CYBERSECURITY AND ETHICAL HACKING PROJECT

TITLE: ENTERPRISE-LEVEL THREAT DETECTION & INCIDENT RESPONSE SYSTEM

-By KARTHIKEYA VALISETTY

Submission date – 02/04/2025

Security Monitoring and Incident Response Report

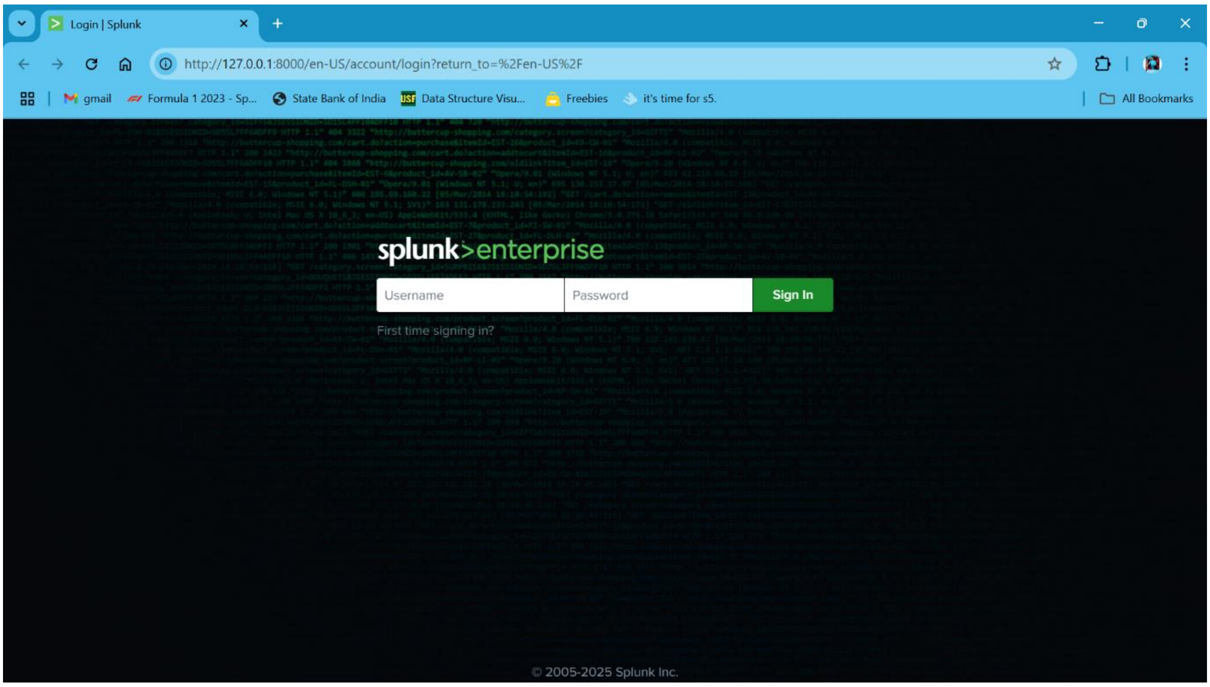
1. Introduction

This report provides a detailed analysis of the security monitoring architecture, log forwarding setup, automation scripts, intrusion detection system (IDS) configurations, and penetration testing activities. The goal is to document the setup, analyze logs, and provide security recommendations.

2. Splunk Setup Architecture

2.1 Overview

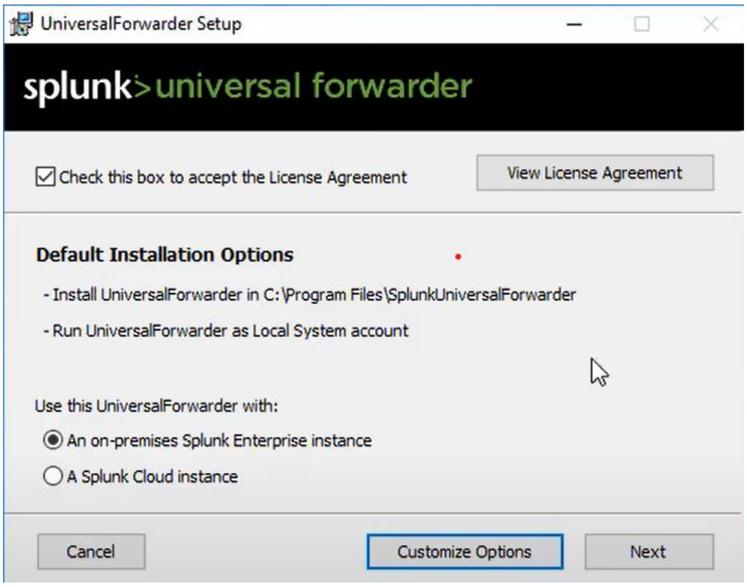
Splunk has been configured to collect and analyze logs from multiple devices using the Universal Forwarder. The logs are then indexed and monitored through Splunk's web interface.



Continued….

2.2 Configuration Details

- **Universal Forwarder Setup:** Installed and configured on multiple devices.



UniversalForwarder Setup

splunk>universal forwarder

☑ Check this box to accept the License Agreement

Default Installation Options

- Install UniversalForwarder in C:\Program Files\SplunkUniversalForwarder

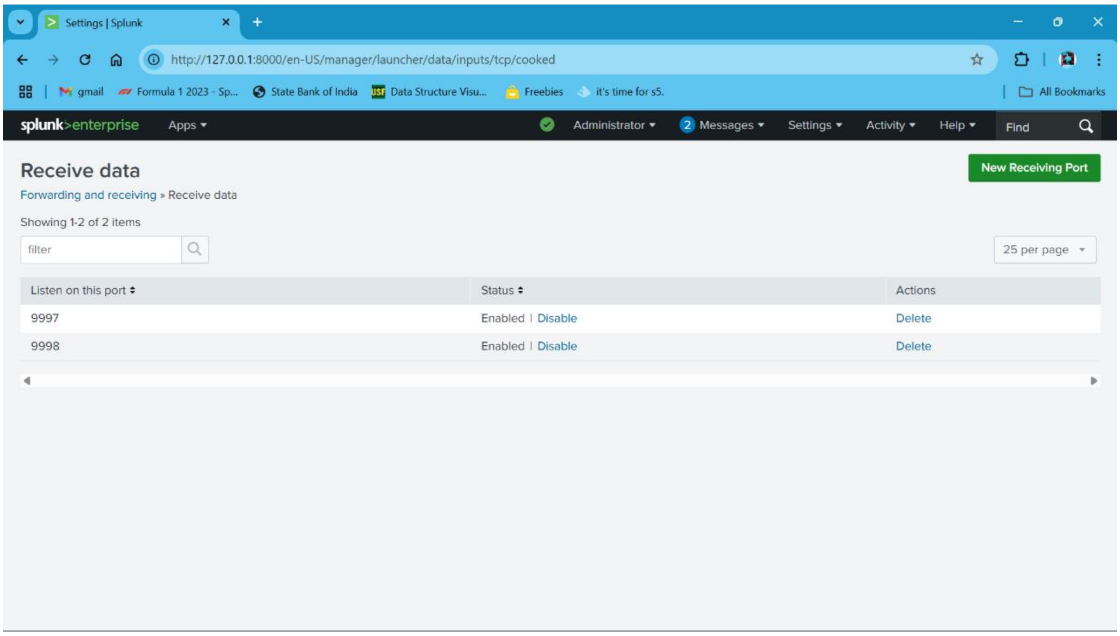
- Run UniversalForwarder as Local System account

Use this UniversalForwarder with:

● An on-premises Splunk Enterprise instance

○ A Splunk Cloud instance

- **Receiving Ports:** Configured on Splunk to accept logs from Universal Forwarders.



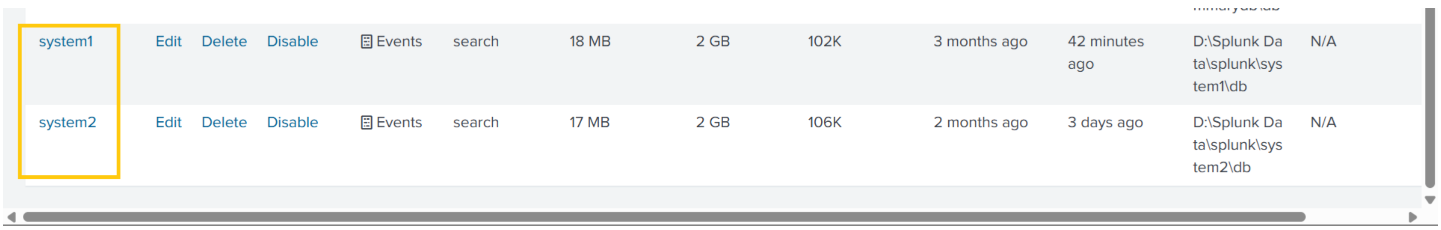
Receive data

Forwarding and receiving » Receive data

Showing 1-2 of 2 items

Listen on this port	Status	Actions
9997	Enabled | Disable	Delete
9998	Enabled | Disable	Delete

