komponen SOC berjalan	15%
Log dari minimal 2 sumber	10%
Minimal 3 deteksi terbukti	15%
Dashboard dan query investigasi	10%
Case incident lengkap	15%
Playbook automation dengan audit log	10%
Laporan dan presentasi	10%
Total	**100%**

Nilai Tambahan

Mahasiswa mendapat nilai tambahan jika:

- memakai MISP dengan IOC JSON;
- membuat mapping MITRE ATT&CK;
- menghitung KPI dengan jelas;
- memakai AI lokal secara aman;
- repository rapi dan disanitasi;
- dokumentasi mudah diikuti orang lain.

10.21 Kesalahan Umum Proyek Akhir

Kesalahan yang sering terjadi:

- terlalu fokus instalasi, lupa analisis;
- memasang banyak tools tetapi tidak ada data;
- dashboard ramai tetapi tidak menjawab pertanyaan;
- alert muncul tetapi tidak ada evidence;
- case dibuat tetapi tidak ada timeline;
- automation berjalan tetapi tidak ada audit log;
- blokir otomatis tanpa approval;
- laporan panjang tetapi tidak tajam;
- repository berisi credential;
- AI dipakai untuk membuat kesimpulan tanpa log.

Proyek SOC gagal bukan karena tools kurang, tetapi karena alur bukti tidak jelas.

10.22 Peran AI Open Source dalam Proyek Akhir

AI open source dapat dimasukkan sebagai komponen pendukung, bukan inti pengambil keputusan.

Gunakan AI untuk:

- meringkas alert JSON;
- menjelaskan field log;
- membuat draft laporan;
- menyusun lesson learned;
- membuat daftar pertanyaan triage;
- membuat ringkasan presentasi;
- membantu mapping MITRE;
- membantu menulis rekomendasi awal.

Contoh prompt aman:

Anda adalah asisten SOC analyst.
Analisis data alert berikut dalam konteks lab tertutup.
Tugas:
1. ringkas kejadian,
2. sebutkan evidence,
3. sebutkan kemungkinan false positive,
4. sarankan pertanyaan triage,
5. buat draft rekomendasi.

Jangan menyimpulkan incident jika bukti belum cukup.
Jangan membuat klaim di luar data.

Aturan:

- data harus disanitasi;
- jangan masukkan token;
- jangan masukkan password;
- jangan masukkan data pribadi;
- semua output AI harus divalidasi dengan log.

AI membuat laporan lebih cepat. Bukti membuat laporan sah.

10.23 Rangkuman Bab

Bab ini menggabungkan seluruh isi buku menjadi proyek akhir Mini SOC berbasis open source. Mahasiswa tidak hanya belajar konsep, tetapi membangun sistem yang dapat didemokan: log masuk, alert muncul, query berjalan, dashboard terbaca, case dibuat, IOC dipakai, playbook berjalan, KPI dihitung, dan laporan ditulis.

Proyek akhir ini menegaskan bahwa SOC bukan hanya tools. SOC adalah alur kerja:

data → deteksi → investigasi → respons → dokumentasi → pembelajaran.

Dengan proyek ini, mahasiswa memiliki portofolio nyata untuk menunjukkan kemampuan dasar sebagai calon SOC analyst, security engineer, atau incident responder.

10.24 Output Bab

Setelah menyelesaikan Bab 10, pembaca diharapkan memiliki:

- Mini SOC berbasis open source;
- minimal 3 komponen SOC berjalan;
- log dari minimal 2 sumber;
- minimal 3 rule/query deteksi;
- minimal 1 dashboard SOC;
- minimal 1 case incident;
- minimal 1 playbook automation;
- laporan akhir 5–10 halaman;
- presentasi 10 menit;
- repository proyek yang aman dan disanitasi;
- pemahaman etika dan legalitas;
- roadmap awal menuju karier SOC analyst.

Pesan akhir Bab 10:
Bangun SOC kecil yang benar, bukan SOC besar yang rapuh; dari lab yang rapi lahir analis yang tajam, kritis, dan siap menjaga sistem nyata.

Lampiran A — Kebutuhan Hardware dan Software Lab

Lampiran ini menjadi acuan minimum untuk membangun **SOC lab berbasis open source**. Fokusnya bukan membuat infrastruktur besar, tetapi membuat lab yang **aman, cukup kuat, terdokumentasi, dan bisa diulang**.

Rujukan utama mengikuti arah buku terlampir: lab menggunakan **Linux, VM, Docker, Wazuh, Security Onion, OpenSearch, Suricata, Zeek, MISP, DFIR-IRIS, Shuffle, dan Python** sebagai fondasi praktik SOC open source.

A.1 Spesifikasi Minimum Laptop

Spesifikasi minimum untuk mahasiswa:

Komponen	Minimum
CPU	4 core
RAM	16 GB
Storage kosong	150 GB
Disk	SSD sangat disarankan
Network	Wi-Fi/LAN standar
OS host	Linux
Virtualisasi	KVM/libvirt atau Proxmox VE
Internet	Dibutuhkan untuk instalasi paket awal

Catatan:

- **16 GB RAM cukup untuk lab kecil**, tetapi tidak nyaman jika semua tools dijalankan bersamaan.
- Gunakan **SSD**, karena log indexing dan VM sangat berat di disk lambat.
- Mulai dari **Wazuh + 1 endpoint + OpenSearch**, lalu tambah komponen lain bertahap.

A.2 Spesifikasi Ideal Lab

Spesifikasi ideal untuk mahasiswa serius atau lab kampus:

Komponen	Ideal
CPU	8 core atau lebih
RAM	32–64 GB
Storage	500 GB–1 TB SSD
Network	1–2 interface jaringan
Platform	Proxmox VE atau KVM/libvirt
Backup	Storage eksternal / server backup
Segmentasi	Jaringan lab terpisah
Dokumentasi	Repository lokal + laporan praktikum

Catatan:

- Untuk Security Onion, sensor jaringan, dan OpenSearch, **RAM dan storage sangat menentukan kenyamanan lab**.
- Security Onion secara resmi diposisikan sebagai platform terbuka untuk *network visibility*, *host visibility*, *intrusion detection*, *log management*, dan *case management*. (Dokumentasi Security Onion)

A.3 Daftar VM

Desain minimum:

VM	Fungsi	RAM	Storage	Catatan
`soc-server`	Wazuh / OpenSearch	8–12 GB	80–150 GB	Pusat SIEM/XDR
`linux-endpoint-01`	Endpoint yang dimonitor	2–4 GB	20–40 GB	Mengirim log ke Wazuh
`lab-target-01`	Target uji legal	2–4 GB	20–40 GB	Untuk simulasi lab
`network-sensor-01`	Suricata/Zeek/Security Onion	8–16 GB	100–200 GB	Sensor trafik
`case-ti-soar`	MISP, DFIR-IRIS, Shuffle	8–16 GB	100–200 GB	Bisa digabung jika resource terbatas

Desain hemat:

- Gabungkan **MISP + DFIR-IRIS + Shuffle** pada satu VM.
- Jalankan **Suricata/Zeek** tanpa Security Onion jika resource terbatas.
- Gunakan dataset kecil agar OpenSearch tidak cepat memenuhi storage.

