Red vs Blue

Design and Implementation of a Virtual Enterprise IT Infrastructure

Final Project Documentation

By
Team C
Pranay Garapati

Swarupa Jeedimetla

Dheeraj Kumar Ammineni

Manisharan Chakali

Brahmendra Chowdary Ponduri

A Master's Research Project Submitted to
The School for Professional Studies
in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Cybersecurity

Instructor
Syam Sai Siddabhattula

Saint Louis University
December 2025

Powered by **Ears Up Cyber Security LLC**

Table of Contents

Final Project Documentation

Introduction

This was a project that created and secured a complete virtualized enterprise network setting based on Proxmox VE, VyOS routing, pfSense firewalling, inner-network segmentation, and Wazuh monitoring. Both the Red Team's offensive testing and the Blue Team's defensive monitoring were supported by the environment. Within a few weeks, the team has configured the base infrastructure, segmented the internal network, installed security tooling, and conducted realistic attack-and-defense scenarios.

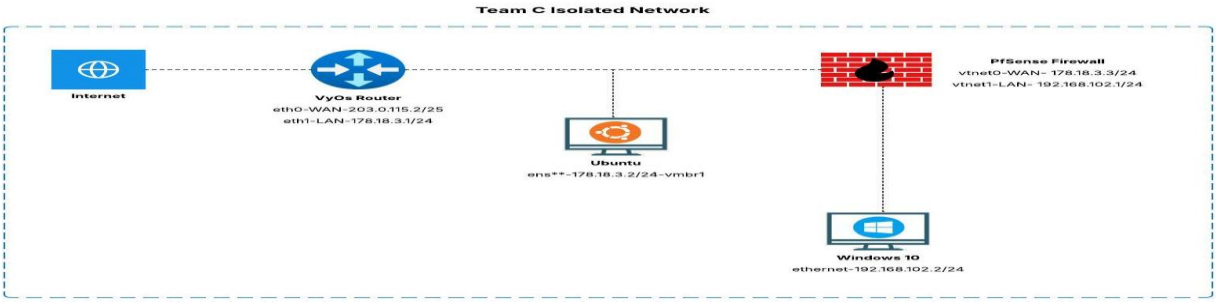
Virtualization Setup

The first task was to prepare the groundwork of our virtual laboratory environment with the help of Proxmox Virtual Environment (VE). We accomplished it through the installation of a complete virtual network, which comprised WAN and LAN bridges (Ford, 2019). This setup included:

- VyOS Router in the public WAN IP (203.0.115.2/25), and private LAN IP (178.18.3.1/24).
- Ubuntu on the LAN bridge that had the IP of 178.18.3.2/24.
- PfSense firewall with a WAN facing IP (178.18.3.3/24) and internal LAN IP (192.168.102.1/24).
- Windows 10 VM, with the internal IP 192.168.102.2/24, which we configured to work behind and get the PfSense firewall protection.

Figure 1

Network Diagram



Proxmox Virtual Environment

Proxmox VE is an open-source and full-fledged server management software to virtualize businesses. It is a program that develops and operates virtual machines (VMs) (Ford, 2019).

Figure 2

Proxmox VE Login

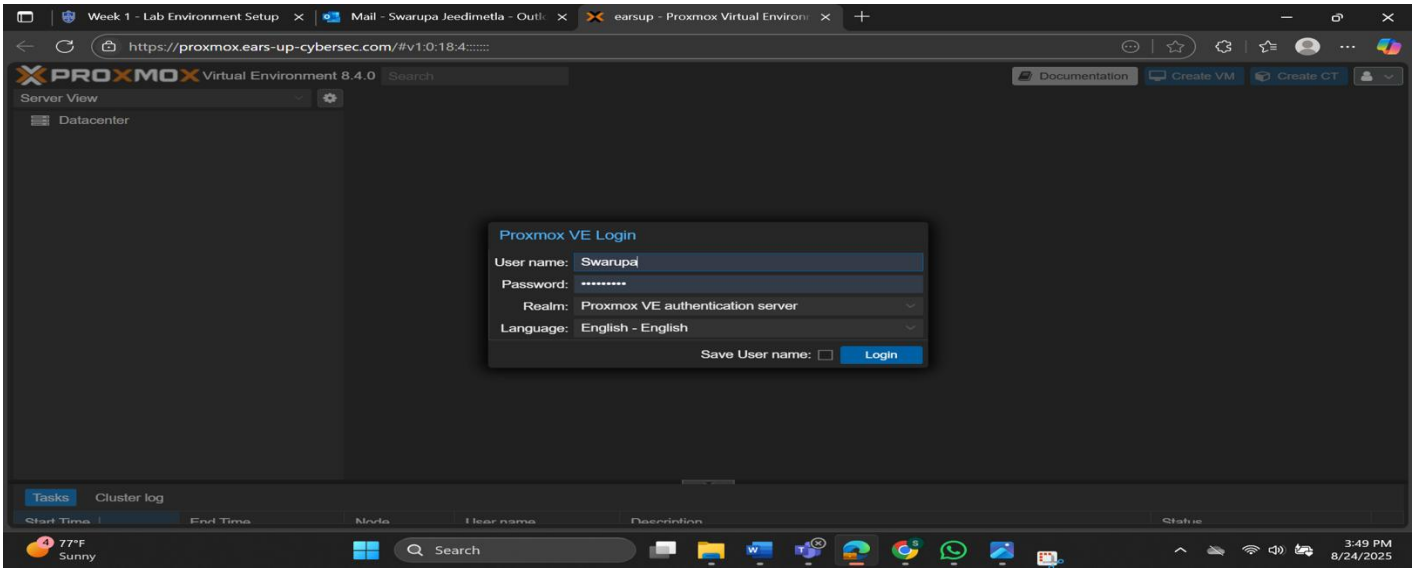
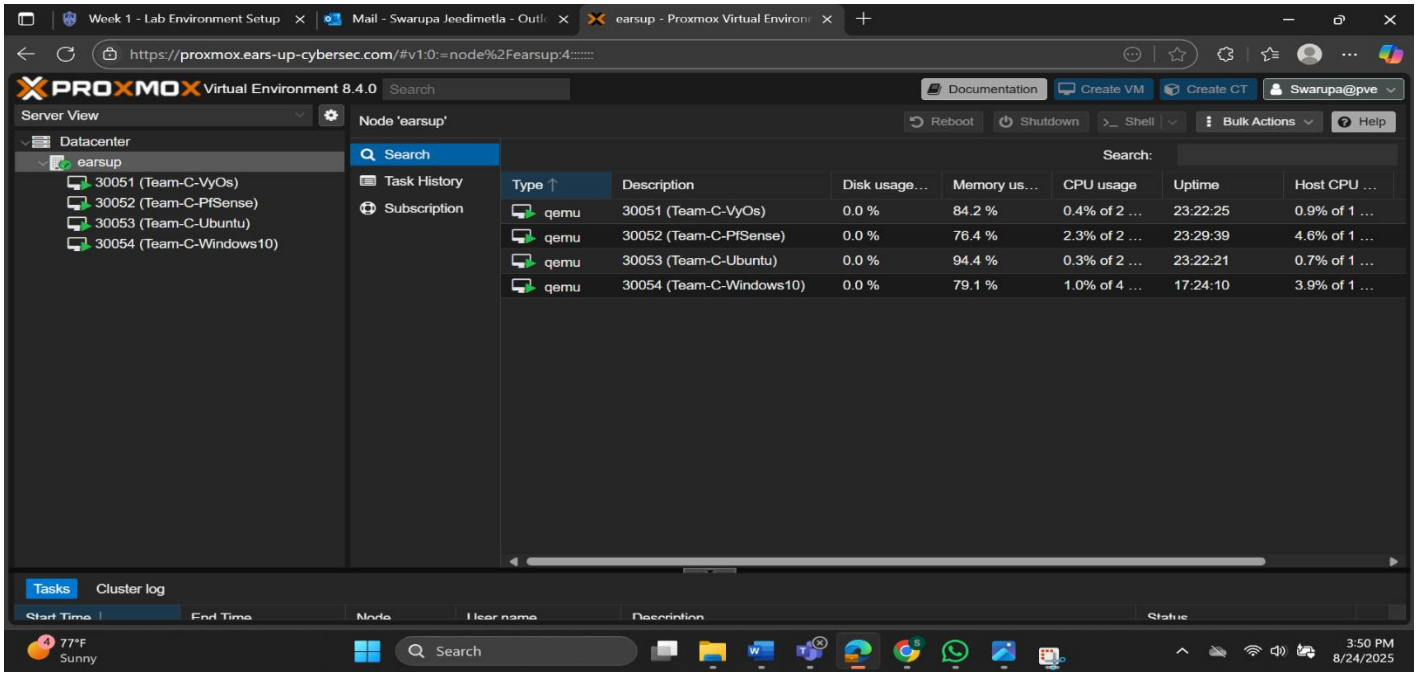