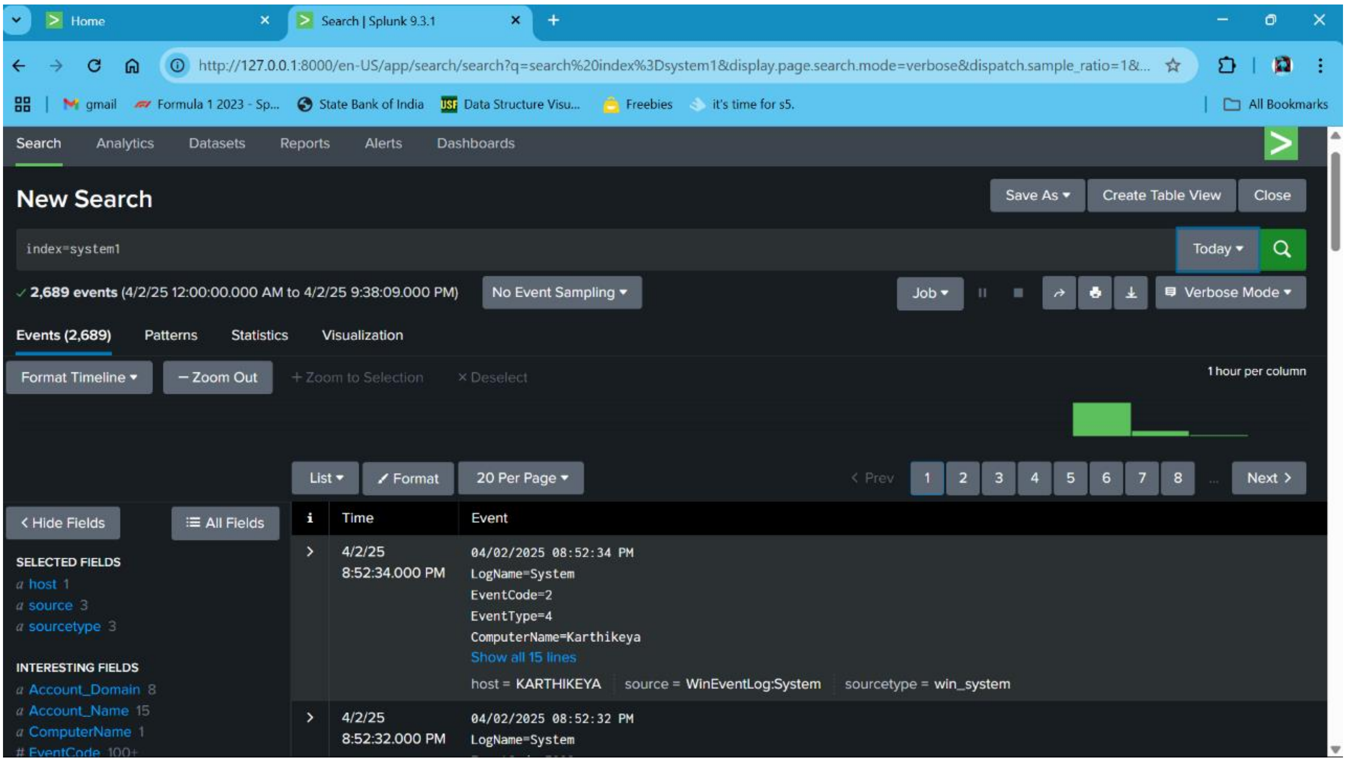
- **Indexing:** Separate indexes created for each system.



system1	Edit	Delete	Disable	Events	search	18 MB	2 GB	102K	3 months ago	42 minutes ago	D:\Splunk Data\splunk\system1\db	N/A
system2	Edit	Delete	Disable	Events	search	17 MB	2 GB	106K	2 months ago	3 days ago	D:\Splunk Data\splunk\system2\db	N/A

Continued….

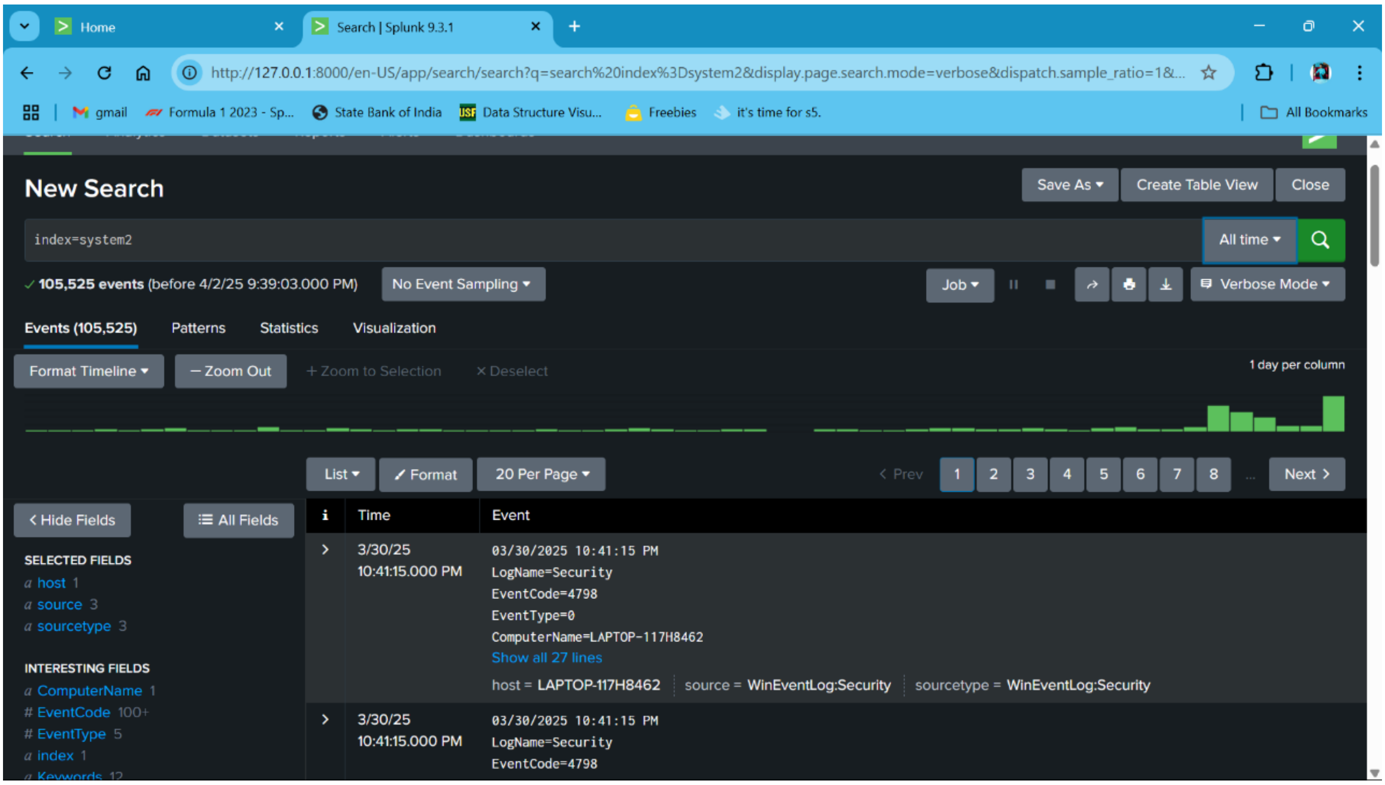
- **Log Forwarding:** Successfully verified for each system.



```
index=system1
```

✓ 2,689 events (4/2/25 12:00:00.000 AM to 4/2/25 9:38:09.000 PM)

```
4/2/25 8:52:34.000 PM
04/02/2025 08:52:34 PM
LogName=System
EventCode=2
EventType=4
ComputerName=Karthikeya
Show all 15 lines
host = KARTHIKEYA   source = WinEventLog:System   sourcetype = win_system

4/2/25 8:52:32.000 PM
04/02/2025 08:52:32 PM
LogName=System
```