A.4 Estimasi RAM dan Storage

Komponen	RAM Minimum	Storage Minimum
Wazuh single-node	8 GB	80 GB
OpenSearch lab	4–8 GB	50–100 GB
Security Onion lab	12–16 GB	150 GB
Suricata standalone	2–4 GB	20–50 GB
Zeek standalone	2–4 GB	20–50 GB
MISP	4–8 GB	50 GB
DFIR-IRIS	4–8 GB	50 GB
Shuffle	4–8 GB	40–80 GB
Endpoint Linux	2 GB	20 GB

Prinsip:

- **Lebih baik sedikit tools tetapi stabil**, daripada banyak tools tetapi lambat dan tidak bisa dianalisis.
- Untuk lab 100 halaman / 10 bab, target realistis adalah **3–5 komponen aktif**, bukan semua komponen berjalan penuh setiap saat.

A.5 Rekomendasi OS

Rekomendasi utama:

- **Ubuntu Server LTS** untuk VM server.
- **Debian / Ubuntu Server** untuk endpoint lab.
- **Security Onion ISO** untuk sensor jaringan jika resource mencukupi.
- **Proxmox VE** untuk lab kampus.
- **KVM/libvirt** untuk laptop Linux.

Ubuntu cocok untuk lab karena merupakan sistem operasi open source berbasis Linux yang banyak dipakai pada server, cloud, dan infrastruktur. (Ubuntu) Docker Engine juga memiliki dokumentasi instalasi resmi untuk Ubuntu LTS modern. (Docker Documentation)

A.6 Tools Utama

Tool	Fungsi dalam Lab
Ubuntu Server	OS server lab
Docker	Menjalankan layanan berbasis container
Wazuh	SIEM/XDR endpoint, log, alert, FIM, vulnerability
Security Onion	Network visibility dan sensor SOC
OpenSearch	Indexing, query, dashboard, analytics
Suricata	Deteksi jaringan berbasis rule dan Eve JSON
Zeek	Metadata jaringan, conn.log, dns.log, http.log
MISP	Threat intelligence dan IOC sharing
DFIR-IRIS	Incident response dan case management
Shuffle	SOAR dan workflow automation
Python	Script API, parsing JSON, otomasi ringan

Wazuh adalah platform keamanan **free and open source** yang menyatukan kemampuan SIEM dan XDR. (Wazuh) OpenSearch adalah suite open source berlisensi Apache 2.0 untuk *ingest*, *search*, *visualize*, dan *analyze data*. (OpenSearch) MISP adalah solusi open source untuk mengumpulkan, menyimpan, mendistribusikan, dan berbagi indikator serta ancaman siber. (MISP Threat Intelligence) DFIR-IRIS adalah platform *free and open-source* untuk kolaborasi incident response. (DFIR-IRIS) Shuffle adalah platform otomasi open source yang dibangun untuk profesional keamanan. (GitHub)

Lampiran B — Cheatsheet Command Dasar SOC

Catatan penting: command berikut untuk **lab tertutup dan legal**. Jangan dipakai untuk mengakses, memindai, atau menguji sistem publik tanpa izin.

B.1 Command Linux Dasar

```
pwd
ls -lah
cd /path/folder
mkdir -p soc-lab/{configs,logs,reports,rules}
cp file1 file2
mv file1 file-baru
rm file
nano file.txt
```

Fungsi singkat:

- `pwd` melihat lokasi folder aktif.
- `ls -lah` melihat isi folder dengan detail.
- `cd` pindah folder.
- `mkdir -p` membuat folder bertingkat.
- `cp` menyalin file.
- `mv` memindah atau mengganti nama file.
- `rm` menghapus file.
- `nano` membuka editor teks terminal.

B.2 Docker dan Docker Compose

```
docker --version
docker compose version
docker ps
docker ps -a
docker images
docker compose up -d
docker compose down
docker compose logs -f
docker restart nama-container
```

Fungsi singkat:

- `docker --version` melihat versi Docker.
- `docker compose version` melihat versi Compose.
- `docker ps` melihat container aktif.
- `docker ps -a` melihat semua container.
- `docker images` melihat image.
- `docker compose up -d` menjalankan layanan di background.
- `docker compose down` menghentikan layanan.
- `docker compose logs -f` melihat log berjalan.
- `docker restart` memulai ulang container.

B.3 Cek Service

```
sudo systemctl status wazuh-manager
sudo systemctl status wazuh-agent
sudo systemctl status docker
sudo systemctl restart wazuh-agent
sudo systemctl restart wazuh-manager
sudo systemctl enable docker
```

Fungsi singkat:

- `systemctl status` melihat status service.
- `restart` memulai ulang service.
- `enable` membuat service aktif otomatis saat boot.
- `sudo` menjalankan perintah dengan hak admin.

B.4 Cek Log

```
sudo journalctl -u wazuh-agent -f
sudo journalctl -u wazuh-manager -f
tail -f /var/log/syslog
tail -n 100 /var/log/auth.log
grep "Failed password" /var/log/auth.log
```

Fungsi singkat:

- `journalctl -u` membaca log service.
- `-f` mengikuti log secara real-time.
- `tail -f` membaca log yang terus bertambah.
- `tail -n 100` menampilkan 100 baris terakhir.
- `grep` mencari kata tertentu.

B.5 Cek Port

```
ss -tulpen
sudo lsof -i -P -n
```

Fungsi singkat:

- `ss -tulpen` melihat port TCP/UDP yang aktif.
- `lsof -i` melihat proses yang memakai koneksi jaringan.
- `-P -n` mempercepat output dengan tidak melakukan resolusi nama.

B.6 Cek Koneksi

```
ip addr
ip route
ping 10.10.10.10
curl -k https://10.10.10.10
traceroute 10.10.10.10
```

Fungsi singkat:

- `ip addr` melihat alamat IP.
- `ip route` melihat routing.
- `ping` menguji konektivitas dasar.
- `curl` menguji HTTP/HTTPS/API.
- `traceroute` melihat jalur koneksi.

B.7 Cek Resource

```
free -h
df -h
du -sh *
top
htop
uptime
```

Fungsi singkat:

- `free -h` melihat RAM.
- `df -h` melihat disk.
- `du -sh *` melihat ukuran folder/file.
- `top` melihat proses aktif.
- `htop` tampilan proses lebih interaktif.
- `uptime` melihat lama sistem aktif dan load.