Figure 3

Proxmox Dashboard

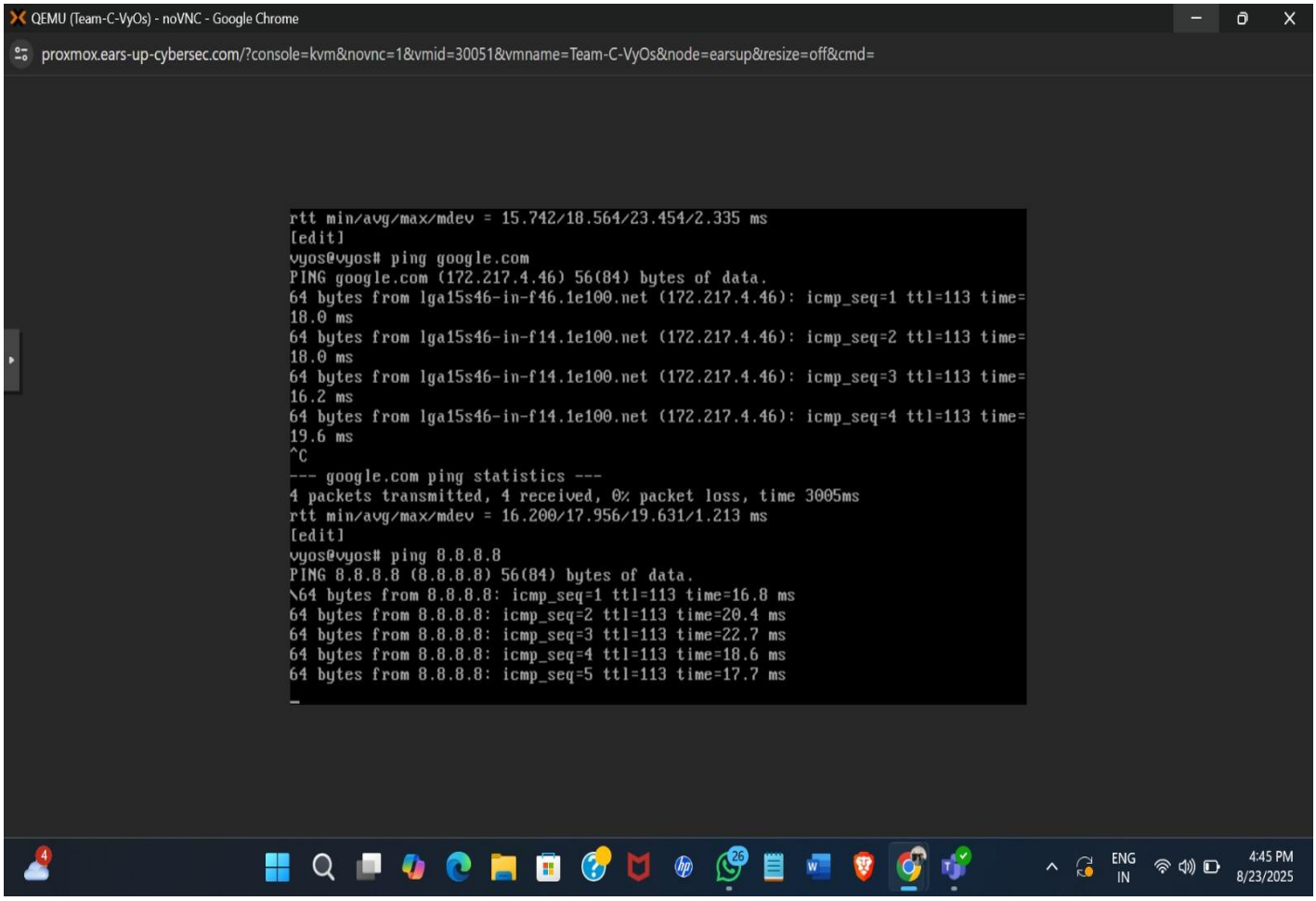


VyOS Router Configuration

VyOS router was configured as the central routing and NAT device with the WAN interface configured as 203.0.115.2/25 and a default route configured with 203.0.115.1 and the LAN interface as 178.18.3.1/24. The 8.8.8.8 and 1.1.1.1 DNS servers were configured to allow name resolution. To offer access to the internet to internal hosts, a Source NAT rule was established to mask all the traffic originating from the WAN interface eth0 and the 178.18.3.0/24 LAN out. Ping 8.8.8.8 tests confirmed that the internet was reachable, and ping google.com tests confirmed that DNS resolution was operating properly (VyOS, 2024).

Figure 4

Connection Verification



pfSense Firewall Configuration

To separate internet and internal traffic, we connected vtnet0 to the WAN (vmbr1) and vtnet1 to the LAN (vmbr2). In the console, the LAN IP was configured as 192.168.102.1/24 with the gateway being blank, and IPv6 was ignored. The range of the LAN DHCP server was set to 192.168.102.100 - 192.168.102.200. Using pfSense (WAN 178.18.3.3/24, LAN 192.168.102.1/24), we were able to ping the Windows VM (192.168.102.2) and 8.8.8.8 and could confirm being connected.

Figure 5

pfSense IP address

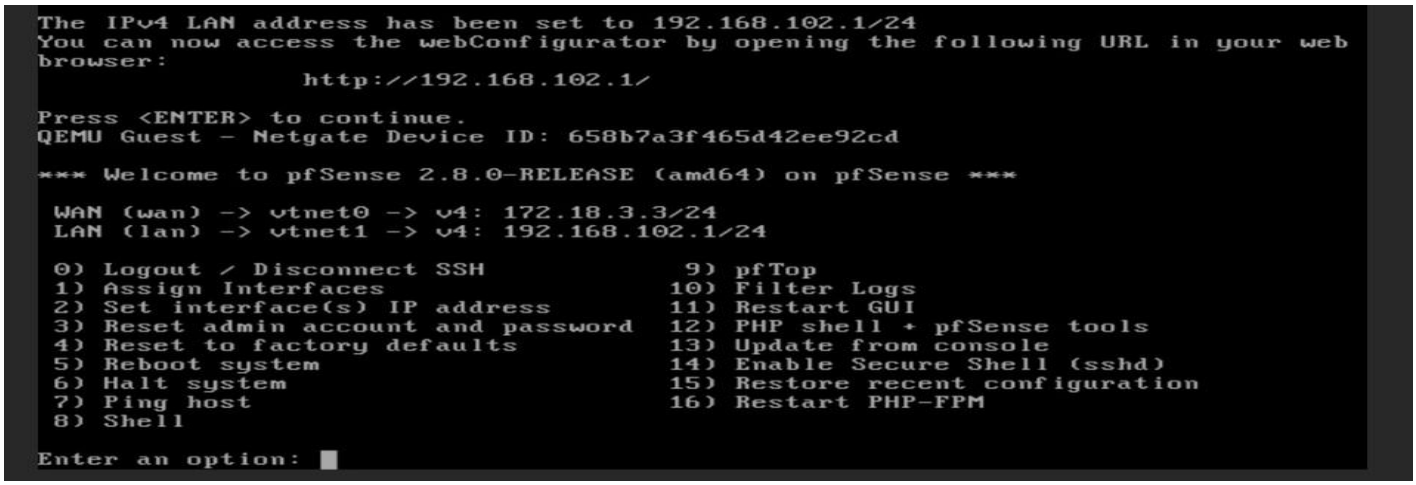


Figure 6

Connection Verification

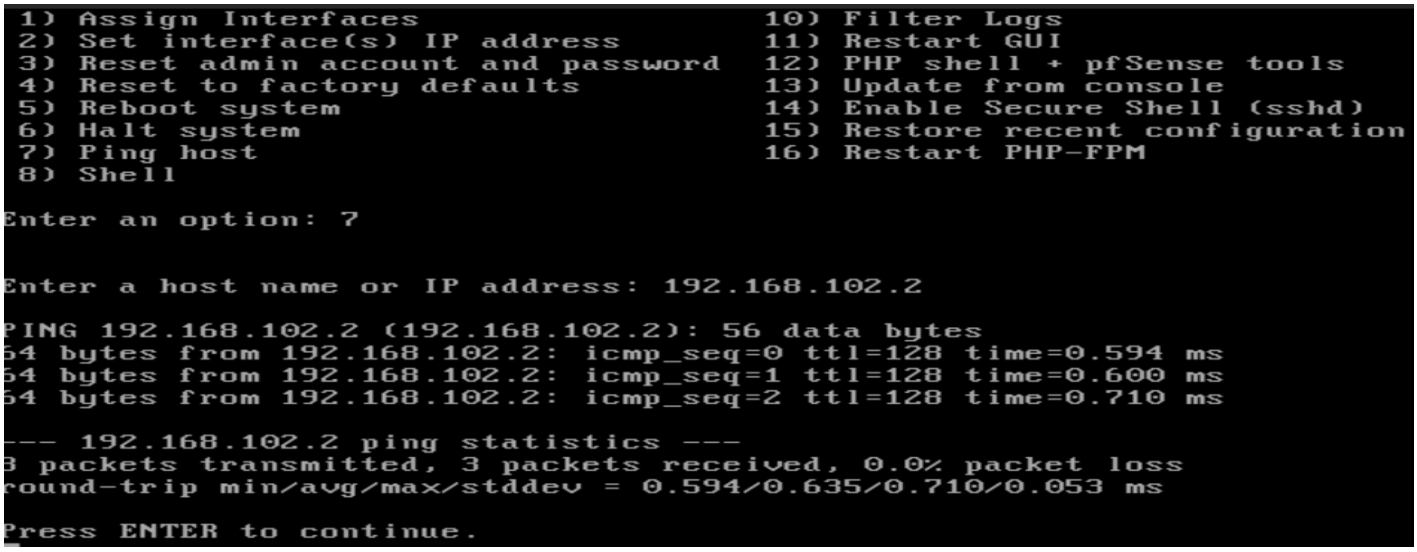
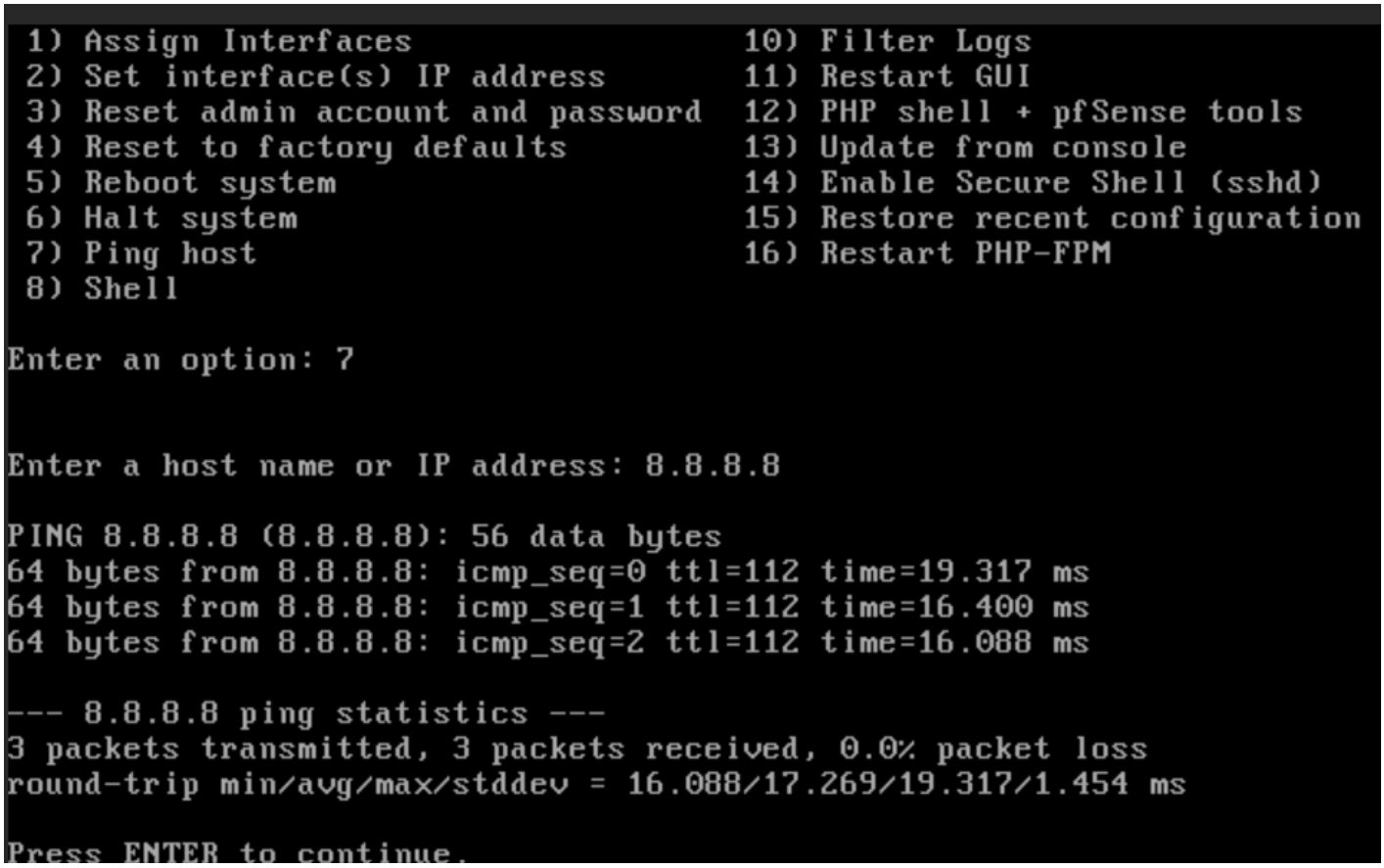


Figure 7

Connection Verification



Windows VM Configuration

The Windows VM was configured with a static IP (192.168.102.2), subnet mask (255.255.255.0), default gateway (192.168.102.1), and DNS of Google public DNS (8.8.8.8) to allow the system to resolve the name. This manual setup also enabled Windows VM to communicate with the pfSense gateway and the rest of the systems on the network. We confirmed that we had internet connectivity by being able to ping the pfSense LAN gateway (192.168.102.1) and 8.8.8.8 to make sure that we can use the internet and DNS resolution by going to Google.com, which confirmed that the Windows VM can communicate with the local environment and the internet.