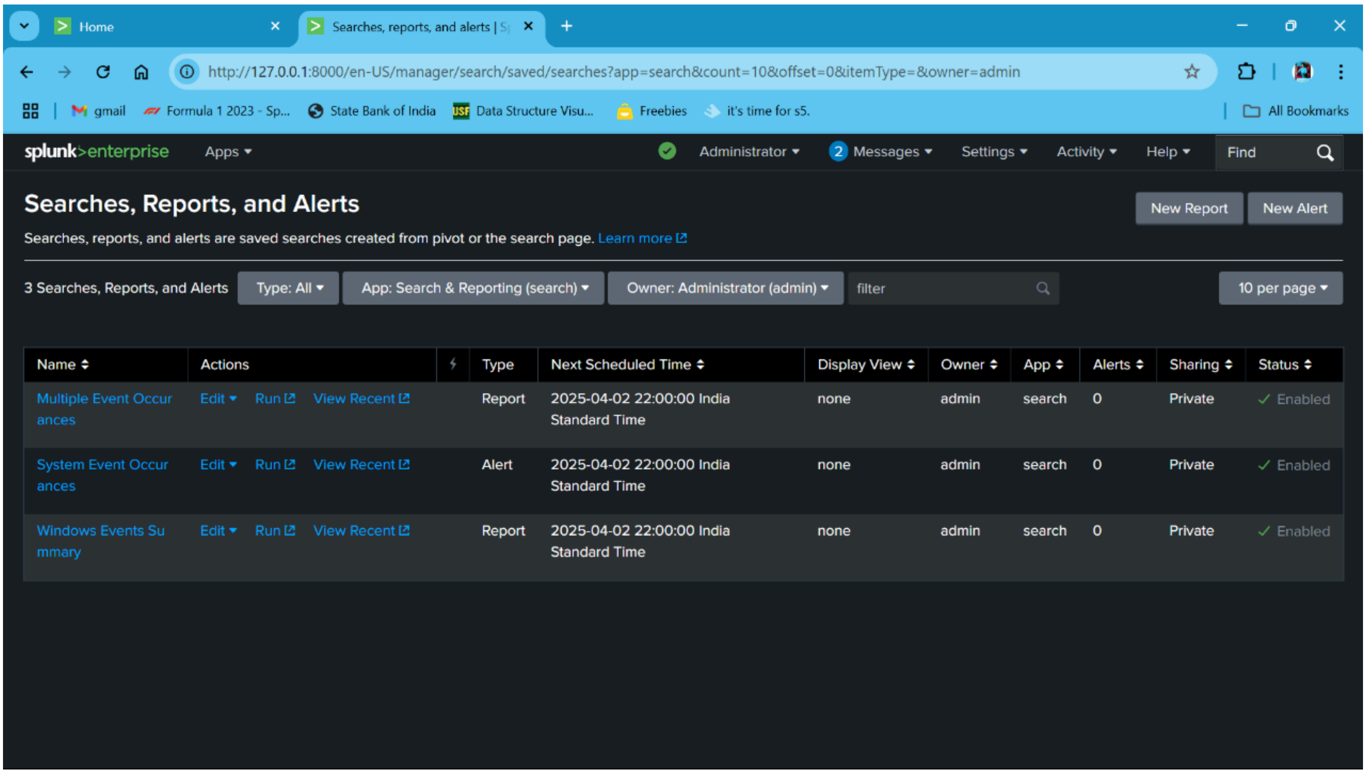
```
index=system2
```

✓ 105,525 events (before 4/2/25 9:39:03.000 PM)

```
3/30/25 10:41:15.000 PM
03/30/2025 10:41:15 PM
LogName=Security
EventCode=4798
EventType=0
ComputerName=LAPTOP-117H8462
Show all 27 lines
host = LAPTOP-117H8462   source = WinEventLog:Security   sourcetype = WinEventLog:Security

3/30/25 10:41:15.000 PM
03/30/2025 10:41:15 PM
LogName=Security
EventCode=4798
```

Continued….

- **Alerts and Reports:** Configured periodic alerts and automated reports.



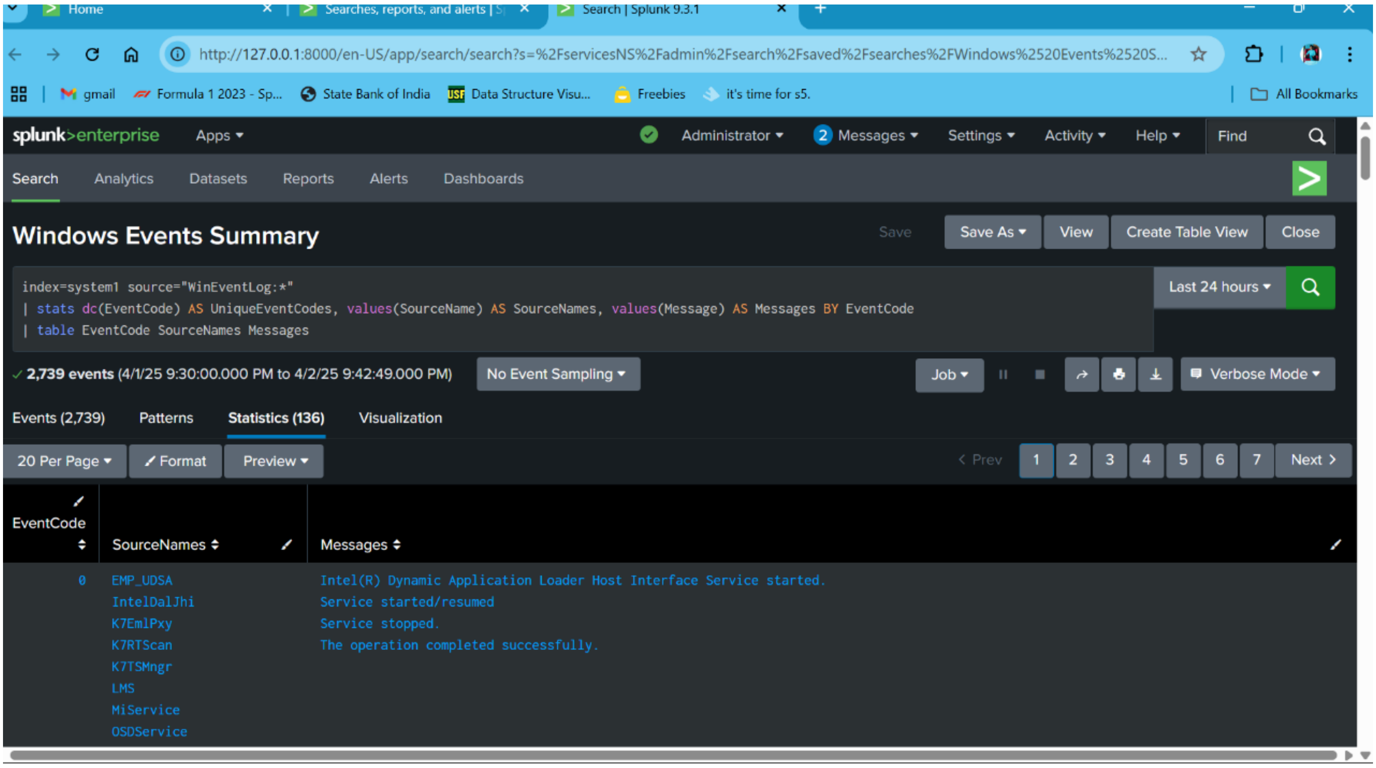
Searches, Reports, and Alerts

Searches, reports, and alerts are saved searches created from pivot or the search page. Learn more

3 Searches, Reports, and Alerts

Name	Actions		Type	Next Scheduled Time	Display View	Owner	App	Alerts	Sharing	Status
Multiple Event Occurances	Edit Run View Recent		Report	2025-04-02 22:00:00 India Standard Time	none	admin	search	0	Private	✓ Enabled
System Event Occurances	Edit Run View Recent		Alert	2025-04-02 22:00:00 India Standard Time	none	admin	search	0	Private	✓ Enabled
Windows Events Summary	Edit Run View Recent		Report	2025-04-02 22:00:00 India Standard Time	none	admin	search	0	Private	✓ Enabled

- **Search Queries:** Queries for monitoring logs and triggering alerts.



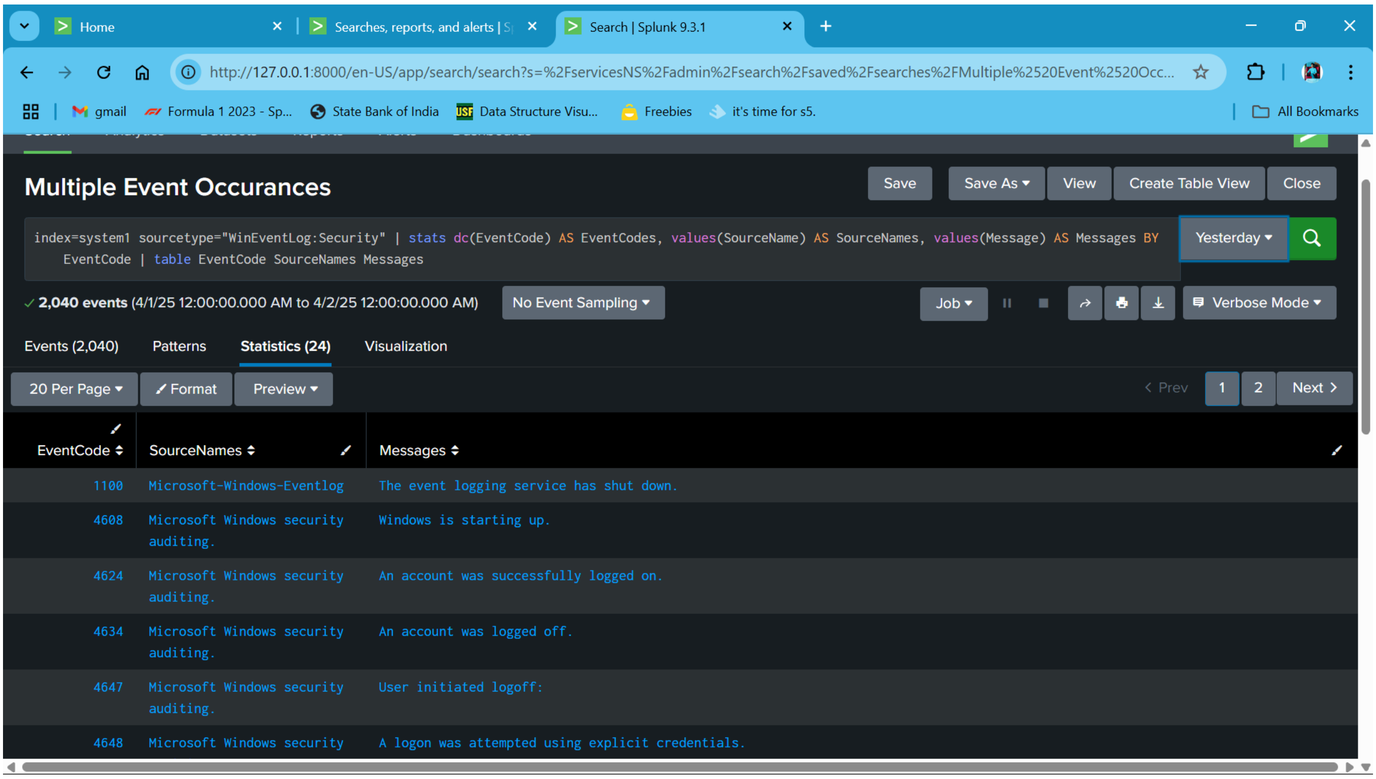
Windows Events Summary

```
index=system1 source="WinEventLog:*"
| stats dc(EventCode) AS UniqueEventCodes, values(SourceName) AS SourceNames, values(Message) AS Messages BY EventCode
| table EventCode SourceNames Messages
```

✓ 2,739 events (4/1/25 9:30:00.000 PM to 4/2/25 9:42:49.000 PM)

EventCode	SourceNames	Messages
0	EMP_UDSA IntelDalJhi K7EmlPxy K7RTScan K7TSMngr LMS MiService OSDService	Intel(R) Dynamic Application Loader Host Interface Service started. Service started/resumed Service stopped. The operation completed successfully.