B.8 Cek File Berubah

```
stat /etc/ssh/sshd_config
sha256sum /etc/ssh/sshd_config
sudo find /etc -mtime -1
sudo find /var/www -type f -mtime -1
```

Fungsi singkat:

- `stat` melihat metadata file.
- `sha256sum` membuat hash file.
- `find -mtime -1` mencari file yang berubah dalam 1 hari terakhir.
- `-type f` membatasi hasil hanya file.

B.9 Query Sederhana curl

```
curl -k https://localhost:9200
curl -k -X GET "https://localhost:9200/_cat/indices?v"
curl -k -X GET
"https://localhost:9200/soc-alerts-*/_search?pretty"
```

Fungsi singkat:

- `curl` mengirim request HTTP/HTTPS.
- `-k` mengizinkan sertifikat lab/self-signed.
- `-X GET` memakai metode GET.
- `_cat/indices` melihat index OpenSearch.
- `_search` melakukan pencarian.
- `pretty` membuat output JSON lebih mudah dibaca.

B.10 Query API JSON

Contoh query OpenSearch:

```
curl -k -X GET
"https://localhost:9200/soc-alerts-*/_search?pretty" \
  -H "Content-Type: application/json" \
  -d '{
    "query": {
      "term": {
        "event_type": "authentication_failed"
      }
    }
  }'
```

Fungsi singkat:

- `-H "Content-Type: application/json"` menyatakan data dikirim dalam format JSON.
- `-d` mengirim body request.
- `term` mencari nilai field secara presisi.
- `event_type` adalah field.
- `authentication_failed` adalah nilai yang dicari.

Lampiran C — Template Alert JSON

Template ini dipakai agar alert dari Wazuh, Suricata, Zeek, OpenSearch, atau workflow SOAR dapat ditulis dalam format yang konsisten.

```
{
  "timestamp": "2026-01-01T10:00:00Z",
  "agent_name": "linux-endpoint-01",
  "agent_ip": "10.10.10.21",
  "rule_id": "100200",
  "rule_description": "Login gagal berulang pada endpoint
Linux",
  "severity": 8,
  "source_ip": "10.10.10.50",
  "destination_ip": "10.10.10.21",
  "username": "admin-test",
  "event_type": "authentication_failed",
  "mitre_tactic": "Credential Access",
  "mitre_technique": "Brute Force",
  "full_log": "Failed password for invalid user admin-test
from 10.10.10.50 port 44122 ssh2"
}
```

Penjelasan Field

Field	Fungsi
`timestamp`	Waktu kejadian
`agent_name`	Nama endpoint / agent
`agent_ip`	IP endpoint
`rule_id`	ID rule deteksi
`rule_description`	Deskripsi rule
`severity`	Tingkat prioritas alert
`source_ip`	IP sumber aktivitas
`destination_ip`	IP tujuan
`username`	User terkait event
`event_type`	Jenis event
`mitre_tactic`	Taktik MITRE ATT&CK
`mitre_technique`	Teknik MITRE ATT&CK
`full_log`	Potongan log asli sebagai bukti

MITRE ATT&CK adalah basis pengetahuan global tentang taktik dan teknik adversary berdasarkan observasi dunia nyata. (MITRE ATT&CK)

Lampiran D — Template Case Incident

Template ini dipakai untuk mengubah alert menjadi case investigasi yang rapi.

Case ID:
SOC-LAB-001

Judul Insiden:
Login gagal berulang pada endpoint Linux

Waktu Deteksi:
2026-01-01 10:04:00

Sumber Alert:
Wazuh

Aset Terdampak:
linux-endpoint-01 / 10.10.10.21

Severity:
Medium / High

Evidence:
1. Alert JSON Wazuh.
2. Query OpenSearch.
3. Log endpoint.
4. Log jaringan.
5. Screenshot dashboard.
6. Timeline case.

Timeline:
10:00 - Percobaan login gagal pertama.
10:04 - Wazuh menghasilkan alert.
10:07 - Query OpenSearch dilakukan.
10:10 - Case dibuat.
10:15 - Analisis awal selesai.
10:20 - Case ditutup sebagai simulasi lab valid.

Analisis:
Alert menunjukkan login gagal berulang dari satu source IP ke endpoint Linux. Tidak ditemukan bukti login berhasil setelah percobaan gagal. Aktivitas sesuai skenario simulasi lab.

Respons:
1. Monitoring source IP.
2. Validasi log endpoint.
3. Validasi log jaringan.

4. Dokumentasi evidence.
5. Tidak dilakukan blokir otomatis tanpa approval.

Rekomendasi:
1. Gunakan autentikasi kuat.
2. Batasi akses layanan.
3. Aktifkan alert threshold.
4. Tambahkan dashboard login gagal.
5. Tinjau ulang rule agar false positive rendah.

Status Penutupan:
Closed — Lab Simulation Validated

Lesson Learned:
Alert harus divalidasi dengan evidence sebelum dinyatakan sebagai incident.

Checklist Case

- Judul jelas.
- Waktu deteksi ada.
- Sumber alert jelas.
- Aset terdampak tercatat.
- Severity dinilai.
- Evidence disimpan.
- Timeline dibuat.
- Analisis ditulis.
- Respons dicatat.
- Rekomendasi dibuat.
- Status penutupan jelas.

Lampiran E — Template Dashboard SOC

Dashboard SOC harus membantu analis menjawab pertanyaan cepat. Jangan membuat dashboard hanya agar terlihat ramai.

OpenSearch cocok untuk dashboard karena dirancang untuk *ingest*, *search*, *visualize*, dan *analyze data*. (OpenSearch)

E.1 Panel Dashboard Minimum

Panel	Pertanyaan yang Dijawab
Alert per hari	Apakah alert meningkat?
Top source IP	IP mana paling sering memicu alert?
Top destination IP	Aset mana paling sering dituju?
Top endpoint	Endpoint mana paling banyak alert?
Alert per severity	Berapa alert high/medium/low?
Alert per rule ID	Rule mana paling bising?
Alert per MITRE tactic	Taktik adversary apa yang dominan?
Alert brute force	Apakah ada percobaan login berulang?
Alert malware	Apakah ada indikator file mencurigakan?
Alert FIM	File penting apa yang berubah?