Figure 8

IPv4 Properties

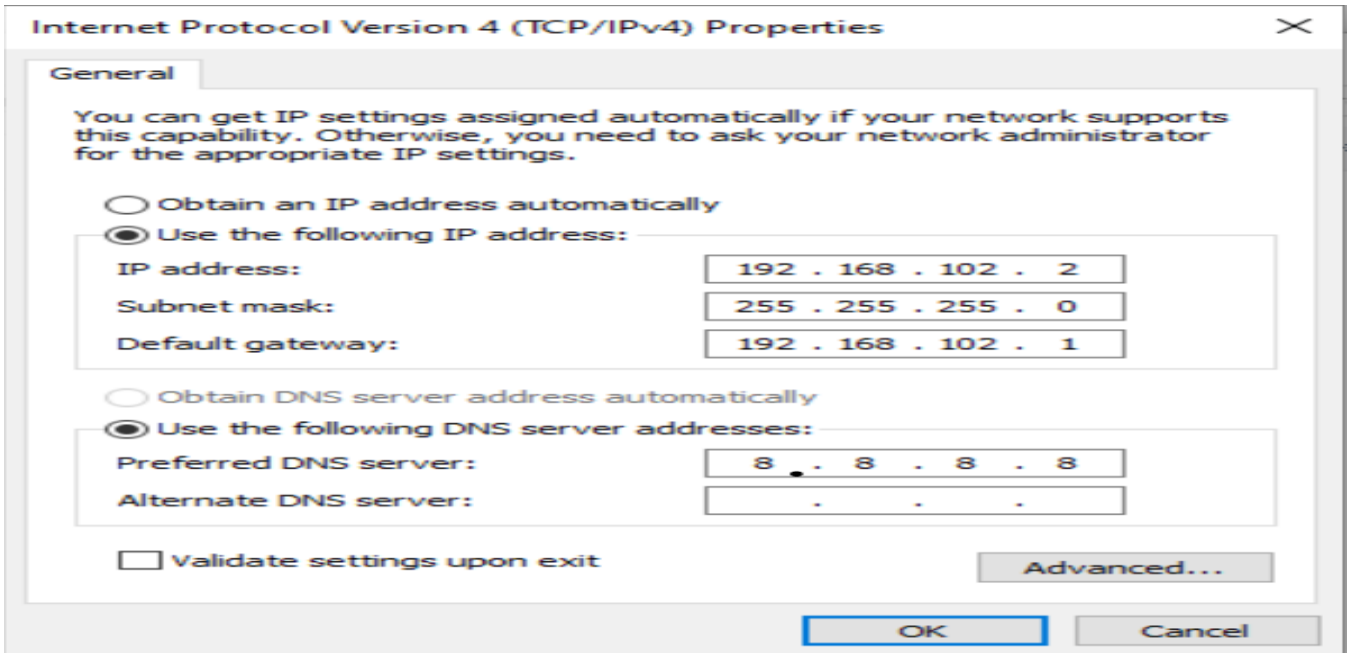


Figure 9

IP Settings

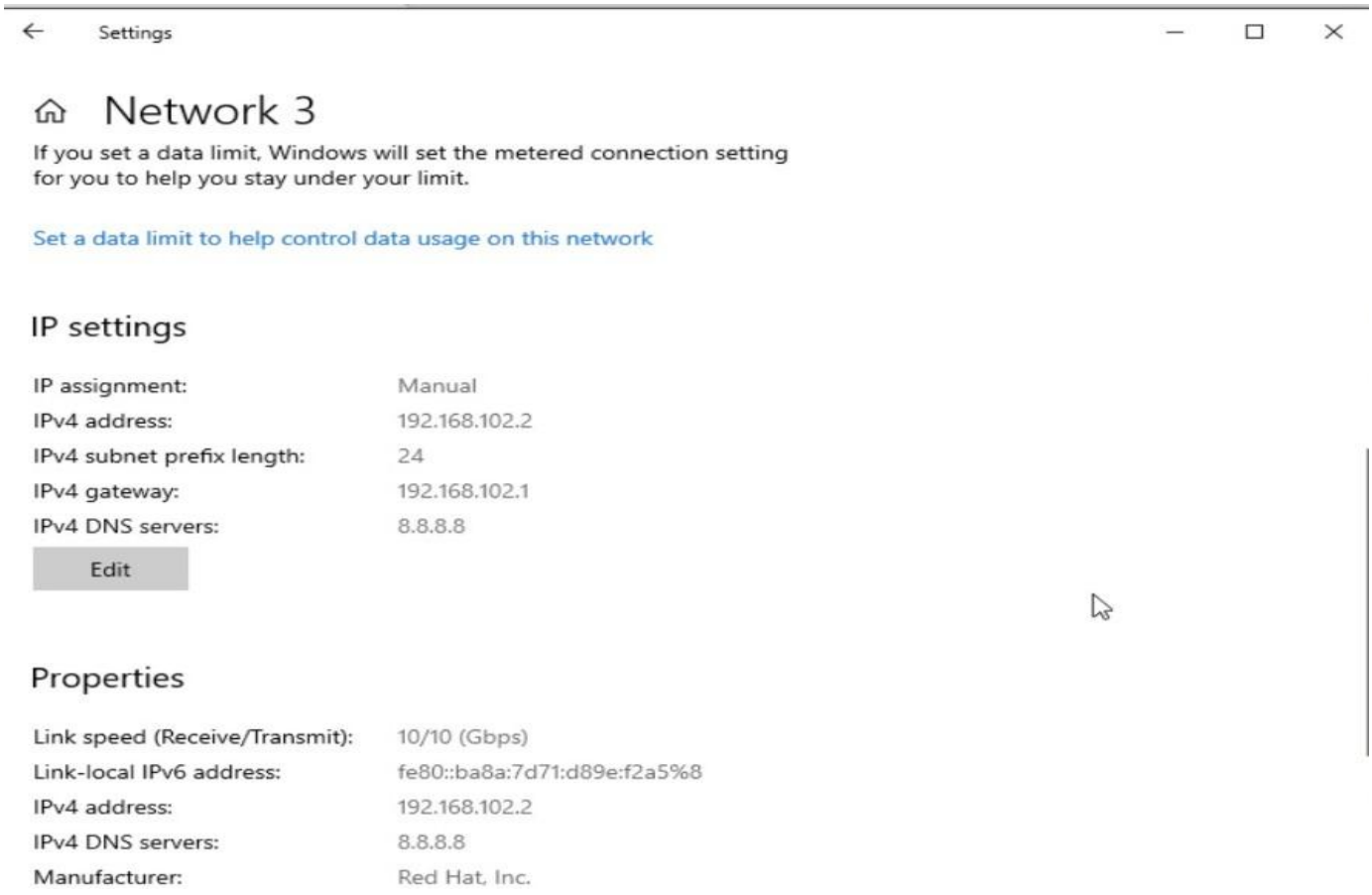


Figure 10

Windows IP Address

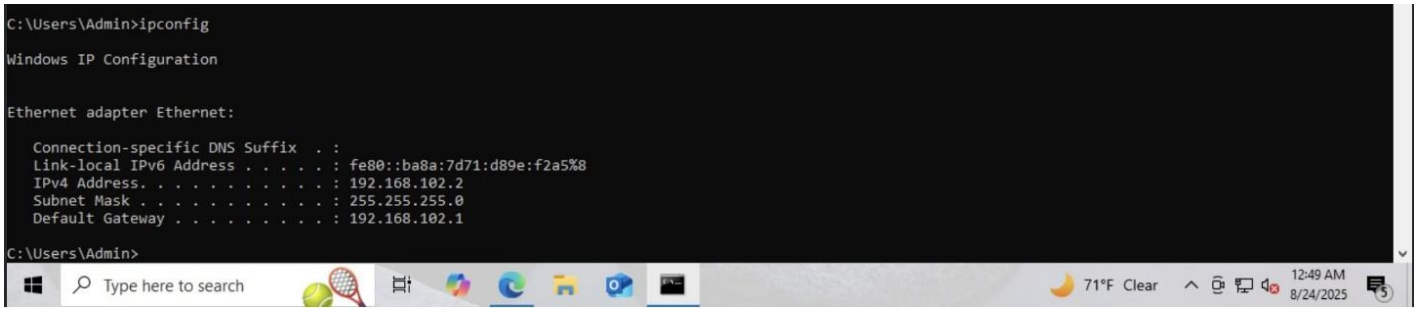
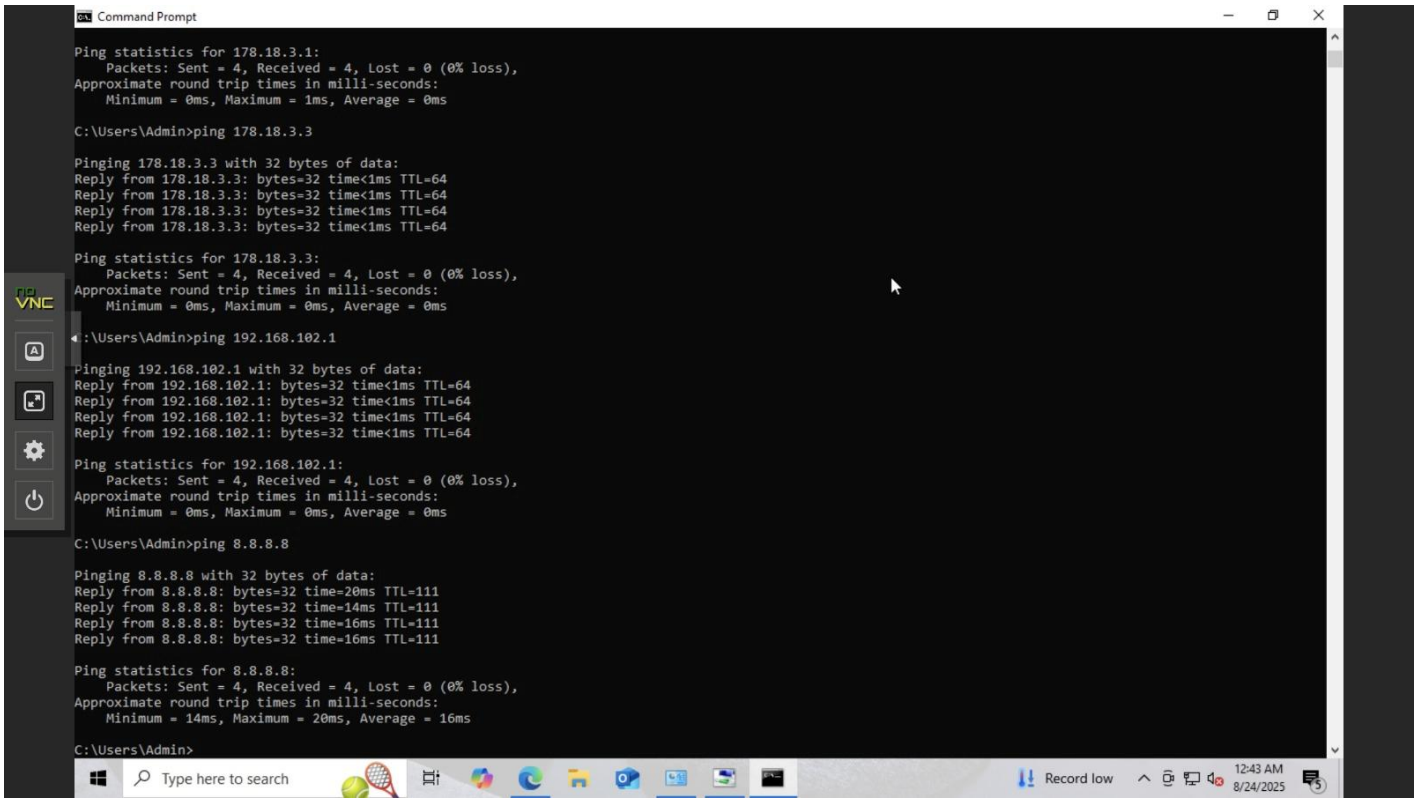