Continued….

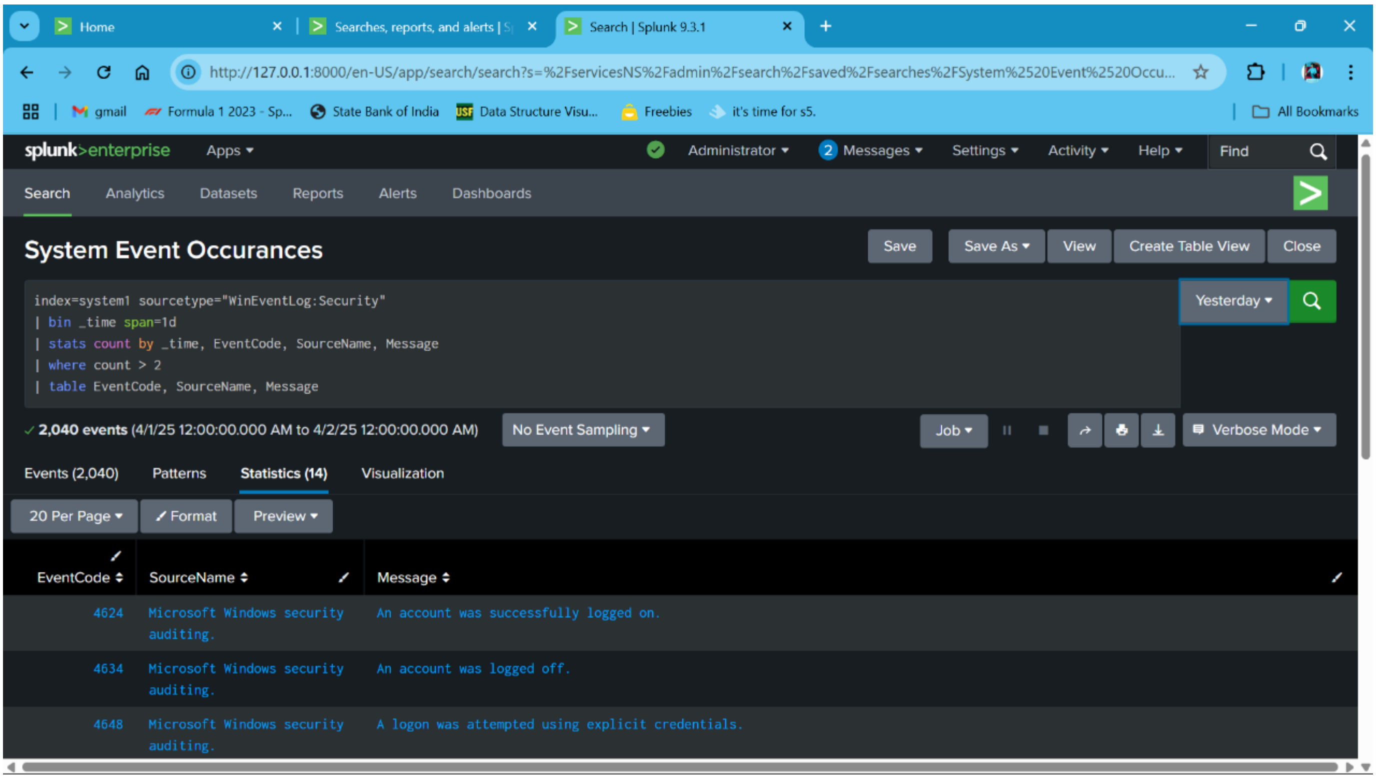


Multiple Event Occurances

```
index=system1 sourcetype="WinEventLog:Security" | stats dc(EventCode) AS EventCodes, values(SourceName) AS SourceNames, values(Message) AS Messages BY
    EventCode | table EventCode SourceNames Messages
```

✓ 2,040 events (4/1/25 12:00:00.000 AM to 4/2/25 12:00:00.000 AM)

EventCode	SourceNames	Messages
1100	Microsoft-Windows-Eventlog	The event logging service has shut down.
4608	Microsoft Windows security auditing.	Windows is starting up.
4624	Microsoft Windows security auditing.	An account was successfully logged on.
4634	Microsoft Windows security auditing.	An account was logged off.
4647	Microsoft Windows security auditing.	User initiated logoff:
4648	Microsoft Windows security	A logon was attempted using explicit credentials.



System Event Occurances

```
index=system1 sourcetype="WinEventLog:Security"
| bin _time span=1d
| stats count by _time, EventCode, SourceName, Message
| where count > 2
| table EventCode, SourceName, Message
```

✓ 2,040 events (4/1/25 12:00:00.000 AM to 4/2/25 12:00:00.000 AM)

EventCode	SourceName	Message
4624	Microsoft Windows security auditing.	An account was successfully logged on.
4634	Microsoft Windows security auditing.	An account was logged off.
4648	Microsoft Windows security auditing.	A logon was attempted using explicit credentials.

Continued….

3. Python Log Analysis Automation

3.1 Overview

A Python script was developed for log analysis automation, executing predefined search queries and sending summarized reports via email.

3.2 Configuration Details

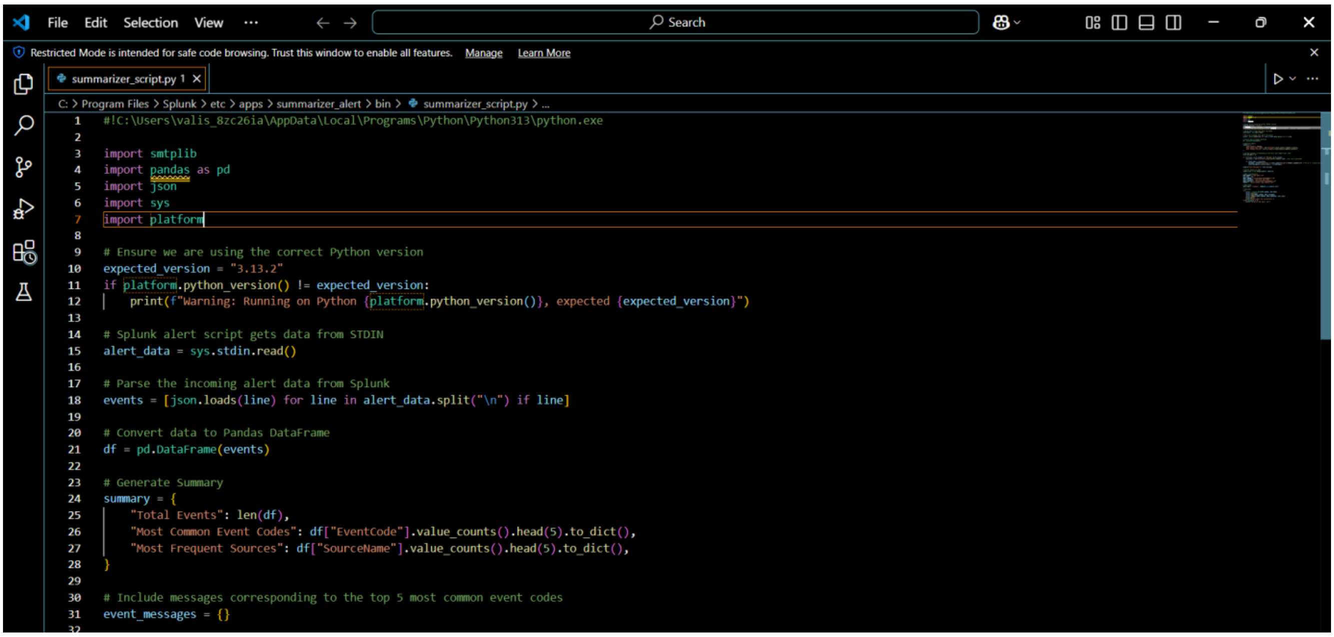
- **Source Code:** Below is the Python script used for log analysis automation.

```
import smtplib
import pandas as pd
import json
import sys
import platform

# Ensure we are using the correct Python version
expected_version = "3.13.2"
if platform.python_version() != expected_version:
    print(f"Warning: Running on Python {platform.python_version()}, expected {expected_version}")

# Splunk alert script gets data from STDIN
alert_data = sys.stdin.read()

# Parse the incoming alert data from Splunk
events = [json.loads(line) for line in alert_data.split("\n") if line]

# Convert data to Pandas DataFrame
df = pd.DataFrame(events)

# Generate Summary
summary = {
    "Total Events": len(df),
    "Most Common Event Codes": df["EventCode"].value_counts().head(5).to_dict(),
    "Most Frequent Sources": df["SourceName"].value_counts().head(5).to_dict(),
}

# Include messages corresponding to the top 5 most common event codes
event_messages = {}

if "EventCode" in df.columns and "Message" in df.columns:
    top_events = df["EventCode"].value_counts().head(5).index  # Get top 5 EventCodes

    for event_code in top_events:
        messages = df[df["EventCode"] == event_code]["Message"].dropna().unique()[:3]  # Get up to 3
unique messages
        event_messages[str(event_code)] = list(messages)

summary["Event Messages"] = event_messages

# Convert Summary to JSON
summary_text = json.dumps(summary, indent=4)
```

```
# Email Configuration
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
EMAIL_SENDER = ""
EMAIL_PASSWORD = ""
EMAIL_RECEIVER = ""
SUBJECT = "Splunk Windows Logs Summary Alert"

# Email Body
email_body = f"Subject: {SUBJECT}\n\n{summary_text}"

# Send Email
try:
    server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
    server.starttls()
    server.login(EMAIL_SENDER, EMAIL_PASSWORD)
    server.sendmail(EMAIL_SENDER, EMAIL_RECEIVER, email_body)
    server.quit()
    print("Summary email sent successfully.")
except Exception as e:

    print(f"Failed to send email: {e}")
```

Execution command: #!C:\Users\valis_8zc26ia\AppData\Local\Programs\Python\Python313\python.exe

- **Script Code:** Screenshot taken of the Python script.