E.2 Field yang Disarankan

```
timestamp
agent_name
agent_ip
source_ip
destination_ip
rule_id
rule_description
severity
event_type
mitre_tactic
mitre_technique
```

E.3 Contoh Layout Dashboard

```
[Alert per Hari]          [Alert per Severity]

[Top Source IP]           [Top Destination IP]

[Top Endpoint]            [Top Rule ID]

[Alert per MITRE Tactic]  [Alert Brute Force]

[Alert Malware]           [Alert FIM]
```

E.4 Query Dasar Dashboard

Alert per Severity

```
{
  "aggs": {
    "alert_per_severity": {
      "terms": {
        "field": "severity"
      }
    }
  }
}
```

Top Source IP

```
{
  "aggs": {
    "top_source_ip": {
      "terms": {
        "field": "source_ip"
      }
    }
  }
}
```

Alert Brute Force

```
{
  "query": {
    "term": {
      "event_type": "authentication_failed"
    }
  }
}
```

Alert FIM

```
{
  "query": {
    "term": {
      "event_type": "file_modified"
    }
  }
}
```

E.5 Prinsip Dashboard yang Baik

- **Sederhana.**
- **Bisa dijelaskan.**
- **Berbasis field yang konsisten.**
- **Memakai filter waktu.**
- **Membantu triage.**
- **Tidak menampilkan data sensitif berlebihan.**
- **Bisa dipakai untuk laporan.**

Lampiran F — 30 Prompt AI untuk SOC Analyst

Catatan penting: prompt AI hanya dipakai sebagai **asisten analisis**, bukan pengganti bukti teknis. Semua jawaban AI harus divalidasi dengan log, alert JSON, dashboard, case, dan evidence.

Gunakan AI open source/lokal jika memungkinkan. Jangan masukkan password, token API, data pribadi, log sensitif, atau informasi rahasia.

F.1 Ringkasan Alert

Prompt 1 — Ringkasan Alert Singkat

> Anda adalah asisten SOC analyst.
> Ringkas alert JSON berikut dalam 5 kalimat.
> Fokus pada waktu, host, source IP, event type, severity, dan bukti utama.
> Jangan membuat kesimpulan di luar data.
>
> [masukkan alert JSON]

Prompt 2 — Ringkasan untuk Mahasiswa

> Jelaskan alert berikut dengan bahasa sederhana untuk mahasiswa cybersecurity.
> Sebutkan apa yang terjadi, mengapa penting, dan data apa yang masih kurang.
>
> [masukkan alert JSON]

Prompt 3 — Ringkasan untuk Dosen/Pembimbing

> Buat ringkasan profesional dari alert berikut untuk laporan praktikum.
> Gunakan format: kejadian, aset terdampak, bukti, status awal, rekomendasi triage.
>
> [masukkan alert JSON]

F.2 Triage Insiden

Prompt 4 — Triage Awal

> Lakukan triage awal terhadap alert berikut.
> Nilai severity, confidence, asset criticality, dan impact.
> Jika data belum cukup, tulis "data belum cukup".
>
> [masukkan alert JSON]

Prompt 5 — Event, Alert, atau Incident

> Klasifikasikan data berikut: apakah masih event, alert, atau sudah layak menjadi incident?
> Jelaskan alasan teknisnya berdasarkan evidence yang tersedia.
>
> [masukkan data alert/case]

Prompt 6 — Pertanyaan Triage Lanjutan

> Buat 10 pertanyaan triage yang perlu dijawab analis sebelum alert ini dijadikan case incident.
>
> [masukkan alert JSON]

F.3 Analisis Log

Prompt 7 — Analisis Log Mentah

> Analisis potongan log berikut.
> Ekstrak timestamp, host, username, source IP, destination IP, event type, dan indikasi risiko.
> Jangan menebak field yang tidak ada.
>
> [masukkan log]

Prompt 8 — Normalisasi Field

> Ubah log berikut menjadi JSON SOC dengan field:
> timestamp, agent_name, source_ip, destination_ip, username, event_type, severity, full_log.
>
> [masukkan log]

Prompt 9 — Korelasi Log

Bandingkan beberapa alert berikut.
Cari pola waktu, source IP, host, rule ID, dan event type.
Buat kesimpulan hanya jika didukung data.

[masukkan daftar alert]

F.4 Mapping MITRE ATT&CK

Prompt 10 — Mapping MITRE Dasar

Mapping alert berikut ke MITRE ATT&CK.
Sebutkan tactic, technique, alasan mapping, dan tingkat keyakinan.
Jika tidak cukup data, nyatakan secara eksplisit.

[masukkan alert JSON]

Prompt 11 — Validasi Mapping MITRE

Tinjau mapping MITRE berikut.
Apakah mapping sudah tepat?
Berikan koreksi jika ada, tetapi jangan menambahkan klaim tanpa bukti.

[masukkan alert + mapping]

Prompt 12 — Penjelasan MITRE untuk Laporan

Jelaskan mapping MITRE berikut dalam bahasa Indonesia yang mudah dipahami mahasiswa.
Gunakan narasi singkat dan tajam.

[masukkan mapping MITRE]

F.5 Pembuatan Laporan Manajemen

Prompt 13 — Ringkasan Eksekutif

> Buat ringkasan eksekutif 1 paragraf dari case incident berikut.
> Fokus pada dampak, bukti, status respons, dan rekomendasi.
> Jangan menyebut hal yang tidak ada di evidence.
>
> [masukkan case]

Prompt 14 — BLUF untuk Pimpinan

> Buat laporan singkat dengan metode BLUF:
> Bottom Line, Evidence, Impact, Action, Next Step.
>
> [masukkan case incident]

Prompt 15 — Rekomendasi Strategis

> Buat rekomendasi strategis dari incident berikut.
> Pisahkan menjadi: quick win, perbaikan teknis, perbaikan proses, dan pembelajaran SOC.
>
> [masukkan case]

F.6 Pembuatan Timeline

Prompt 16 — Timeline dari Alert

> Susun timeline kronologis dari data alert berikut.
> Format: waktu - kejadian - sumber evidence.
> Jangan mengisi waktu yang tidak tersedia.
>
> [masukkan daftar alert]

Prompt 17 — Timeline Incident Response

> Buat timeline incident response dari case berikut:
> deteksi, triage, validasi, containment, recovery, closure, lesson learned.
>
> [masukkan case]

Prompt 18 — Gap Timeline

> Cari gap dalam timeline berikut.
> Sebutkan data apa yang hilang dan log apa yang perlu dicari.
>
> [masukkan timeline]

F.7 Validasi IOC

Prompt 19 — Ekstraksi IOC

> Ekstrak IOC dari teks berikut.
> Kelompokkan menjadi IP, domain, URL, hash, username, hostname, file path.
> Jangan menambahkan IOC yang tidak ada.
>
> [masukkan teks/log]

Prompt 20 — IOC ke JSON

> Ubah daftar IOC berikut menjadi JSON dengan field:
> type, value, source, confidence, context, recommended_action.
>
> [masukkan IOC]

Prompt 21 — Analisis IOC

> Analisis IOC berikut dalam konteks case lab.
> Bedakan IOC yang kuat, lemah, dan perlu validasi tambahan.
>
> [masukkan IOC JSON]

F.8 Pembuatan Playbook

Prompt 22 — Playbook Triage Alert

> Buat playbook SOC untuk alert berikut.
> Gunakan struktur: trigger, condition, enrichment, analyst approval, action, audit log, rollback.
>
> [masukkan alert type]

Prompt 23 — Playbook Brute Force

Buat playbook untuk deteksi login gagal berulang di lab SOC.
Pastikan ada approval sebelum tindakan blokir.
Gunakan tools open source: Wazuh, OpenSearch, DFIR-IRIS, MISP, Shuffle.