Figure 11

Connection Verification



Ubuntu VM Configuration

We then cleared the old addresses on the Ubuntu VM with sudo ip addr flush dev ens18 (substituting eth0 with ens18, which is our network device of the interface) to put the interface in a state where it was

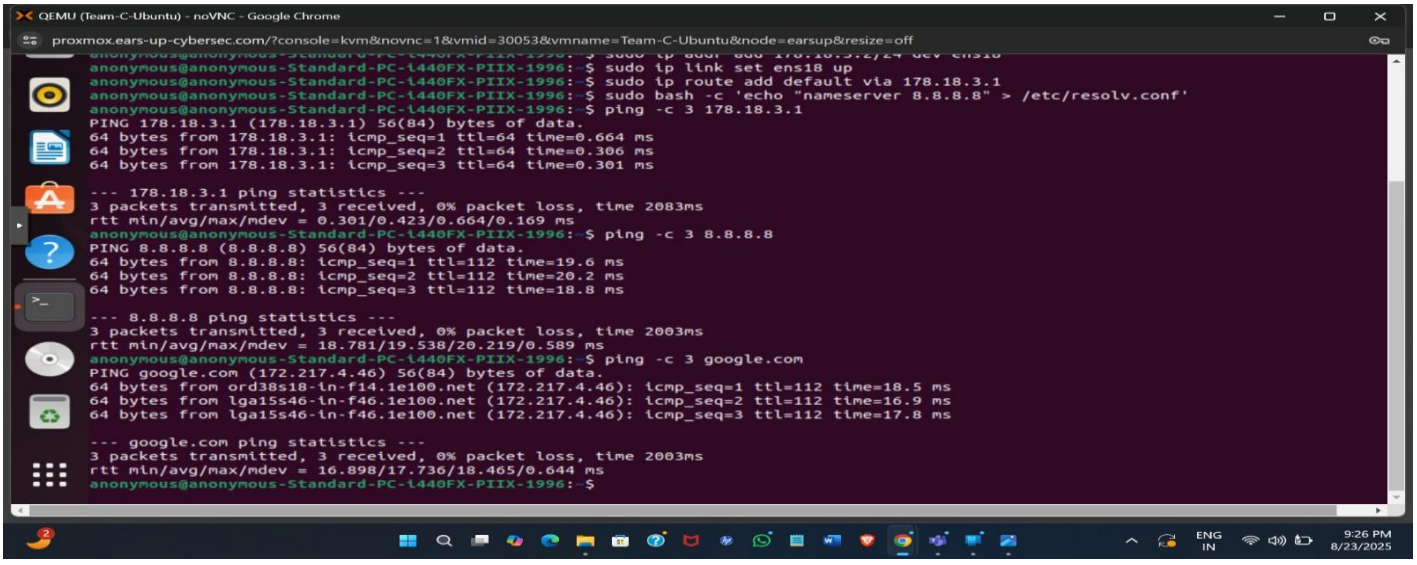
prepared to start with a new IP. At that point, we gave the VM 178.18.3.2/24 sudo ip addr add 178.18.3.2/24 dev ens18 and brought the interface up sudo ip link set ens18 up. VyOS router configured as the default gateway by sudo ip route add default via 178.18.3.1, which makes it available to the other networks and the internet. To set DNS with Google Public DNS, we used sudo bash -c echo nameserver 8.8.8.8 /etc/resolv.conf. Ping of the router (178.18.3.1), 8.8.8.8, and Google.com was successfully done, and it confirmed that there was proper access to the local and internet with domain name resolution.

Figure 12

Ubuntu IP address

Figure 13

Connection Verification



Firewall & Router Setup

VyOS and pfSense Connectivity Check

We configured VyOS router by setting the WAN interface (eth0) with the IP 203.0.115.2/25 and default gateway 203.0.115.1 and the LAN interface (eth1) with the IP 178.18.3.1/24. To configure a NAT and add DNS servers (8.8.8.8 and 1.1.1.1) we added a masquerade rule for the subnet 178.18.3.0/24 and redirected traffic flowing over eth0. Once the configuration has been saved and the configuration was verified, we ensured that we could ping the destination successfully to verify that we had internet connectivity and that we could resolve DNS addresses (Samuel, 2018).

Figure 14

Configuration of VyOS Router

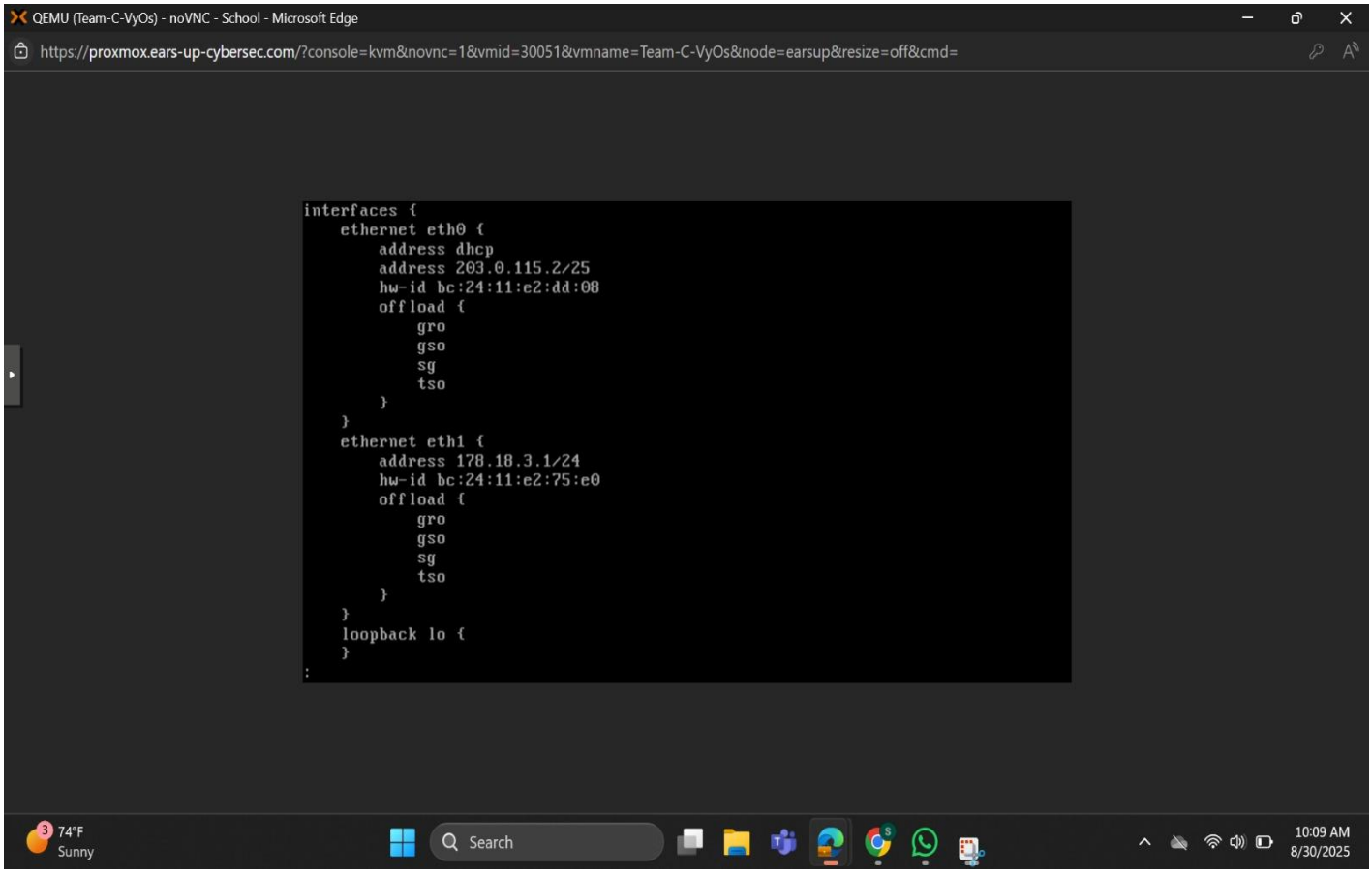


Figure 15

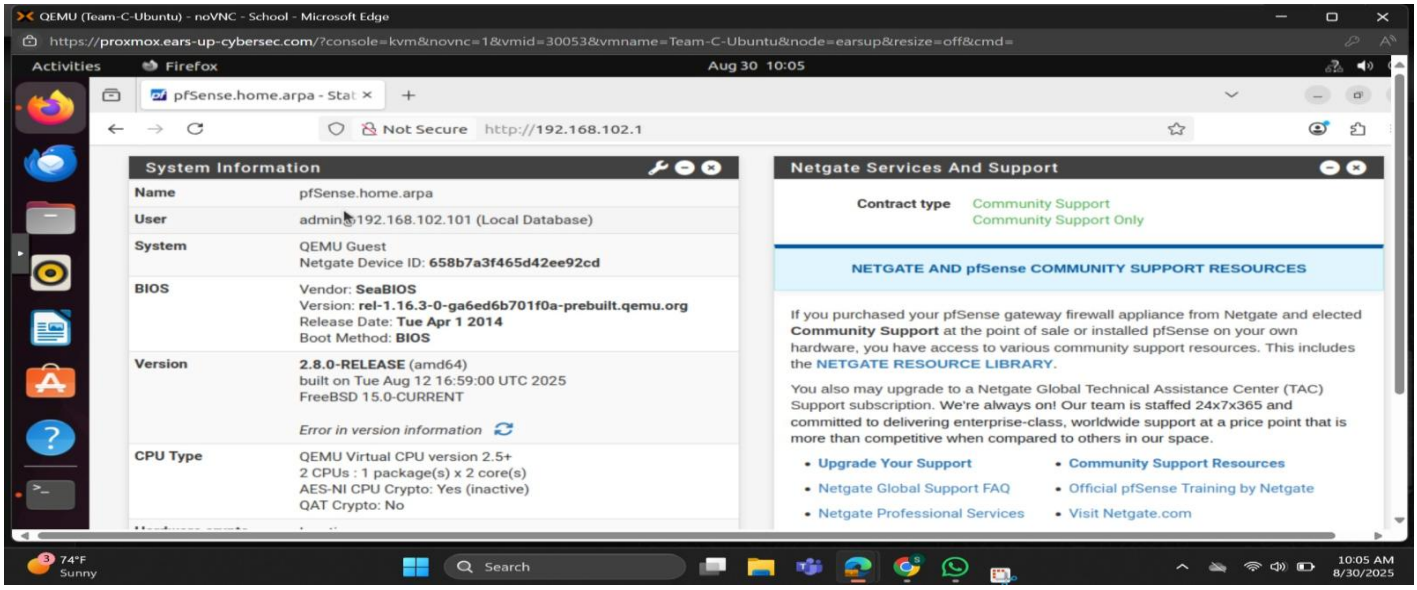
Interfaces of VyOS Router

```
eth1             178.18.3.1/24         bc:24:11:e2:75:e0  default    1500  u/u
lo               127.0.0.1/8           00:00:00:00:00:00  default   65536  u/u
                 ::1/128
vyos@vyos:~$ sudo ip addr del 192.168.102.101/24 dev eth0
vyos@vyos:~$ ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP grou
p default qlen 1000
    link/ether bc:24:11:e2:dd:08 brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    altname ens18
    inet 203.0.115.2/25 brd 203.0.115.127 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::be24:11ff:fee2:dd08/64 scope link
       valid_lft forever preferred_lft forever
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address        MAC                VRF         MTU  S/L    Descripti
on
---------        ----------        ---                ---         ---  ---    ---------
---
eth0             203.0.115.2/25    bc:24:11:e2:dd:08  default    1500  u/u
eth1             178.18.3.1/24     bc:24:11:e2:75:e0  default    1500  u/u
lo               127.0.0.1/8       00:00:00:00:00:00  default   65536  u/u
                 ::1/128
vyos@vyos:~$
```

We adjusted the pfSense VM by configuring the WAN (pfSense) into vtnet0 (vmbr1) and configuring the LAN (pfSense) into vtnet1 (vmbr2) and setting LAN IP to 192.168.102.1/24 and DHCP server on to 192.168.102.100-192.168.102.200. Once the WAN was set to 178.18.3.3/24, we verified connectivity by being able to ping the windows VM (192.168.102.2) and the 8.8.8.8. Then, we were able to connect to the pfSense Web GUI using the Ubuntu VM, which ensured that the network was correctly integrated, and it was possible to manage it.