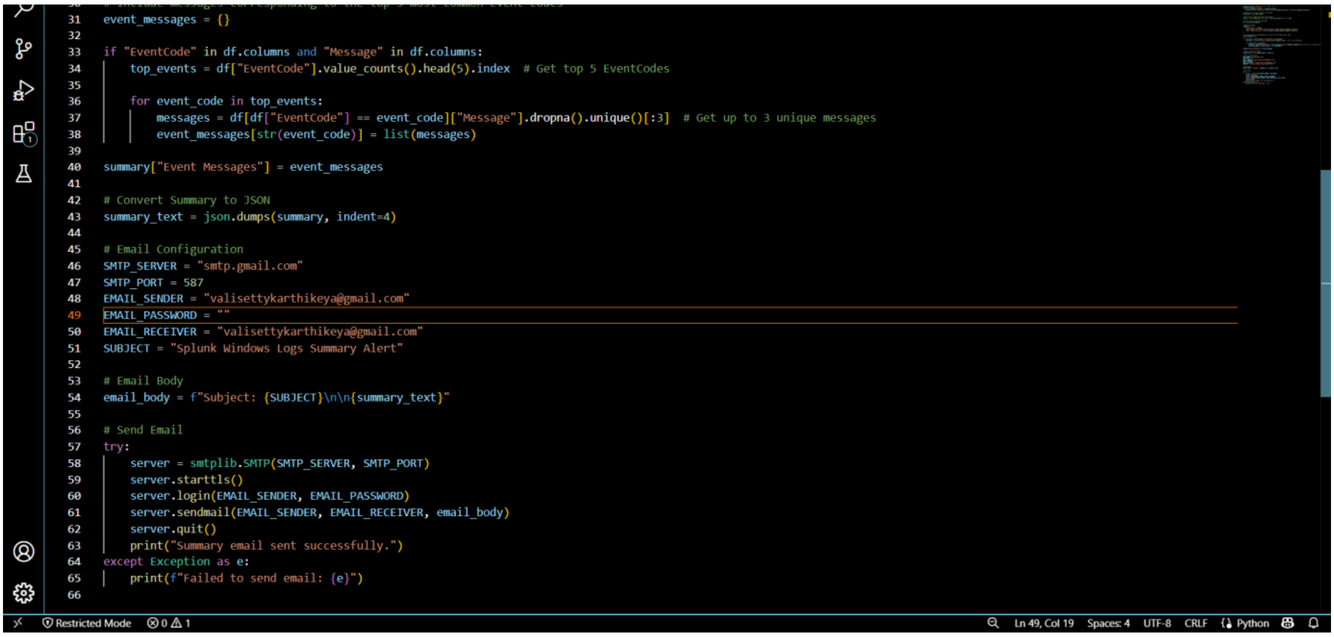
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

C: > Program Files > Splunk > etc > apps > summarizer_alert > bin > summarizer_script.py > ...

```
#!C:\Users\valis_8zc26ia\AppData\Local\Programs\Python\Python313\python.exe

import smtplib
import pandas as pd
import json
import sys
import platform

# Ensure we are using the correct Python version
expected_version = "3.13.2"
if platform.python_version() != expected_version:
    print(f"Warning: Running on Python {platform.python_version()}, expected {expected_version}")

# Splunk alert script gets data from STDIN
alert_data = sys.stdin.read()

# Parse the incoming alert data from Splunk
events = [json.loads(line) for line in alert_data.split("\n") if line]

# Convert data to Pandas DataFrame
df = pd.DataFrame(events)

# Generate Summary
summary = {
    "Total Events": len(df),
    "Most Common Event Codes": df["EventCode"].value_counts().head(5).to_dict(),
    "Most Frequent Sources": df["SourceName"].value_counts().head(5).to_dict(),
}

# Include messages corresponding to the top 5 most common event codes
event_messages = {}
```

Continued….

- **Script Code:** Screenshot taken of the Python script.

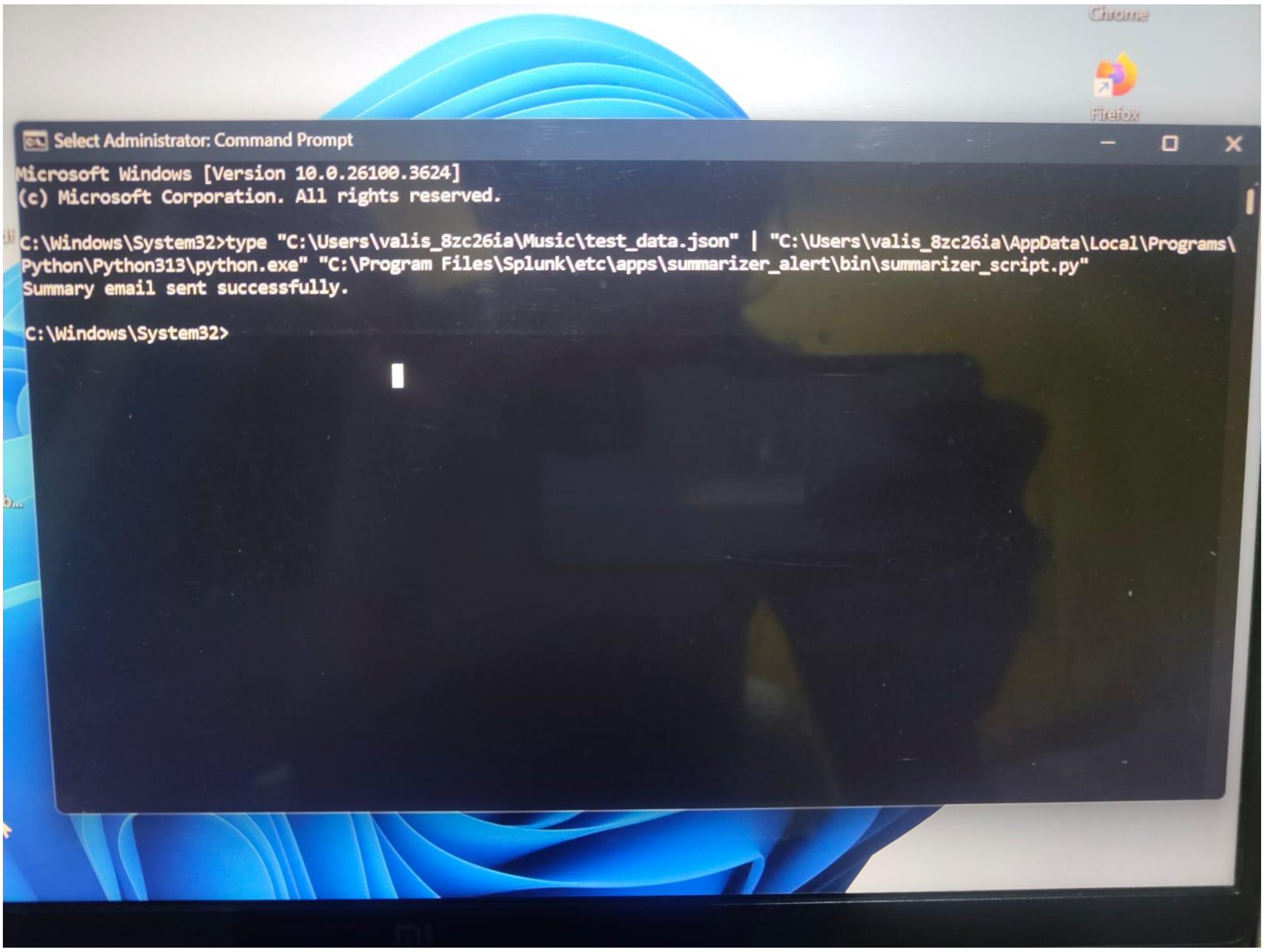


```
event_messages = {}

if "EventCode" in df.columns and "Message" in df.columns:
    top_events = df["EventCode"].value_counts().head(5).index  # Get top 5 EventCodes

    for event_code in top_events:
        messages = df[df["EventCode"] == event_code]["Message"].dropna().unique()[:3]  # Get up to 3 unique messages
        event_messages[str(event_code)] = list(messages)

summary["Event Messages"] = event_messages

# Convert Summary to JSON
summary_text = json.dumps(summary, indent=4)

# Email Configuration
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
EMAIL_SENDER = "valisettykarthikeya@gmail.com"
EMAIL_PASSWORD = ""
EMAIL_RECEIVER = "valisettykarthikeya@gmail.com"
SUBJECT = "Splunk Windows Logs Summary Alert"

# Email Body
email_body = f"Subject: {SUBJECT}\n\n{summary_text}"

# Send Email
try:
    server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
    server.starttls()
    server.login(EMAIL_SENDER, EMAIL_PASSWORD)
    server.sendmail(EMAIL_SENDER, EMAIL_RECEIVER, email_body)
    server.quit()
    print("Summary email sent successfully.")
except Exception as e:
    print(f"Failed to send email: {e}")
```

- **Execution Commands:** Commands required to run the script documented.