[masukkan konteks lab]

Prompt 24 — Review Playbook

Review playbook berikut.
Cari risiko false positive, kehilangan bukti, tindakan berlebihan, dan kekurangan audit log.

[masukkan playbook]

F.9 Deteksi False Positive

Prompt 25 — Evaluasi False Positive

Evaluasi apakah alert berikut kemungkinan false positive.
Gunakan evidence yang tersedia.
Sebutkan data tambahan yang diperlukan sebelum membuat keputusan.

[masukkan alert JSON]

Prompt 26 — Penyebab Noise

Analisis mengapa rule berikut menghasilkan terlalu banyak alert.
Berikan saran tuning tanpa melemahkan deteksi utama.

[masukkan rule/alert sample]

Prompt 27 — Tuning Rule

Buat rekomendasi tuning rule berdasarkan alert berikut.
Pisahkan rekomendasi menjadi threshold, field filter, asset context, dan time window.

[masukkan data alert]

F.10 Lesson Learned

Prompt 28 — Lesson Learned Case

> Buat lesson learned dari case berikut.
> Format: apa yang terjadi, apa yang berhasil, apa yang gagal, apa yang harus diperbaiki.
>
> [masukkan case incident]

Prompt 29 — Rekomendasi Perbaikan SOC

> Dari case berikut, buat rekomendasi perbaikan SOC untuk:
> people, process, technology, data, dan governance.
>
> [masukkan case]

Prompt 30 — Ringkasan Portofolio Mahasiswa

> Buat ringkasan portofolio proyek Mini SOC berbasis open source.
> Sebutkan tools, skenario, deteksi, dashboard, case, playbook, KPI, dan lesson learned.
> Gunakan bahasa profesional, singkat, dan siap dimasukkan ke README.
>
> [masukkan ringkasan proyek]

Catatan Akhir Lampiran

AI membantu mempercepat analisis, tetapi bukti teknis tetap sumber kebenaran.
Dalam SOC, keputusan harus selalu kembali ke log, alert JSON, timeline, case, IOC, query, dashboard, dan audit trail.

Glossary

Istilah	Definisi
Active Response	Mekanisme respons otomatis atau semi-otomatis ketika alert tertentu terpenuhi. Harus digunakan hati-hati agar tidak memblokir aset penting atau menghapus bukti.
Agent	Komponen kecil yang dipasang pada endpoint untuk mengirim log, event, dan telemetry ke server SOC seperti Wazuh.
AI Lokal	Model AI yang berjalan di infrastruktur sendiri. Dalam SOC, AI lokal dipakai untuk membantu ringkasan alert, analisis log, laporan, dan triage tanpa mengirim data sensitif ke layanan eksternal.
Alert	Sinyal keamanan yang muncul ketika event memenuhi rule, query, atau pola deteksi tertentu. Alert belum tentu incident.
Alert Fatigue	Kondisi ketika analis terlalu banyak menerima alert sehingga sulit membedakan sinyal penting dari noise.
Analyst Notes	Catatan analis selama investigasi incident. Berisi observasi, asumsi, validasi, keputusan sementara, dan rekomendasi.
API	*Application Programming Interface*, mekanisme agar tools SOC dapat saling berinteraksi secara terstruktur. Contohnya Wazuh API, OpenSearch API, MISP API, dan webhook SOAR.
Asset Criticality	Tingkat kepentingan sebuah aset terhadap organisasi atau lab. Alert pada aset kritikal biasanya memiliki prioritas lebih tinggi.
Attack Surface	Seluruh titik masuk yang dapat dimanfaatkan penyerang, seperti service terbuka, endpoint, aplikasi, API, akun, dan jaringan.
Audit Log	Catatan aktivitas sistem atau workflow. Dalam SOAR, audit log mencatat trigger, action, approval, hasil, dan error agar otomasi dapat dipertanggungjawabkan.
Automation	Proses menjalankan tugas berulang secara otomatis, misalnya parsing alert, enrichment IOC, membuat case, atau mengirim notifikasi.
Baseline	Gambaran kondisi normal sistem atau jaringan. Baseline membantu SOC membedakan aktivitas wajar dan anomali.
Brute Force	Teknik mencoba banyak kombinasi kredensial untuk memperoleh akses. Dalam lab SOC, sering dideteksi melalui login gagal berulang.
Case	Wadah investigasi incident. Case berisi alert, observables, evidence, timeline, task, analyst notes, respons, status, dan laporan akhir.

Case Management	Proses mengelola incident secara terstruktur dari awal sampai penutupan. Tool contoh: DFIR-IRIS.
Containment	Tahap incident response untuk membatasi dampak insiden agar tidak menyebar lebih luas. Contoh: isolasi host lab atau blokir sementara setelah approval.
Correlation	Proses menghubungkan beberapa event, alert, log, IOC, atau data jaringan agar menjadi konteks investigasi yang lebih utuh.
Cortex	Engine analyzer yang sering dikaitkan dengan workflow TheHive untuk menganalisis observables seperti IP, hash, domain, dan URL.
Cyber Kill Chain	Model tahapan serangan, mulai dari reconnaissance sampai actions on objectives. Dipakai untuk memahami alur serangan secara sederhana.
Dashboard SOC	Tampilan visual yang membantu analis membaca alert, severity, top source IP, top endpoint, rule ID, MITRE tactic, dan tren keamanan.
Data	Fakta mentah seperti IP, hash, timestamp, log, atau event. Data belum tentu bermakna sebelum dianalisis.
Decoder	Mekanisme untuk membaca log mentah dan mengekstrak field penting agar bisa diproses rule. Dalam Wazuh, decoder sangat penting untuk analisis log.
Detection Engineering	Disiplin merancang, menguji, memperbaiki, dan mendokumentasikan rule atau query deteksi agar akurat dan minim false positive.
DFIR	*Digital Forensics and Incident Response*, bidang yang menggabungkan investigasi forensik digital dan respons insiden.
DFIR-IRIS	Platform open source untuk incident response dan case management. Cocok untuk dokumentasi case, evidence, observables, timeline, dan laporan.
Docker	Platform container untuk menjalankan aplikasi dan layanan secara terisolasi. Banyak tools SOC dapat dijalankan melalui Docker Compose.
Docker Compose	Tool untuk menjalankan beberapa container sekaligus menggunakan file konfigurasi. Cocok untuk lab SOC karena mudah diulang.
Endpoint	Perangkat atau host yang dimonitor, misalnya server Linux, VM lab, atau mesin pengguna. Endpoint menjadi sumber log penting bagi SOC.
EDR	*Endpoint Detection and Response*, kemampuan deteksi dan respons pada endpoint. Dalam buku ini konsep EDR dipelajari melalui agent, log, FIM, osquery, dan telemetry endpoint.