Figure 16

pfSense Web GUI open from the Ubuntu VM



Connectivity Testing Check List

Table 1

Connectivity Checklist

Source to Destination	Expected Result	Status (UP/Down)
VyOS to Internet	Ping 8.8.8.8 succeeds	**Up**
VyOS to Ubuntu	Ping 178.18.3.2 succeeds	**Up**
VyOS to pfSense	Ping 178.18.3.3 succeeds	**Up**
Ubuntu to Internet	Ping 8.8.8.8 Succeeds	**Up**
Ubuntu to VyOS	Ping 178.18.3.1 succeeds	**Up**
Ubuntu to pfSense	Ping 178.18.3.3 succeeds	**Up**
pfSense to Internet	Ping 8.8.8.8 succeeds	**Up**
PfSense to VyOS	Ping 178.18.3.1 succeeds	**Up**
pfSense to Ubuntu	Ping 178.18.3.2 succeeds	**Up**

Firewall Security & Controlled Exposure with pfSense & Windows 10

Creating Non-Admin User and Enabling Services

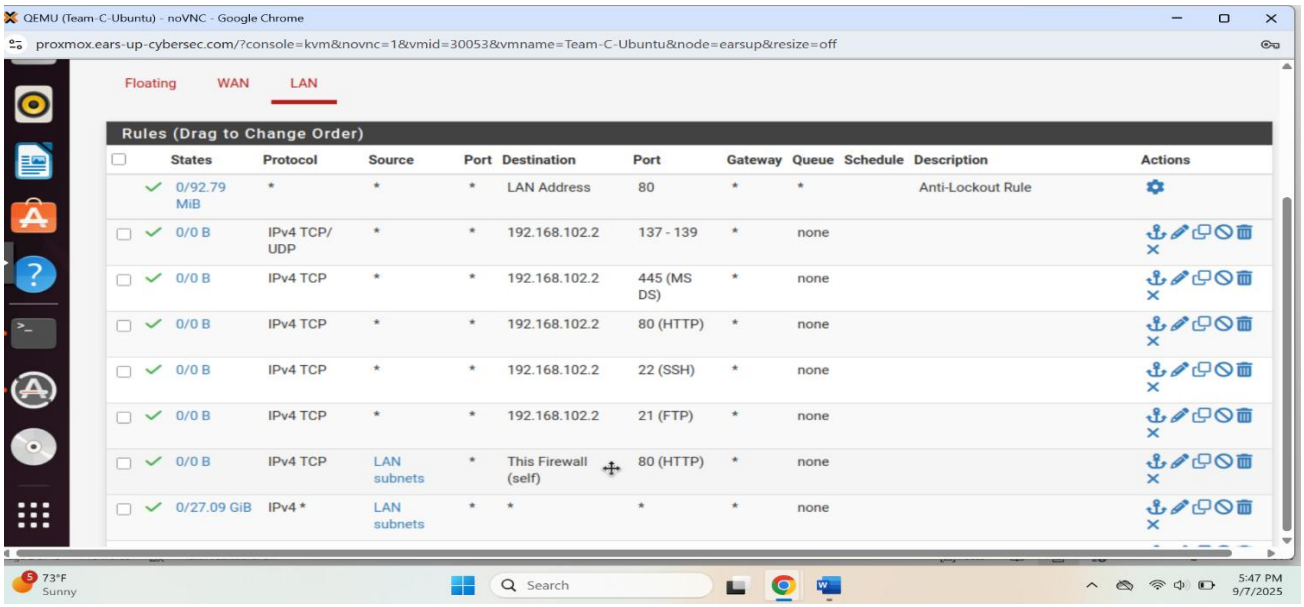
Windows 10 VM has been configured with a static Ip 192.168.102.2, and the connection has been verified using ipconfig and ping 8.8.8.8. The default access was made by one of the non-admin users named Jimmy. Major services were enabled and tested, FTP through IIS Server (port 21), SSH through OpenSSH Server, SMB with shared folder, NetBIOS through TCP/IP, and an HTTP server through Python on ports 80 and all the ports that were evaluated to be correctly functioning.

pfSense Firewall Rules

The Windows 10 VM was permitted to access the LAN firewall rules and WAN NAT on pfSense (192.168.102.2) through FTP, SSH, HTTP, SMB, and NetBIOS. Ubuntu tests were performed.

Figure 17

pfSense Rules



Connectivity and Ports Testing Checklist

Connection tests showed that pfSense could connect with Windows (192.168.102.2), windows could connect with pfSense via the LAN/WAN and the internet and that all services were active including FTP, SSH, HTTP, SMB, and NetBIOS.

Table 2

Connectivity Checklist

Source to Destination	Expected Result	Status (UP/Down)
PfSense to Windows	Ping 192.168.102.2 succeeds	**Up**
Windows to PfSense (WAN)	Ping 178.18.3.3 succeeds	**Up**
Windows to PfSense (LAN)	Ping 192.168.102.1 succeeds	**Up**
Windows to Internet	Ping 8.8.8.8 succeeds	**Up**

Table 3

Ports Testing

Ports	Status (UP/Down)
FTP (21)	**Up**
SSH (22)	**Up**
HTTP (80)	**Up**
SMB (445)	**Up**
NetBIOS (137,138,139)	**Up**

Wazuh & Sysmon Configuration

Ubuntu-Wazuh Manager Deployment

Ubuntu VM was then updated and the Wazuh Manager installed, and its service monitored as active by systemctl status wazuh-manager. Wazuh web dashboard was opened, default credentials were used to successfully log in, and everything was deployed, the dashboard was available.

Figure 18

Wazuh Dashboard in Ubuntu

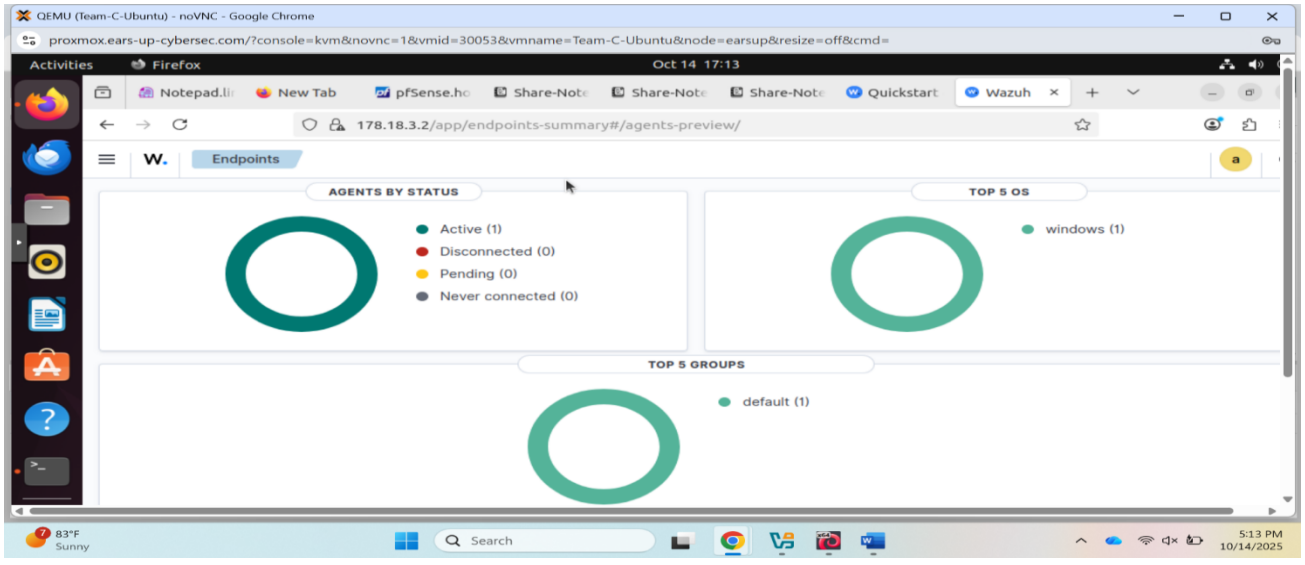
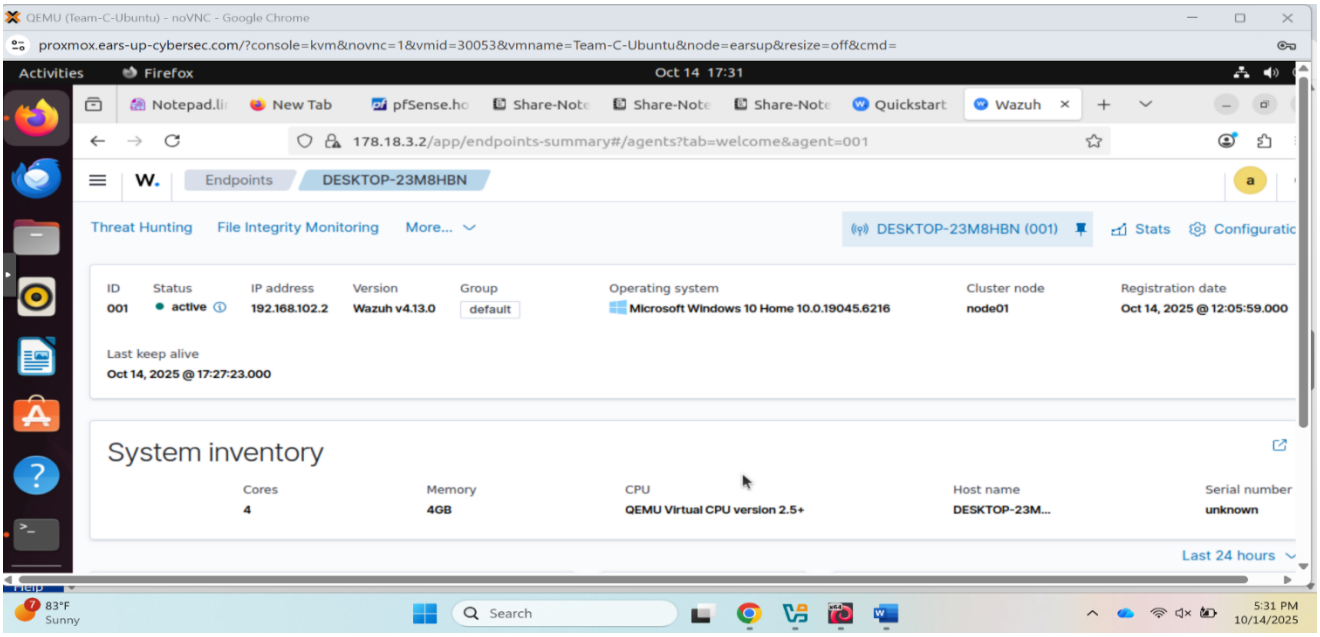