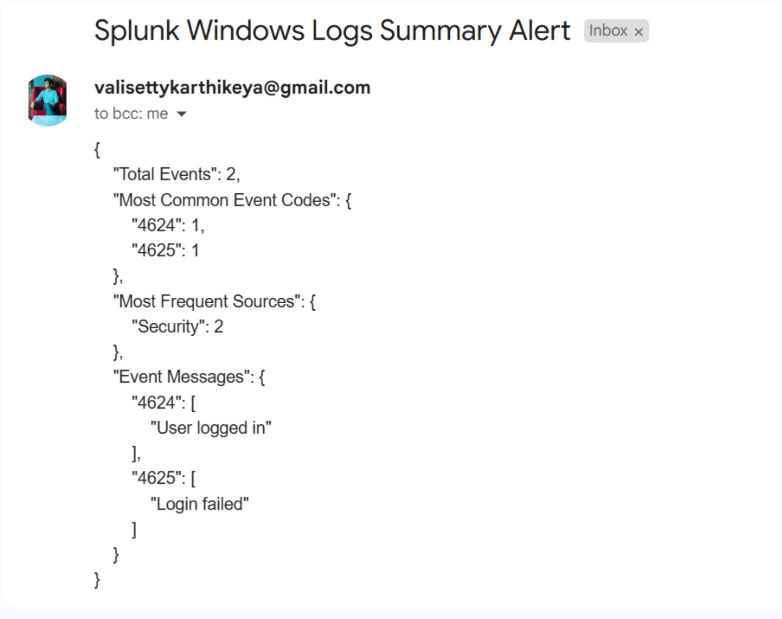
  "C:\Users\valis_8zc26ia\Music\test_data.json" | "C:\Users\valis_8zc26ia\AppData\Local\Programs\Python\Python313\python.exe" "C:\Program Files\Splunk\etc\apps\summarizer_alert\bin\summarizer_script.py"

- **Output Verification:** Screenshot of script execution results.



```
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>type "C:\Users\valis_8zc26ia\Music\test_data.json" | "C:\Users\valis_8zc26ia\AppData\Local\Programs\Python\Python313\python.exe" "C:\Program Files\Splunk\etc\apps\summarizer_alert\bin\summarizer_script.py"
Summary email sent successfully.

C:\Windows\System32>
```

- **Email Notifications:** Verified successful email alerts upon execution.



Splunk Windows Logs Summary Alert Inbox ×

valisettykarthikeya@gmail.com

to bcc: me

```
{
    "Total Events": 2,
    "Most Common Event Codes": {
        "4624": 1,
        "4625": 1
    },
    "Most Frequent Sources": {
        "Security": 2
    },
    "Event Messages": {
        "4624": [
            "User logged in"
        ],
        "4625": [
            "Login failed"
        ]
    }
}
```

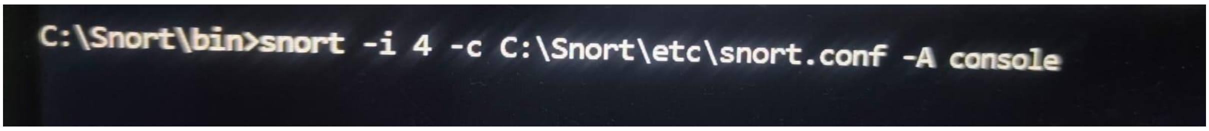
4. Snort Intrusion Detection System Setup

4.1 Overview

Snort has been installed and configured to detect network intrusions and generate logs for security analysis.

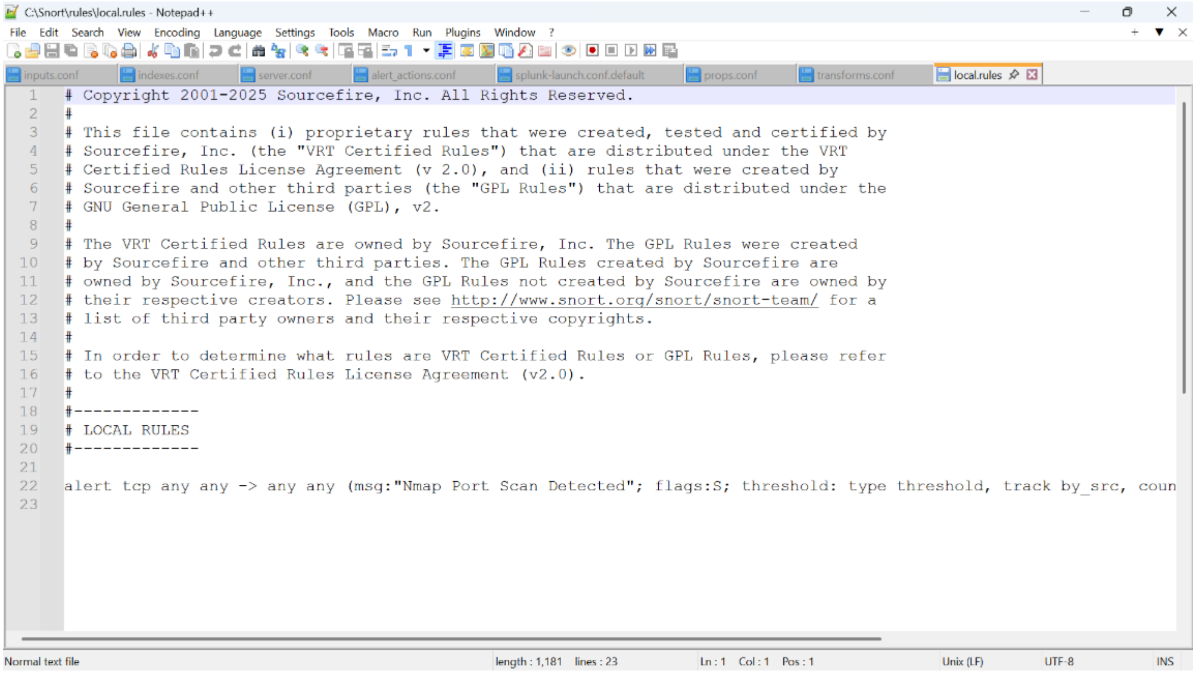
4.2 Configuration Details

- **Snort Configuration:** Configured snort.conf for IDS mode.



```
C:\Snort\bin>snort -i 4 -c C:\Snort\etc\snort.conf -A console
```

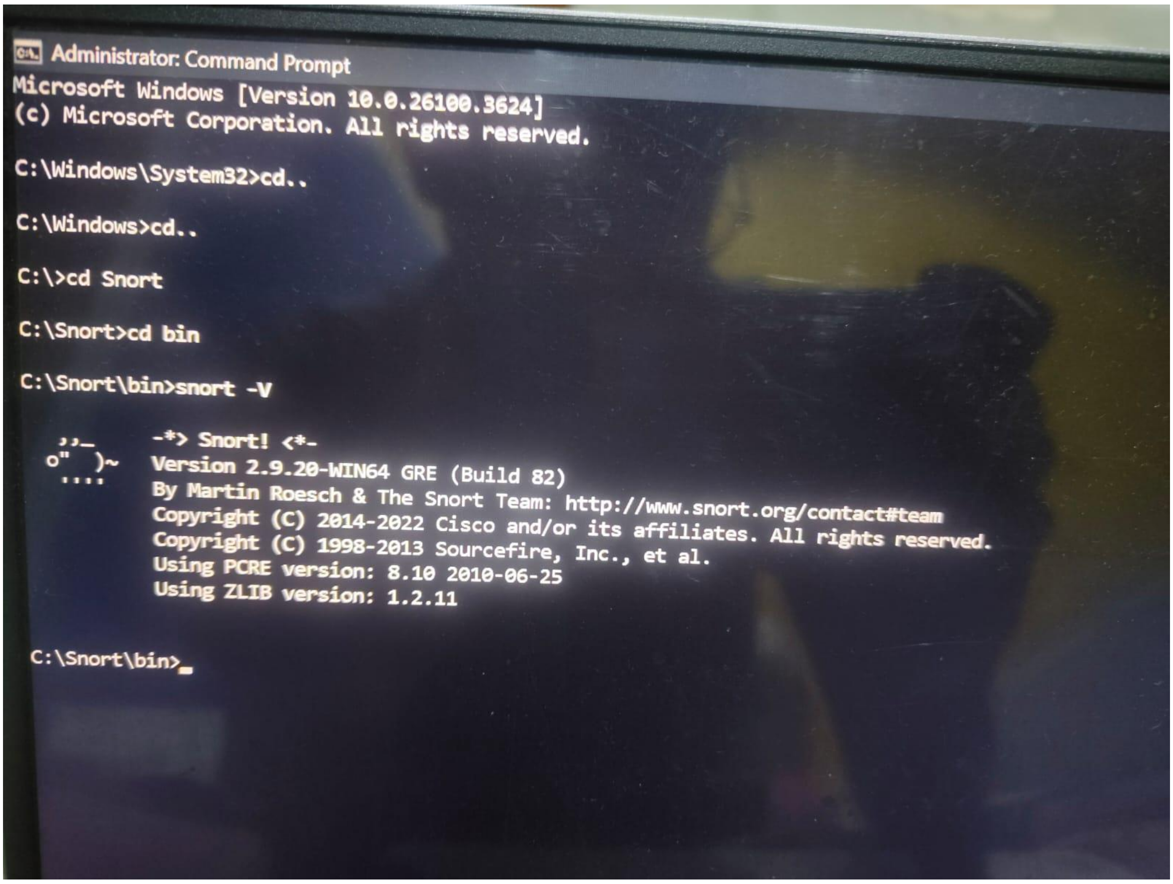
- **Custom Rules:** Added local rules in local.rules file to detect suspicious activities.