Enrichment	Proses memperkaya alert dengan konteks tambahan, misalnya cek IP ke MISP, mapping MITRE, asset criticality, atau riwayat alert.
Event	Kejadian yang tercatat di sistem, aplikasi, endpoint, atau jaringan. Event belum tentu alert dan belum tentu incident.
Evidence	Bukti teknis dalam investigasi, seperti alert JSON, log mentah, hasil query, PCAP, hash, screenshot dashboard, dan timeline.
Eve JSON	Format output JSON dari Suricata yang berisi alert, flow, metadata, dan data jaringan lain. Penting untuk integrasi SIEM dan OpenSearch.
False Positive	Alert yang terlihat mencurigakan tetapi setelah divalidasi bukan ancaman nyata. False positive yang tinggi menyebabkan alert fatigue.
Feed	Sumber threat intelligence yang menyediakan IOC, domain, IP, hash, atau informasi ancaman lain. Di MISP, feed perlu dikurasi agar tidak menambah noise.
FIM	*File Integrity Monitoring*, kemampuan memantau perubahan file penting. Dalam Wazuh, FIM berguna untuk mendeteksi perubahan konfigurasi atau file sensitif.
Flow	Ringkasan koneksi jaringan, misalnya IP sumber, IP tujuan, port, protokol, durasi, dan jumlah byte. Zeek banyak dipakai untuk membaca flow.
Full Log	Potongan log asli yang menjadi sumber alert. Full log penting agar analis tidak hanya bergantung pada ringkasan dashboard.
Galaxy	Struktur konteks dalam MISP untuk mengaitkan event dengan threat actor, malware family, campaign, tool, atau framework tertentu.
Hash	Sidik jari digital file, misalnya SHA256. Dipakai sebagai IOC untuk mengidentifikasi file tertentu.
Host Visibility	Kemampuan melihat aktivitas pada host atau endpoint, seperti login, proses, file berubah, service, dan konfigurasi.
IDS	*Intrusion Detection System*, sistem yang mendeteksi aktivitas mencurigakan dan menghasilkan alert. Contoh: Suricata sebagai NIDS.
Incident	Kejadian keamanan yang sudah divalidasi, memiliki dampak atau potensi dampak, dan membutuhkan respons.
Incident Response	Proses menangani incident dari preparation, identification, containment, eradication, recovery, sampai lessons learned. SOC modern menggabungkan triage, investigasi, evidence, dan dokumentasi respons.

Indicator of Compromise	Indikator teknis yang dapat menunjukkan aktivitas berbahaya, seperti IP, domain, URL, hash, file path, atau user agent.
Information	Data yang sudah diberi konteks. Contoh: “IP tertentu melakukan login gagal 20 kali dalam 5 menit.”
Intelligence	Informasi yang sudah dianalisis untuk mendukung keputusan. Dalam SOC, intelligence membantu memprioritaskan alert dan respons.
IoC	Singkatan dari *Indicator of Compromise*. Lihat “Indicator of Compromise”.
IP Address	Alamat jaringan sebuah host. Dalam SOC, IP sering dipakai sebagai source IP, destination IP, atau IOC.
JSON	Format data ringan yang banyak dipakai dalam API, alert, webhook, OpenSearch, MISP, dan workflow SOAR.
KPI SOC	Indikator performa SOC, misalnya MTTD, MTTR, jumlah alert, false positive rate, severity, dan incident closed.
Lab SOC	Lingkungan latihan tertutup untuk membangun, menguji, dan mendokumentasikan SOC tanpa menyentuh sistem publik.
Lessons Learned	Pelajaran pasca-incident. Berisi apa yang terjadi, apa yang berhasil, apa yang gagal, dan perbaikan yang perlu dilakukan.
Log	Catatan aktivitas sistem, aplikasi, endpoint, jaringan, atau tools SOC. Log adalah bahan dasar deteksi dan investigasi.
Log Collection	Proses mengumpulkan log dari endpoint, server, aplikasi, sensor jaringan, dan tools keamanan.
Log Management	Proses menyimpan, mengindeks, mencari, memfilter, menormalisasi, dan mempertahankan log agar dapat dipakai SOC.
Malware Detection	Proses mendeteksi file atau aktivitas mencurigakan yang mengarah pada malware. Dalam lab open source dapat dibantu dengan FIM, YARA, rule, dan IOC.
Mapping MITRE	Proses mengaitkan alert atau aktivitas dengan taktik dan teknik MITRE ATT&CK agar mudah dipahami dalam konteks adversary.
Mean Time To Detect	Rata-rata waktu dari aktivitas mencurigakan terjadi sampai terdeteksi sebagai alert. Sering disingkat MTTD.
Mean Time To Respond	Rata-rata waktu dari alert terdeteksi sampai tindakan respons awal dilakukan. Sering disingkat MTTR.
MISP	Platform open source untuk threat intelligence, IOC sharing, event, attribute, feed, tag, dan galaxy.

MITRE ATT&CK	Knowledge base taktik dan teknik adversary yang dipakai untuk threat intelligence, detection engineering, blue team, red team, dan incident response.
MTTD	*Mean Time To Detect*, metrik waktu deteksi. Semakin rendah MTTD, semakin cepat SOC melihat ancaman.
MTTR	*Mean Time To Respond*, metrik waktu respons. Semakin rendah MTTR, semakin cepat SOC melakukan tindakan awal.
Network Metadata	Data ringkasan tentang trafik jaringan, seperti koneksi, DNS, HTTP, TLS, durasi, port, dan byte. Zeek sangat kuat pada metadata.
Network Sensor	Host atau VM yang membaca trafik jaringan untuk menghasilkan alert, flow, metadata, atau PCAP.
Network Visibility	Kemampuan melihat komunikasi jaringan: siapa berbicara dengan siapa, protokol apa, port apa, dan pola koneksi apa.
NIDS	*Network Intrusion Detection System*, IDS yang memantau trafik jaringan. Contoh: Suricata.
NIPS	*Network Intrusion Prevention System*, sistem yang tidak hanya mendeteksi tetapi juga dapat memblokir trafik. Harus digunakan hati-hati.
Noise	Event atau alert yang ramai tetapi tidak penting. Noise membuat analis sulit menemukan ancaman nyata.
Normalization	Proses menyamakan nama, format, dan tipe field dari berbagai sumber log agar query dan dashboard konsisten.
Observable	Objek teknis yang diamati dalam case, seperti IP, domain, hash, URL, username, hostname, atau file path.
OpenSearch	Mesin open source untuk indexing, search, dashboard, query, alert, dan analitik log.
OpenSearch Dashboards	Antarmuka visual untuk eksplorasi data, query, visualisasi, dan dashboard OpenSearch.
Open Source SOC	SOC yang dibangun dengan tools terbuka agar transparan, dapat dipelajari, fleksibel, dan cocok untuk lab mahasiswa.
osquery	Tool open source untuk menanyakan kondisi endpoint menggunakan query mirip SQL, misalnya proses, user, network listener, dan konfigurasi.
PCAP	*Packet Capture*, rekaman paket jaringan. PCAP berisi bukti detail tetapi dapat besar dan sensitif.