Figure 19

Wazuh Dashboard in Ubuntu

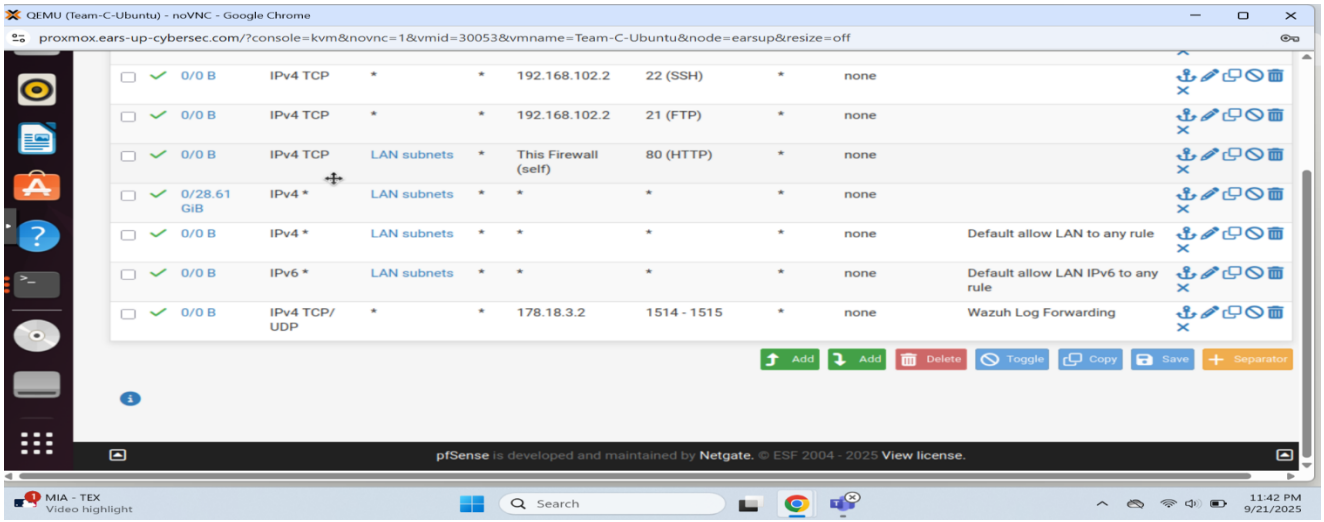


pfSense Firewall Configuration

The Web GUI was used to set the LAN firewall rules in pfSense that it was permitted to pass Wazuh logs to the Ubuntu VM that it accepts TCP/UDP traffic on port 1514 and 1515 to the Wazuh Manager. The rules were saved, accessed and verified as live in the firewall logs.

Figure 20

pfSense Rules



Windows 10 Wazuh Agent & Sysmon setup

Wazuh Agent was installed and configured to communicate with the Wazuh Manager on the Windows 10 Ubuntu, and the service was observed to be running. Sysmon was deployed using a custom XML configuration and the ossec.conf was modified to scan the System and Security logs. When reconnecting agent to manager once again the connection was verified in the Wazuh Dashboard and Sysmon events were generated and recorded in real time to be checked.

We configured Sysmon and used a custom XML configuration file to allow detailed logging of relevant security events, such as process creation, network connections, registry changes, and file activity, and tuned include/exclude rules in the XML and then ensured that the configured events were being logged properly in the Sysmon Operational log and being sent to our monitoring platform (Rodríguez, 2017).