C:\Snort\rules\local.rules - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inputs.conf
indexes.conf
server.conf
alert_actions.conf
splunk-launch.conf.default
props.conf
transforms.conf
local.rules
Normal text file
length : 1,181 lines : 23
Ln : 1 Col : 1 Pos : 1
Unix (LF)
UTF-8
INS

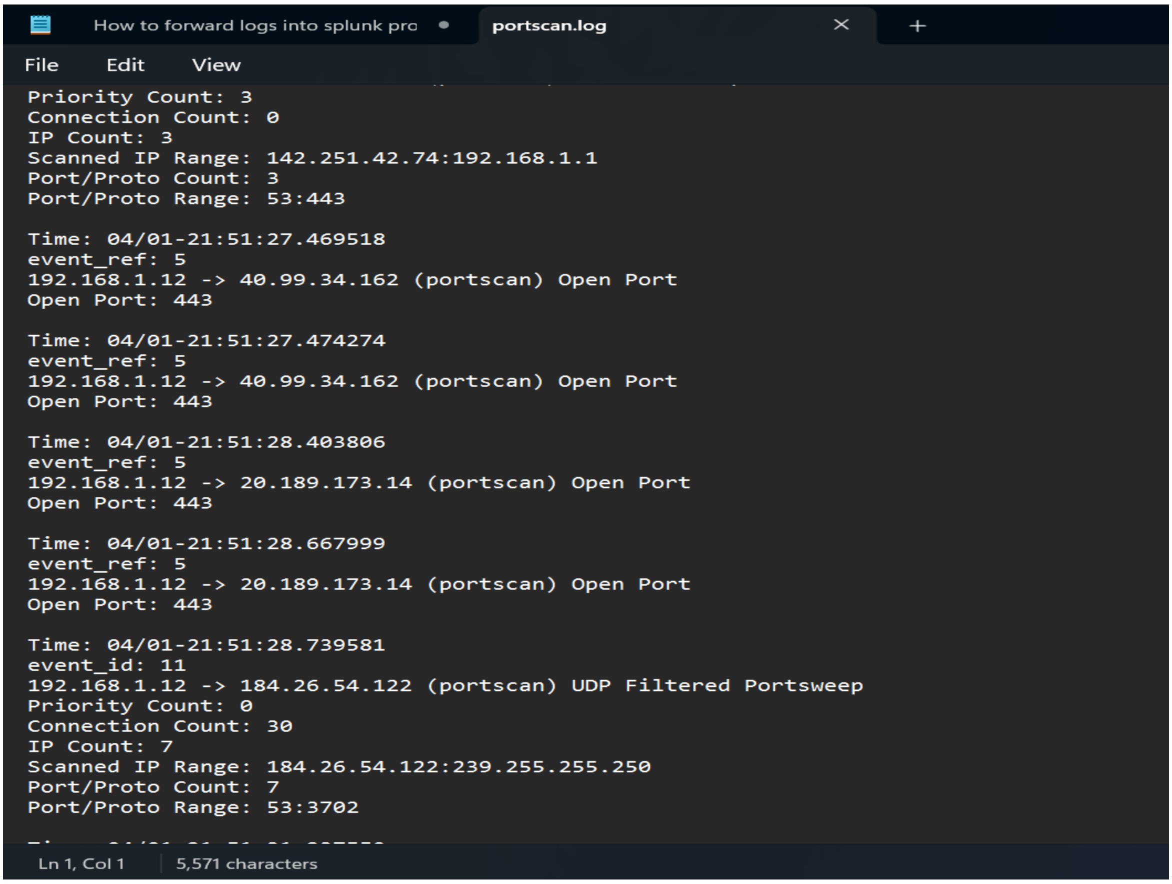
```
# Copyright 2001-2025 Sourcefire, Inc. All Rights Reserved.
#
# This file contains (i) proprietary rules that were created, tested and certified by
# Sourcefire, Inc. (the "VRT Certified Rules") that are distributed under the VRT
# Certified Rules License Agreement (v 2.0), and (ii) rules that were created by
# Sourcefire and other third parties (the "GPL Rules") that are distributed under the
# GNU General Public License (GPL), v2.
#
# The VRT Certified Rules are owned by Sourcefire, Inc. The GPL Rules were created
# by Sourcefire and other third parties. The GPL Rules created by Sourcefire are
# owned by Sourcefire, Inc., and the GPL Rules not created by Sourcefire are owned by
# their respective creators. Please see http://www.snort.org/snort-team/ for a
# list of third party owners and their respective copyrights.
#
# In order to determine what rules are VRT Certified Rules or GPL Rules, please refer
# to the VRT Certified Rules License Agreement (v2.0).
#
#-------------
# LOCAL RULES
#-------------

alert tcp any any -> any any (msg:"Nmap Port Scan Detected"; flags:S; threshold: type threshold, track by_src, coun

```

- **Version Check:** Verified Snort version using terminal commands.



```
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd..

C:\Windows>cd..

C:\>cd Snort

C:\Snort>cd bin

C:\Snort\bin>snort -V

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20-WIN64 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

C:\Snort\bin>_
```

- **Running Snort in IDS Mode:** Documented commands and execution results.
- **Log Analysis:** Verified logs generated in portscan.log, including detection of an Nmap scan.



```
Priority Count: 3
Connection Count: 0
IP Count: 3
Scanned IP Range: 142.251.42.74:192.168.1.1
Port/Proto Count: 3
Port/Proto Range: 53:443

Time: 04/01-21:51:27.469518
event_ref: 5
192.168.1.12 -> 40.99.34.162 (portscan) Open Port
Open Port: 443

Time: 04/01-21:51:27.474274
event_ref: 5
192.168.1.12 -> 40.99.34.162 (portscan) Open Port
Open Port: 443

Time: 04/01-21:51:28.403806
event_ref: 5
192.168.1.12 -> 20.189.173.14 (portscan) Open Port
Open Port: 443

Time: 04/01-21:51:28.667999
event_ref: 5
192.168.1.12 -> 20.189.173.14 (portscan) Open Port
Open Port: 443

Time: 04/01-21:51:28.739581
event_id: 11
192.168.1.12 -> 184.26.54.122 (portscan) UDP Filtered Portsweep
Priority Count: 0
Connection Count: 30
IP Count: 7
Scanned IP Range: 184.26.54.122:239.255.255.250
Port/Proto Count: 7
Port/Proto Range: 53:3702
```

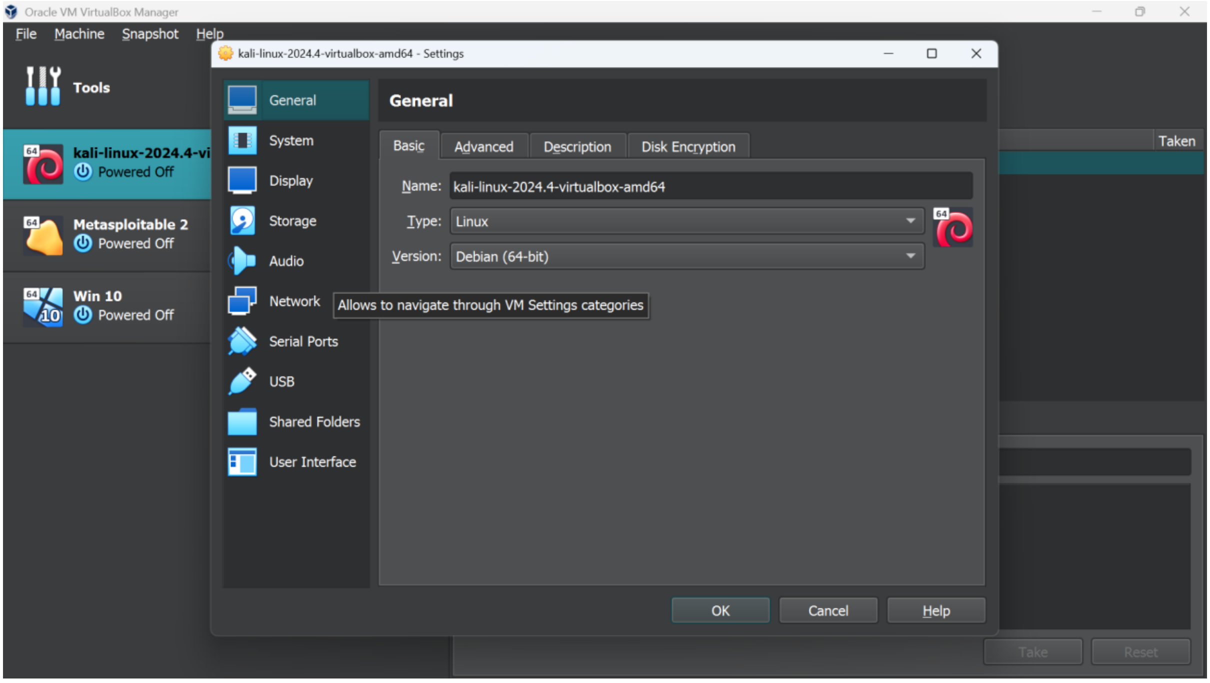
5. Kali Linux Setup and Penetration Testing

5.1 Overview

Kali Linux was used for penetration testing to assess network security.

5.2 Configuration Details

- **Kali Linux Installation:** Successfully set up and configured.