Playbook	Rangkaian langkah respons yang terstruktur, berisi trigger, condition, enrichment, approval, action, audit log, dan rollback.
Port Scan	Aktivitas memeriksa port terbuka pada host. Dalam lab SOC, port scan dipakai untuk menguji deteksi jaringan.
Preparation	Tahap persiapan incident response: menyiapkan tools, SOP, akun, backup, template, dan lab.
Query	Pertanyaan ke data log, misalnya mencari alert severity tinggi, source IP tertentu, atau login gagal.
Retention	Kebijakan berapa lama log disimpan. Retensi harus menyeimbangkan kebutuhan investigasi, kapasitas storage, dan risiko data sensitif.
Rule	Kondisi deteksi yang menentukan kapan event menjadi alert. Rule harus diuji agar tidak terlalu bising atau terlalu lemah.
Security Onion	Distribusi SOC open source untuk network visibility, host visibility, intrusion detection, log management, hunting, dan case management.
Severity	Tingkat prioritas alert atau incident. Severity harus dibaca bersama confidence, asset criticality, dan impact.
Shuffle	Platform SOAR open source untuk membuat workflow keamanan, webhook, enrichment, case creation, notifikasi, dan audit log.
SIEM	*Security Information and Event Management*, sistem untuk mengumpulkan, mengindeks, mengorelasikan, mencari, dan menganalisis log keamanan.
SOAR	*Security Orchestration, Automation, and Response*, pendekatan untuk menghubungkan tools SOC, mengotomasi workflow, dan membantu respons insiden.
SOC	*Security Operations Center*, pusat monitoring, deteksi, analisis, respons, dan perbaikan keamanan. SOC bertanggung jawab melakukan triage, investigasi, dan respons terhadap incident.
SOC Analyst	Analis yang membaca alert, memvalidasi bukti, melakukan triage, menulis case, dan memberi rekomendasi respons.
SOC Pillars	Pilar SOC: people, process, technology, dan data. SOC yang kuat membutuhkan keempatnya, bukan hanya tools.
Source IP	IP sumber aktivitas. Sering dipakai untuk analisis brute force, port scan, koneksi mencurigakan, dan IOC.
Suricata	Engine open source untuk deteksi jaringan berbasis rule. Suricata menghasilkan alert dan Eve JSON.

Tag	Label untuk mengelompokkan event, IOC, case, atau alert. Dalam MISP, tag membantu klasifikasi dan distribusi intelligence.
Task	Tugas investigasi dalam case, misalnya cek alert Wazuh, cek OpenSearch, validasi IOC, atau buat timeline.
TheHive	Platform case management yang banyak dikenal dalam ekosistem SOC. Dalam buku ini dibahas sebagai referensi integrasi, dengan perhatian khusus pada status lisensi/versi.
Threat Actor	Individu, kelompok, atau entitas yang melakukan atau mengarahkan aktivitas ancaman siber.
Threat Hunting	Pencarian proaktif terhadap indikator atau pola ancaman yang belum tentu memicu alert otomatis. Wazuh dapat membantu threat hunting melalui log analysis, MITRE mapping, osquery, dan command monitoring.
Threat Intelligence	Informasi ancaman yang sudah dianalisis agar dapat membantu prioritas alert, threat hunting, incident response, dan strategi keamanan.
Timeline	Urutan kronologis kejadian dalam incident. Timeline menjadi dasar analisis, laporan, dan lessons learned.
Triage	Proses memilah dan memprioritaskan alert berdasarkan severity, confidence, asset criticality, impact, dan evidence.
TTP	*Tactics, Techniques, and Procedures*, cara kerja adversary. TTP lebih tahan lama dibanding IOC karena menggambarkan pola perilaku.
Ubuntu Server	Sistem operasi server berbasis Linux yang dipakai sebagai fondasi lab open source.
VM	*Virtual Machine*, mesin virtual untuk menjalankan endpoint, SOC server, sensor, atau target lab secara terisolasi.
Vulnerability	Kelemahan pada sistem, aplikasi, konfigurasi, atau paket yang dapat dimanfaatkan penyerang.
Vulnerability Detection	Kemampuan mendeteksi kelemahan pada endpoint atau sistem, misalnya paket rentan atau konfigurasi berisiko.
Webhook	Endpoint HTTP untuk menerima data dari tool lain. Dalam SOAR, webhook sering dipakai menerima alert JSON.
Workflow	Rangkaian langkah otomasi, misalnya menerima alert, parsing JSON, enrichment, membuat case, dan mengirim notifikasi.
Wazuh	Platform open source SIEM/XDR untuk log analysis, FIM, vulnerability detection, alert, rule, agent, active response, dan compliance.

Wazuh Agent	Komponen Wazuh pada endpoint yang mengirim event dan log ke Wazuh server.
Wazuh API	API untuk mengakses dan mengelola data Wazuh, seperti agent, alert, status manager, dan integrasi automation.
Wazuh Dashboard	Antarmuka visual Wazuh untuk melihat agent, alert, FIM, vulnerability, compliance, dan status keamanan.
Wazuh Indexer	Komponen penyimpanan dan indexing alert Wazuh agar data dapat dicari dan divisualisasikan.
Wazuh Server	Komponen pusat Wazuh yang menerima event dari agent, menjalankan decoder/rule, dan menghasilkan alert.
YARA	Tool open source untuk mendeteksi pola pada file, sering dipakai untuk deteksi artefak malware atau file mencurigakan dalam lab.
Zeek	Tool open source untuk analisis metadata jaringan, seperti conn.log, dns.log, http.log, dan protokol lain.