Figure 21

Sysmon

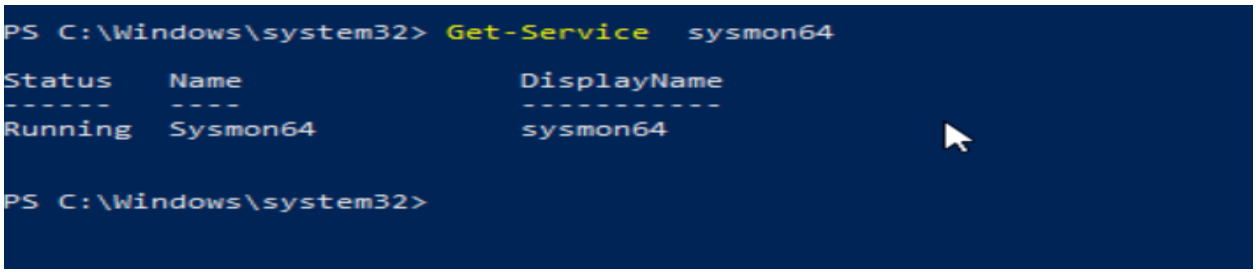


Figure 22

Event Viewer

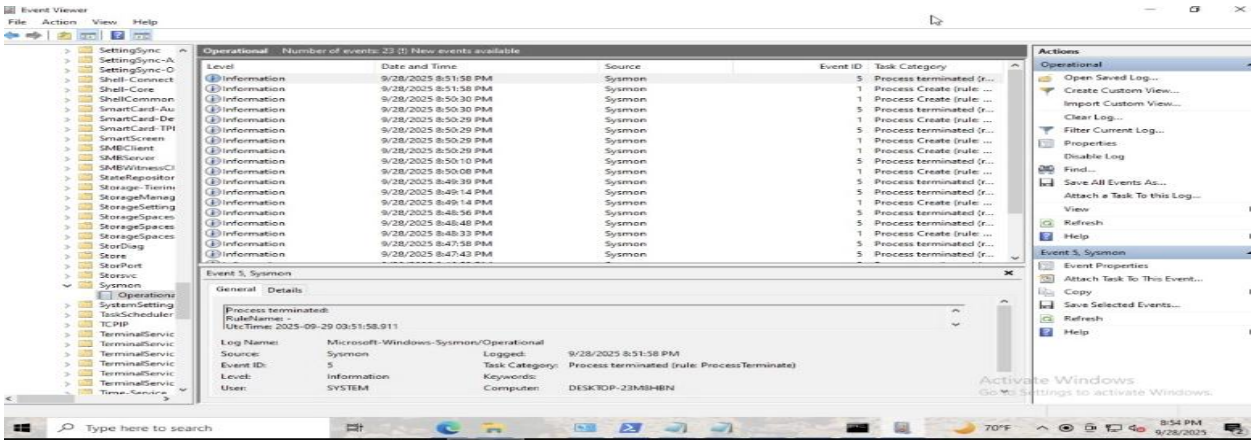


Figure 23

XML file

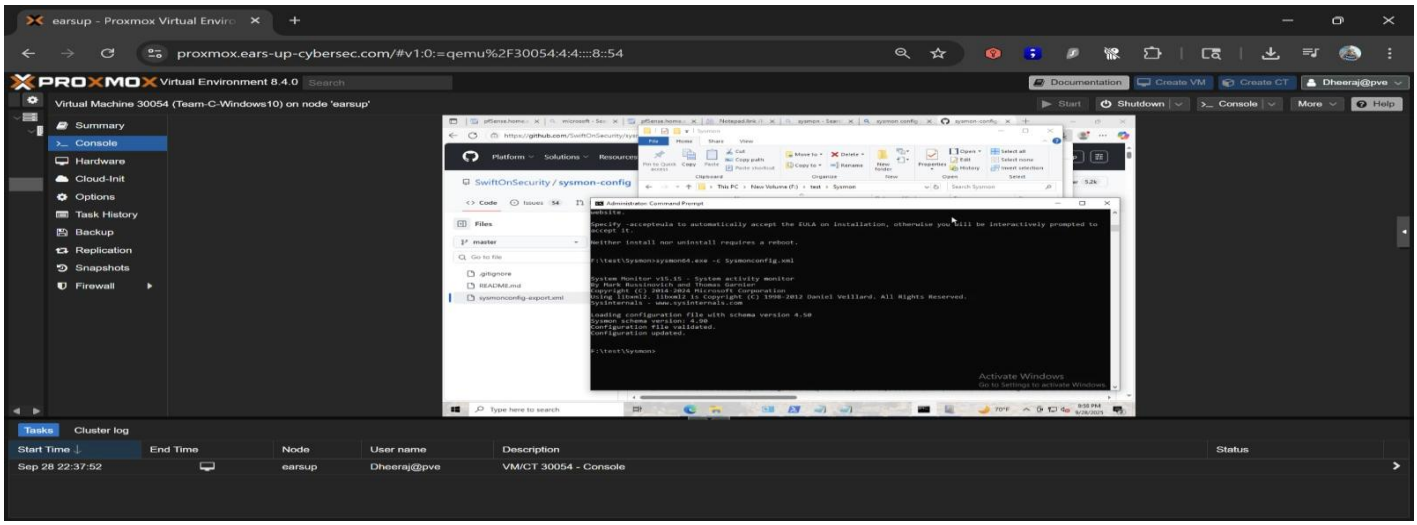


Figure 24

XML File

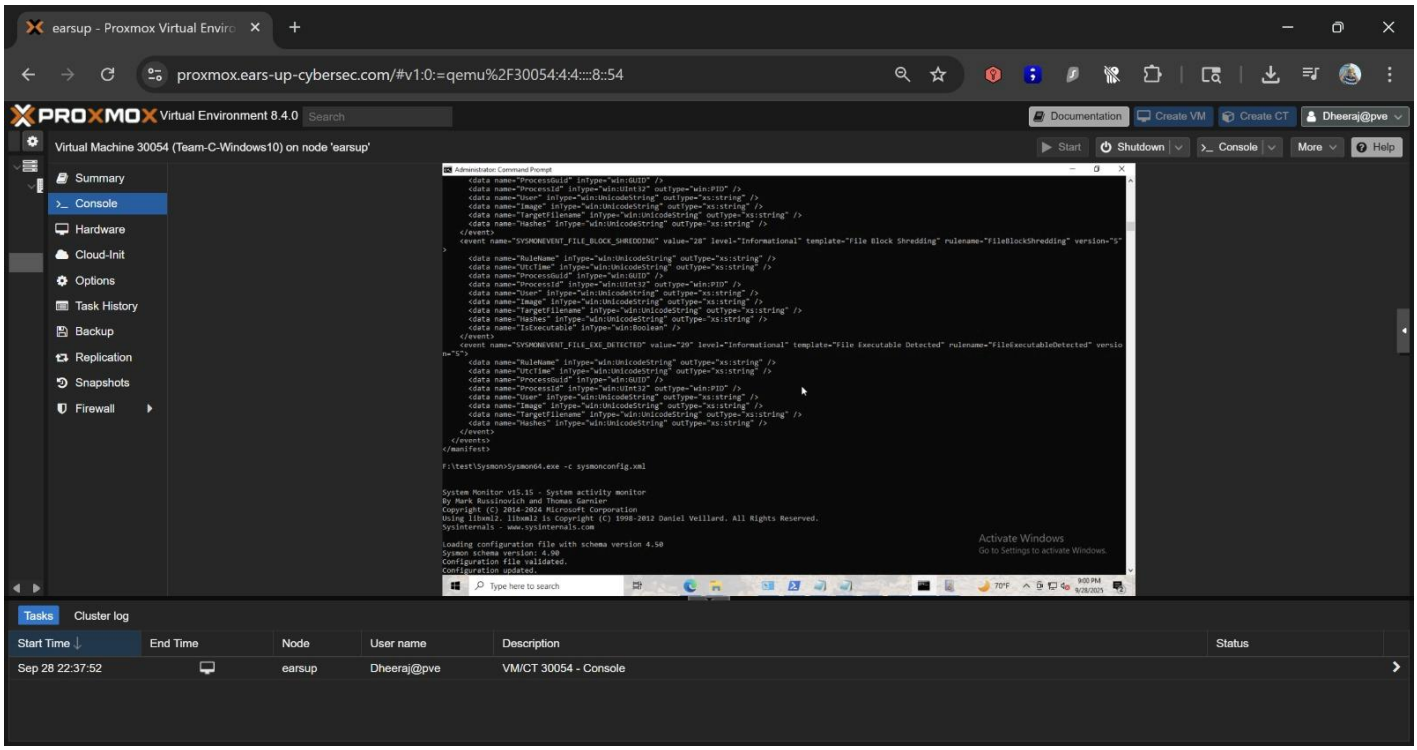
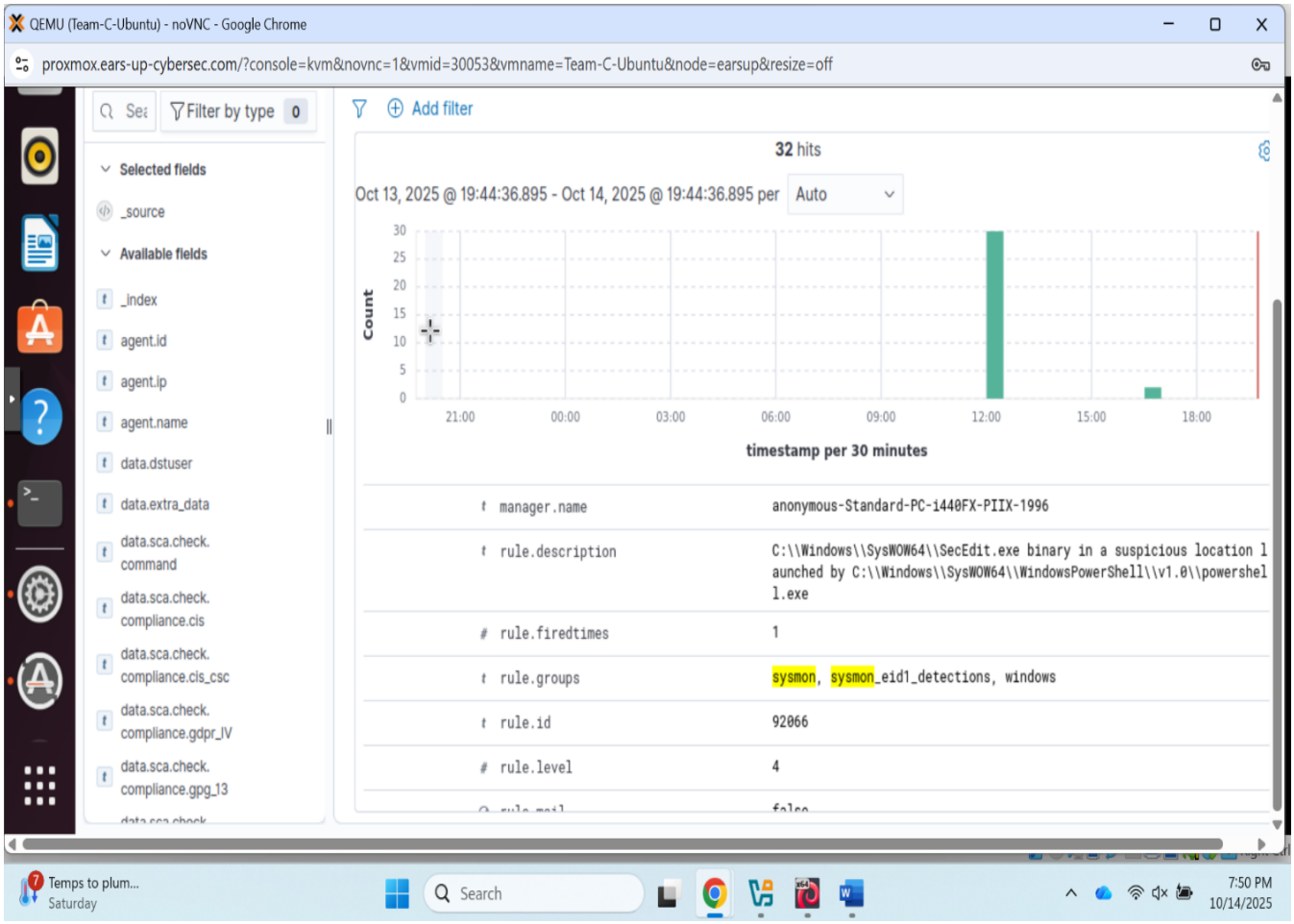


Figure 25

Wazuh Alerts



Windows 10 Home and Ubuntu Hardening

Windows 10 Home was safeguarded by having windows defender on, disabling optional features like Remote Registry, enabling Controlled Folder Access, and reviewing of the Event Viewer logs to ensure that auditing is taking place. Ubuntu was hardened through update, enabling UFW with default-deny rule, permitting SSH to a subnet only within the internal subnet, turned off unnecessary services, giving correct user permissions, and checking the system logs of the system. To ensure that both systems had the appropriate level of security, Wazuh compliance was verified to enable CIS Benchmark modules, hardening policies, and generate reports.

Figure 26

Windows Hardening

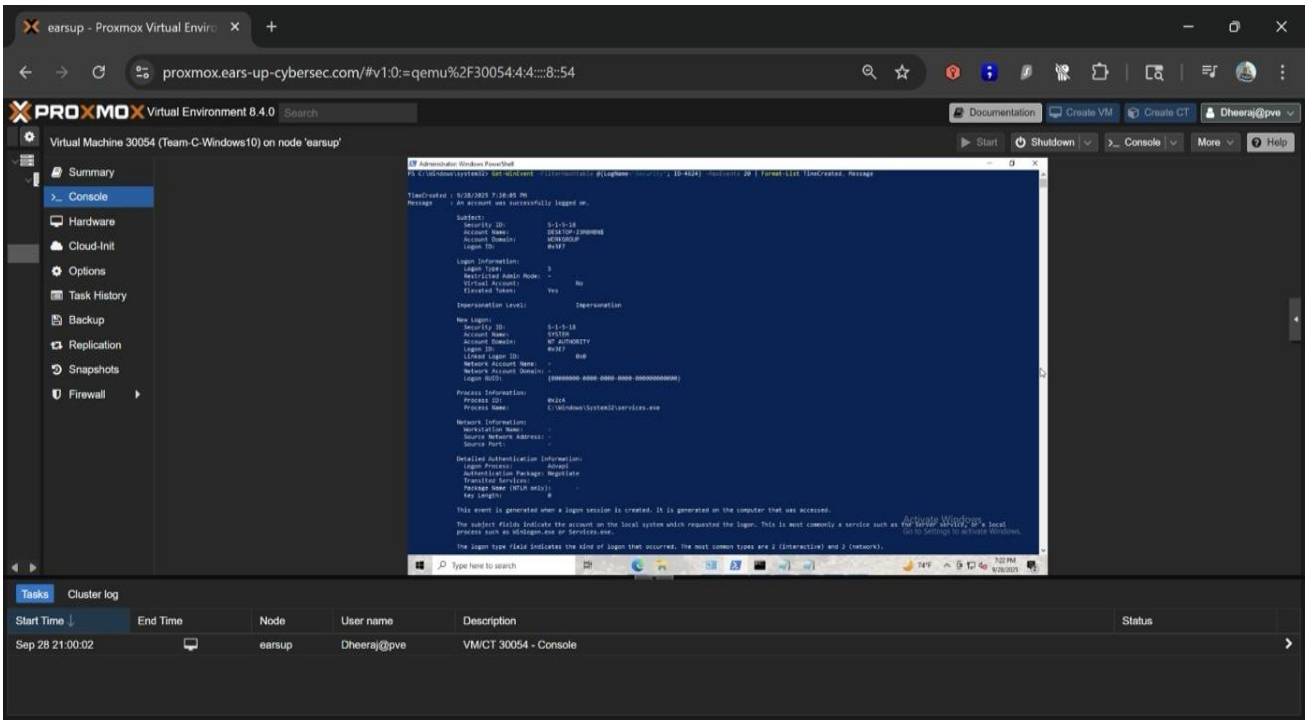
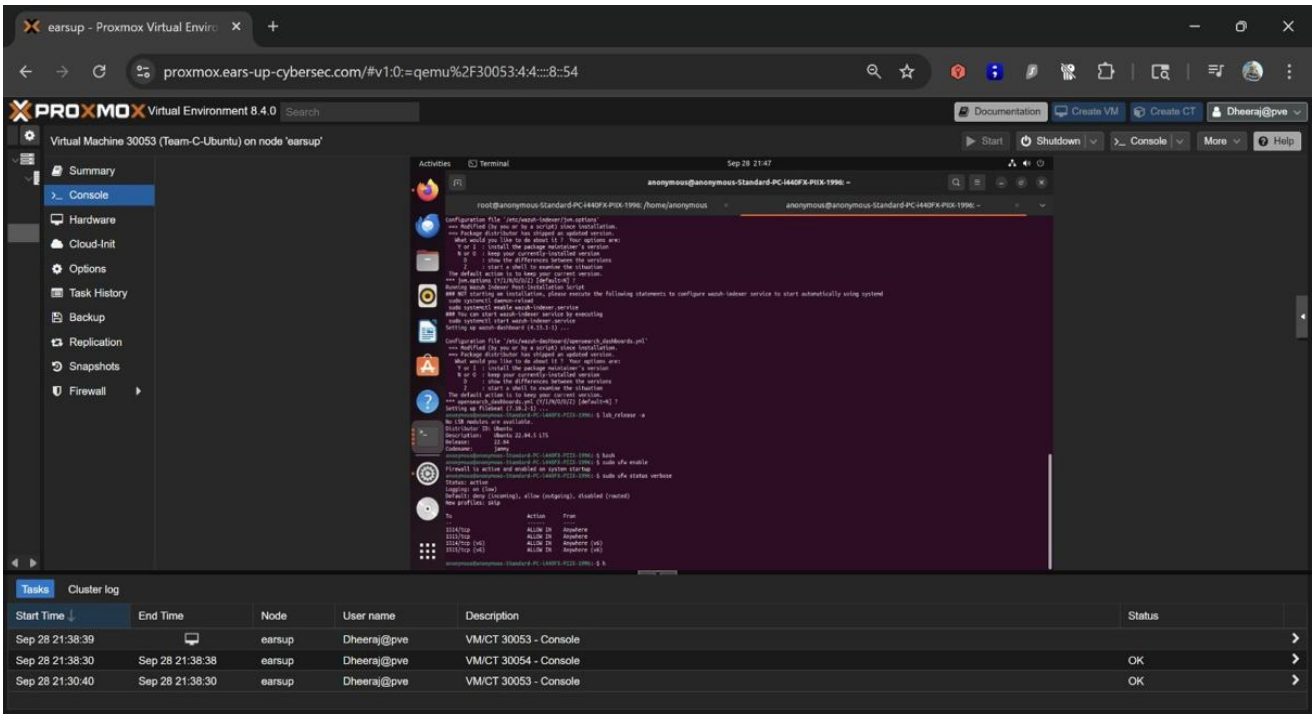


Figure 27

Ubuntu Hardening



Threat Intelligence & MITRE ATT&CK Mapping

We also made the mapping of MITRE ATT+CK in Wazuh possible and verified it using the dashboard. On DESKTOP-23M8HBN, technique A (T1105: Ingress Tool Transfer, rule ID 92213) was observed on October 29 as powershell.exe was used to run a script in a Temp file, which implies tool transfer to Command-and-Control. On the same day, Technique B (T1078: Valid Accounts, rule ID 5501) has been observed on the anonymous-Standard-PC-i440FX-PIIX-1996, and a valid account (root) was used in a PAM session, which could indicate the possibility of privilege escalation (Jiang et al., 2025).