Oracle VM VirtualBox Manager

kali-linux-2024.4-vi — Powered Off

Metasploitable 2 — Powered Off

Win 10 — Powered Off

kali-linux-2024.4-virtualbox-amd64 - Settings

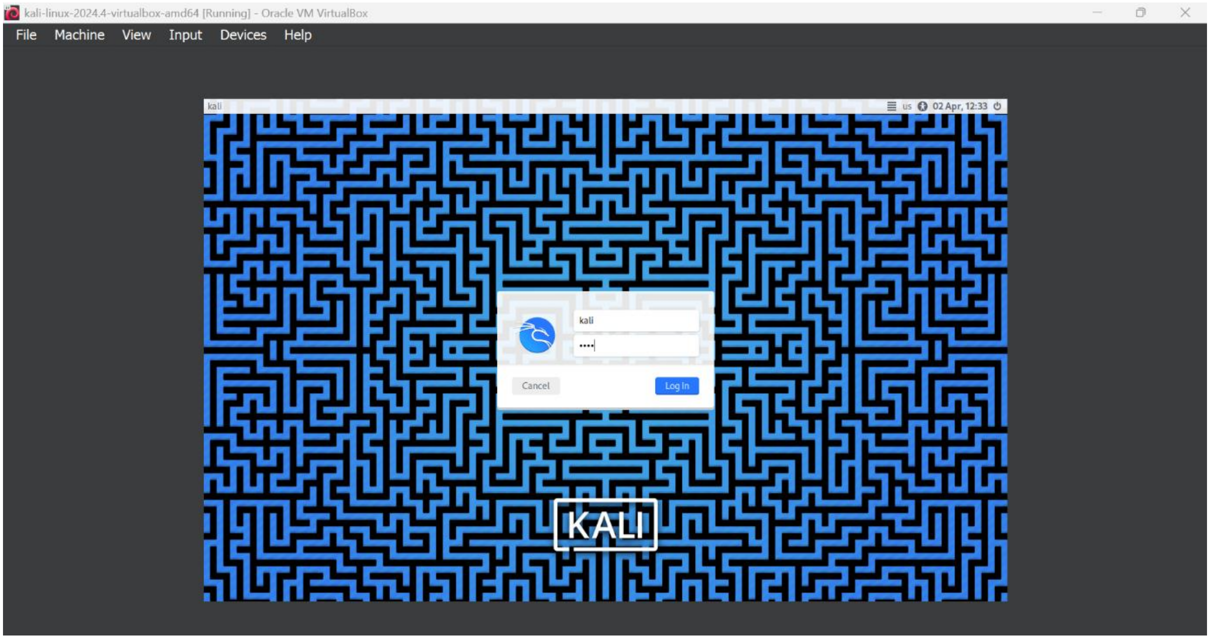
General

Name: kali-linux-2024.4-virtualbox-amd64

Type: Linux

Version: Debian (64-bit)

Allows to navigate through VM Settings categories



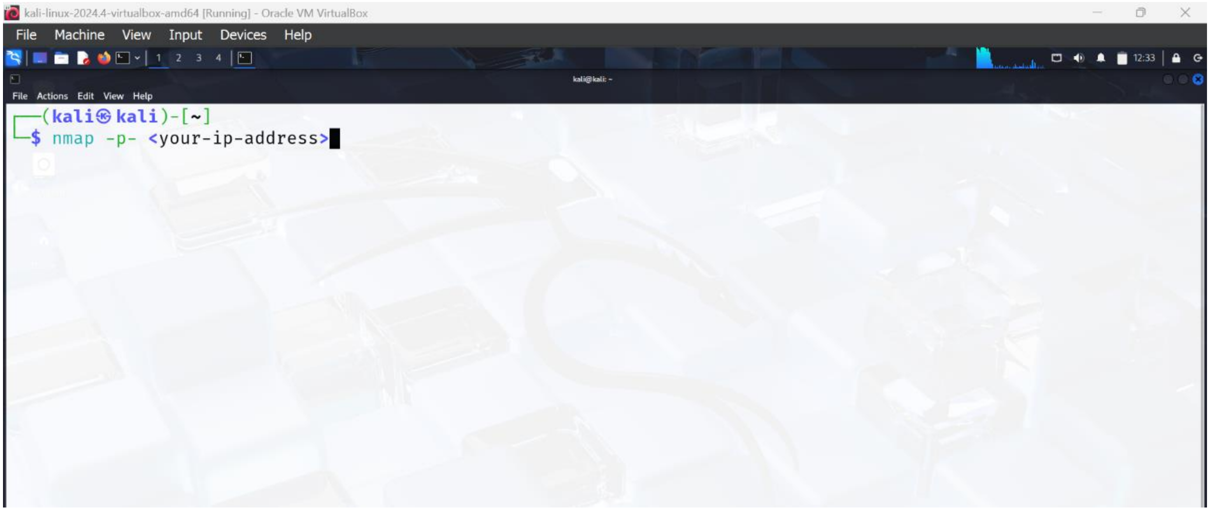
kali-linux-2024.4-virtualbox-amd64 [Running] - Oracle VM VirtualBox

kali

••••

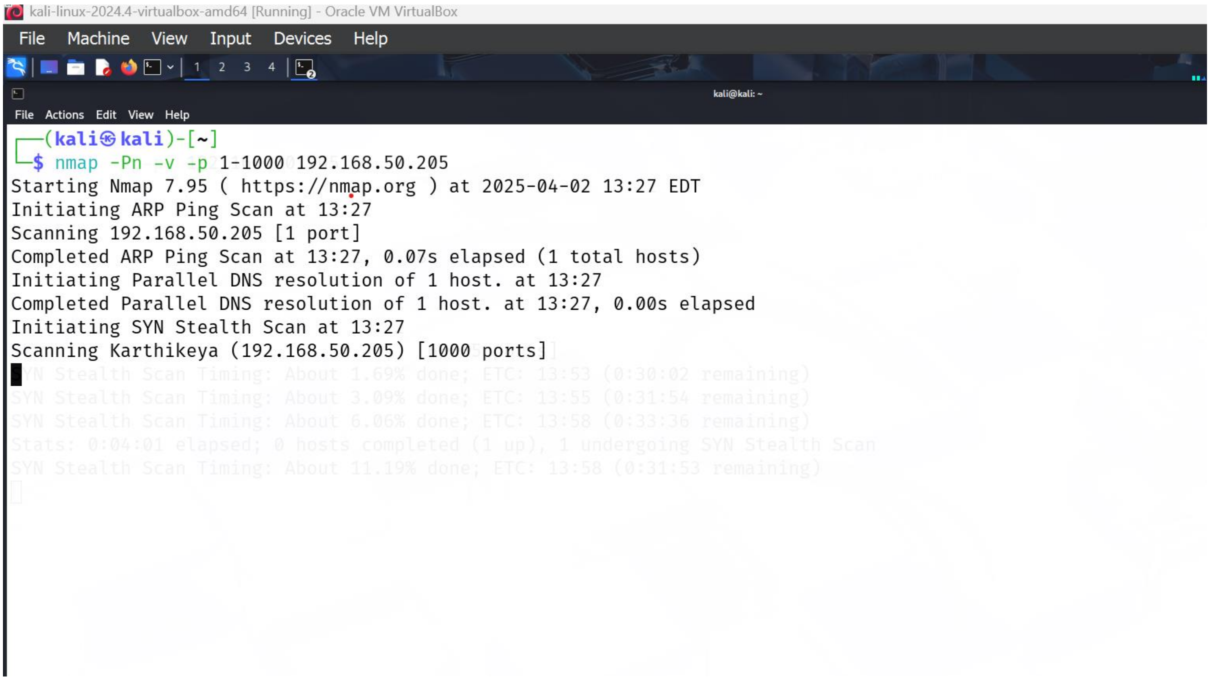
Continued….

- **Nmap Scan Execution:** Conducted an Nmap scan on a target system.


kali-linux-2024.4-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~
File Actions Edit View Help

```
┌──(kali㉿kali)-[~]
└─$ nmap -p- <your-ip-address>
```

- **Command Documentation:** Screenshots of commands used.


kali-linux-2024.4-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~
File Actions Edit View Help

```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -v -p 1-1000 192.168.50.205
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-02 13:27 EDT
Initiating ARP Ping Scan at 13:27
Scanning 192.168.50.205 [1 port]
Completed ARP Ping Scan at 13:27, 0.07s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 13:27
Completed Parallel DNS resolution of 1 host. at 13:27, 0.00s elapsed
Initiating SYN Stealth Scan at 13:27
Scanning Karthikeya (192.168.50.205) [1000 ports]
SYN Stealth Scan Timing: About 1.69% done; ETC: 13:53 (0:30:02 remaining)
SYN Stealth Scan Timing: About 3.09% done; ETC: 13:55 (0:31:54 remaining)
SYN Stealth Scan Timing: About 6.06% done; ETC: 13:58 (0:33:36 remaining)
Stats: 0:04:01 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 11.19% done; ETC: 13:58 (0:31:53 remaining)
```

- **Scan Results:** Analyzed scan output to identify open ports and vulnerabilities.

6. Incident Response and Findings

6.1 Security Event Detection

- Splunk alerts successfully triggered based on predefined queries.
- Snort detected unauthorized network scanning activity (Nmap scan).
- Python automation script provided timely log summaries via email.

6.2 Security Recommendations

- **Splunk Optimization:** Implement advanced correlation rules and dashboards.
- **Snort Enhancement:** Expand rule set to detect a wider range of attacks.
- **Network Hardening:** Limit open ports and monitor suspicious activities.
- **Incident Response Plan:** Establish a standardized incident response framework.

7. Conclusion

This report documents the security setup and its effectiveness in detecting security events. Based on the findings, recommendations have been provided to improve overall cybersecurity resilience. Continuous monitoring, log analysis, and proactive security measures are essential for maintaining a robust security infrastructure.