Referensi

Basta, A., Basta, N., Anwar, W., & Essar, M. I. (2025). *Open-source security operations center (SOC): A complete guide to establishing, managing, and maintaining a modern SOC*. John Wiley & Sons. https://www.wiley.com/en-mx/Open%2BSource%2BSecurity%2BOperations%2BCenter%2B%28SOC%29%3A%2BA%2BComplete%2BGuide%2Bto%2BEstablishing%2C%2BManaging%2C%2Band%2BMaintaining%2Ba%2BModern%2BSOC-p-9781394201600

Canonical Ltd. (2026). *Ubuntu Server documentation*. Ubuntu. https://ubuntu.com/server/docs/

CIRCL. (n.d.). *PyMISP automation documentation*. MISP Project. https://www.circl.lu/doc/misp/automation/

Cichonski, P., Millar, T., Grance, T., & Scarfone, K. (2012). *Computer security incident handling guide* (NIST Special Publication 800-61 Rev. 2). National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf

DFIR-IRIS. (n.d.). *Collaborative incident response platform*. https://www.dfir-iris.org/

DFIR-IRIS. (n.d.). *DFIR-IRIS documentation*. https://docs.dfir-iris.org/

Docker Inc. (n.d.). *Docker documentation*. https://docs.docker.com/

Docker Inc. (n.d.). *Install Docker Engine on Ubuntu*. Docker Documentation. https://docs.docker.com/engine/install/ubuntu/

Gupta, R. (2024). *Security monitoring with Wazuh: A hands-on guide to effective enterprise security using real-life use cases in Wazuh*. Packt Publishing. https://www.packtpub.com/en-us/product/security-monitoring-with-wazuh-9781837632152

MISP Project. (n.d.). *MISP data models: MISP core format*. https://www.misp-project.org/datamodels/

MISP Project. (n.d.). *MISP features and functionalities*. https://www.misp-project.org/features/

MISP Project. (n.d.). *MISP OpenAPI specification*. https://www.misp-project.org/openapi/

MISP Project. (n.d.). *MISP threat intelligence sharing platform*. https://www.misp-project.org/

MITRE. (n.d.). *Enterprise tactics*. MITRE ATT&CK®. https://attack.mitre.org/tactics/

MITRE. (n.d.). *MITRE ATT&CK®*. https://attack.mitre.org/

MITRE. (2026). *ATT&CK updates*. https://attack.mitre.org/resources/updates/

Nelson, A., Barcklow, D., Benzel, T., Booth, H., Dempsey, K., & Pillitteri, V. (2025). *Incident response recommendations and considerations for cybersecurity risk management: A CSF*

2.0 community profile (NIST Special Publication 800-61 Rev. 3). National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r3.pdf

n8n. (n.d.). *n8n workflow automation platform*. https://github.com/n8n-io/n8n

Open Information Security Foundation. (n.d.). *Suricata user guide: Eve JSON format*. Suricata Documentation. https://docs.suricata.io/en/latest/output/eve/eve-json-format.html

Open Information Security Foundation. (n.d.). *Suricata user guide: Eve JSON output*. Suricata Documentation. https://docs.suricata.io/en/latest/output/eve/eve-json-output.html

OpenSearch Project. (n.d.). *Alerting*. OpenSearch Documentation. https://docs.opensearch.org/latest/observing-your-data/alerting/index/

OpenSearch Project. (n.d.). *Create index API*. OpenSearch Documentation. https://docs.opensearch.org/latest/api-reference/index-apis/create-index/

OpenSearch Project. (n.d.). *Dashboards*. OpenSearch Documentation. https://docs.opensearch.org/latest/dashboards/

OpenSearch Project. (n.d.). *Mappings*. OpenSearch Documentation. https://docs.opensearch.org/latest/mappings/

OpenSearch Project. (n.d.). *OpenSearch*. https://opensearch.org/

OpenSearch Project. (n.d.). *Query DSL*. OpenSearch Documentation. https://docs.opensearch.org/latest/query-dsl/

OpenSearch Project. (n.d.). *Range query*. OpenSearch Documentation. https://docs.opensearch.org/latest/query-dsl/term/range/

OpenSearch Project. (n.d.). *Visualizing data*. OpenSearch Documentation. https://docs.opensearch.org/latest/dashboards/visualize/viz-index/

Purbo, O. W. (2026). *Open Source SOC Mastery: Membangun Security Operations Center Modern dengan Tools Gratis*. OnnoCenter.

Security Onion Solutions. (n.d.). *Security Onion documentation: Architecture*. https://docs.securityonion.net/en/2.4/architecture.html

Security Onion Solutions. (n.d.). *Security Onion documentation: Introduction*. https://docs.securityonion.net/en/2.4/introduction.html

Security Onion Solutions. (n.d.). *Security Onion software*. https://securityonionsolutions.com/software/

Shuffle. (n.d.). *Shuffle documentation: Getting started*. https://github.com/shuffle/shuffle-docs/blob/master/docs/getting_started.md

Shuffle. (n.d.). *Shuffle: Security automation platform*. https://github.com/shuffle/shuffle

StrangeBee. (n.d.). *Cortex*. https://strangebee.com/cortex/

StrangeBee. (n.d.). *TheHive and Cortex documentation*. https://docs.strangebee.com/

Wall, T., & Rodrick, J. (2024). *Jump-start your SOC analyst career: A roadmap to cybersecurity success* (2nd ed.). Apress. https://doi.org/10.1007/979-8-8688-0345-1

Wazuh. (n.d.). *Active response*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/capabilities/active-response/index.html

Wazuh. (n.d.). *Detecting a brute-force attack*. Wazuh Documentation. https://documentation.wazuh.com/current/proof-of-concept-guide/detect-brute-force-attack.html

Wazuh. (n.d.). *File integrity monitoring*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/capabilities/file-integrity/index.html

Wazuh. (n.d.). *File integrity monitoring and YARA*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/capabilities/malware-detection/fim-yara.html

Wazuh. (n.d.). *Getting started*. Wazuh Documentation. https://documentation.wazuh.com/current/getting-started/index.html

Wazuh. (n.d.). *MITRE ATT&CK framework*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/ruleset/mitre.html

Wazuh. (n.d.). *Rules*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/ruleset/rules/index.html

Wazuh. (n.d.). *Testing decoders and rules*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/ruleset/testing.html

Wazuh. (n.d.). *User manual*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/index.html

Wazuh. (n.d.). *Wazuh API reference*. Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/api/reference.html

Wazuh. (n.d.). *Wazuh platform*. https://wazuh.com/

YARA. (n.d.). *YARA documentation*. https://yara.readthedocs.io/

Zeek Project. (n.d.). *Book of Zeek*. Zeek Documentation. https://docs.zeek.org/

Zeek Project. (n.d.). *conn.log*. Zeek Documentation. https://docs.zeek.org/en/master/reference/logs/conn.html

Zeek Project. (n.d.). *dns.log*. Zeek Documentation. https://docs.zeek.org/en/master/reference/logs/dns.html

Zeek Project. (n.d.). *Zeek logs*. Zeek Documentation. https://docs.zeek.org/en/v6.2.1/logs/

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