Figure 28

MITRE ATT&CK Dashboard

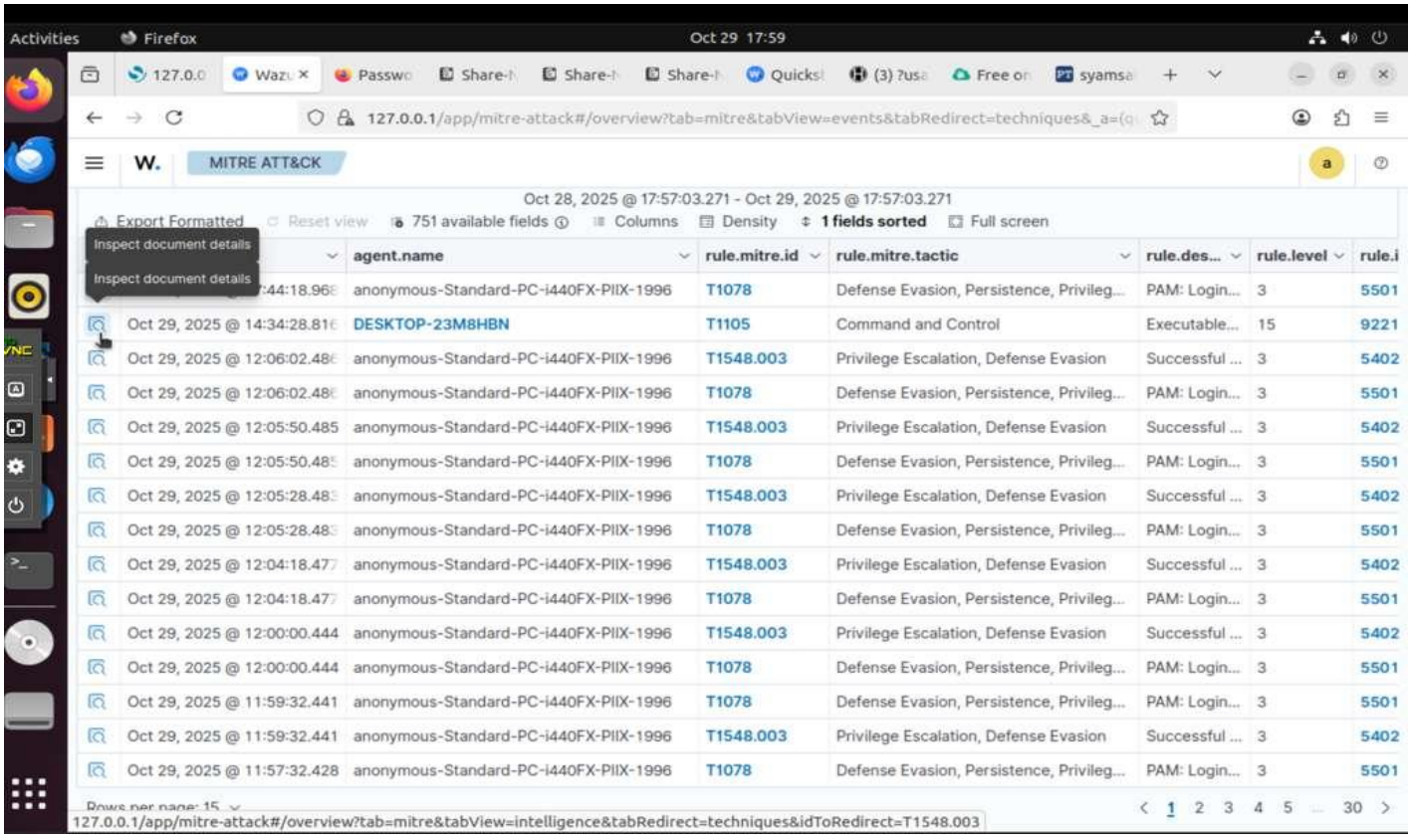


Figure 29

T1105 Details

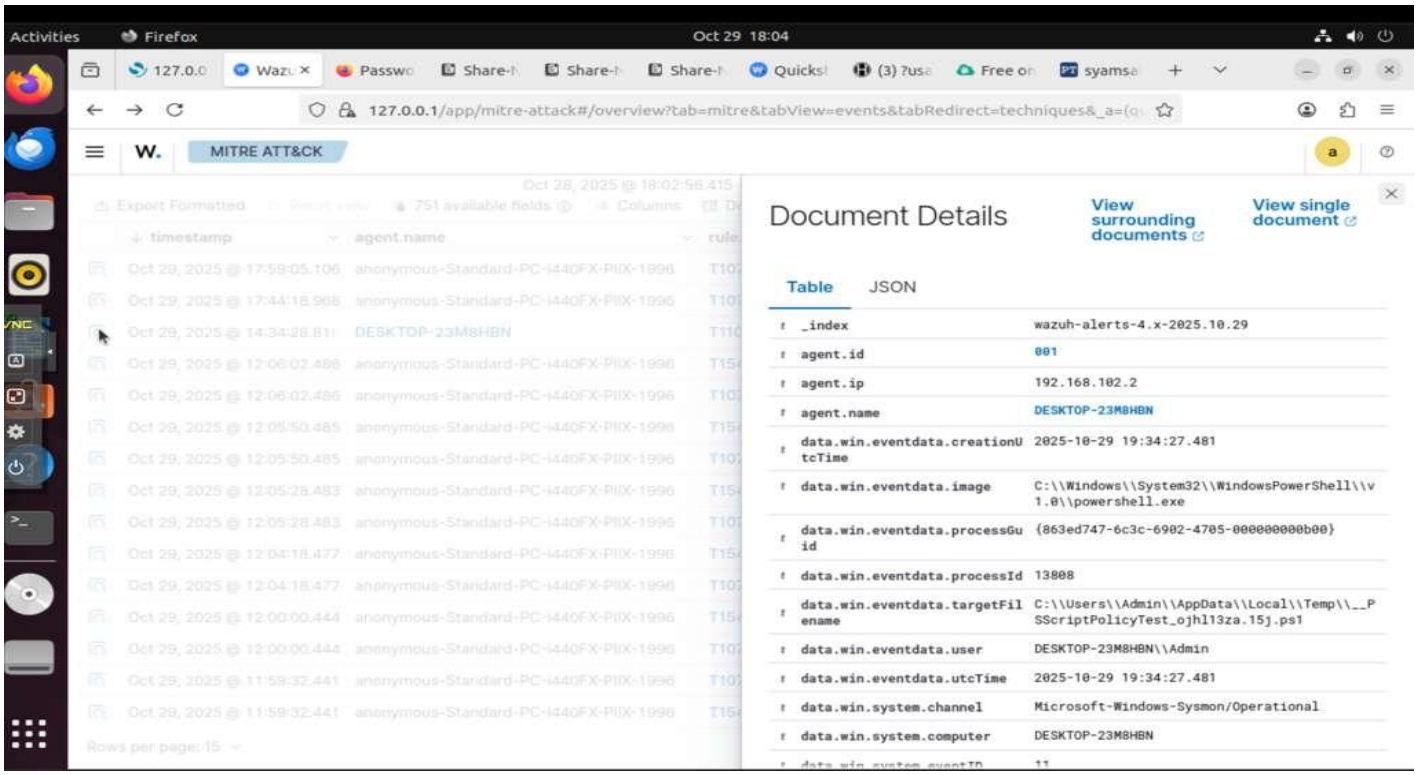


Figure 30

T1105 Details

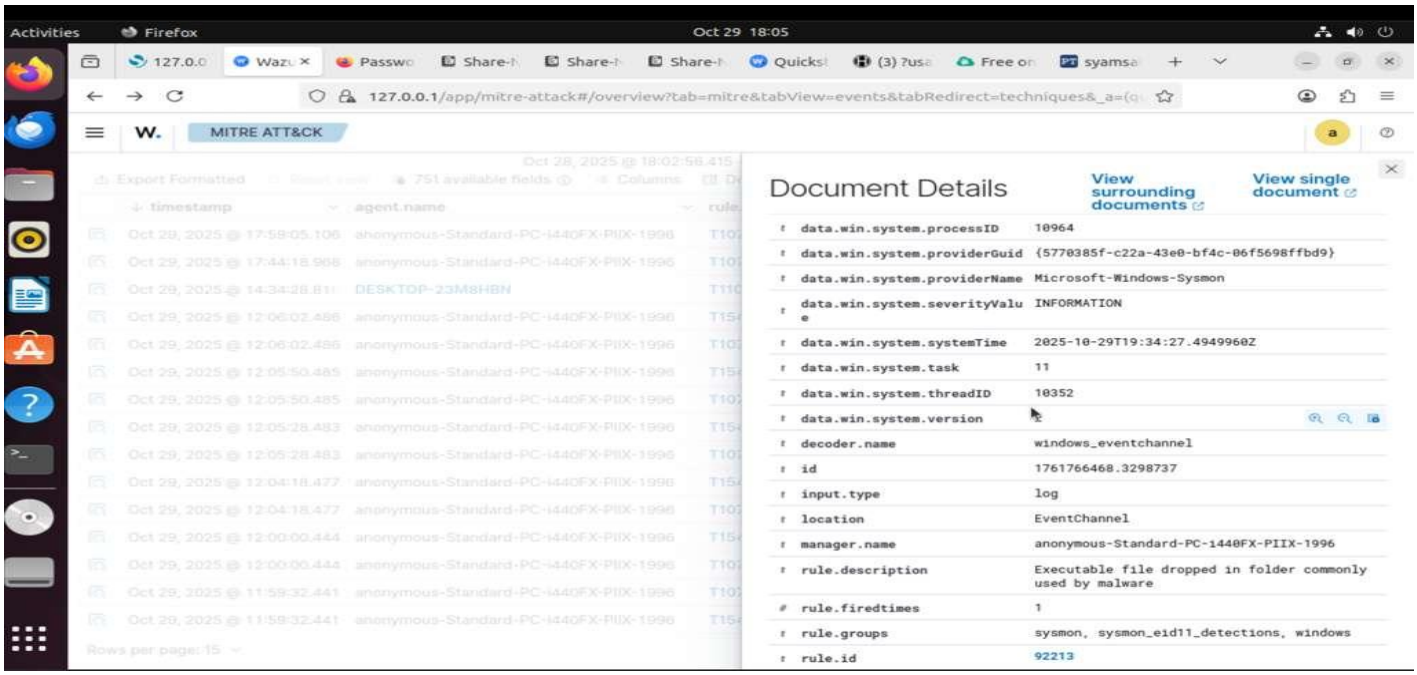


Figure 31

T1105 Details

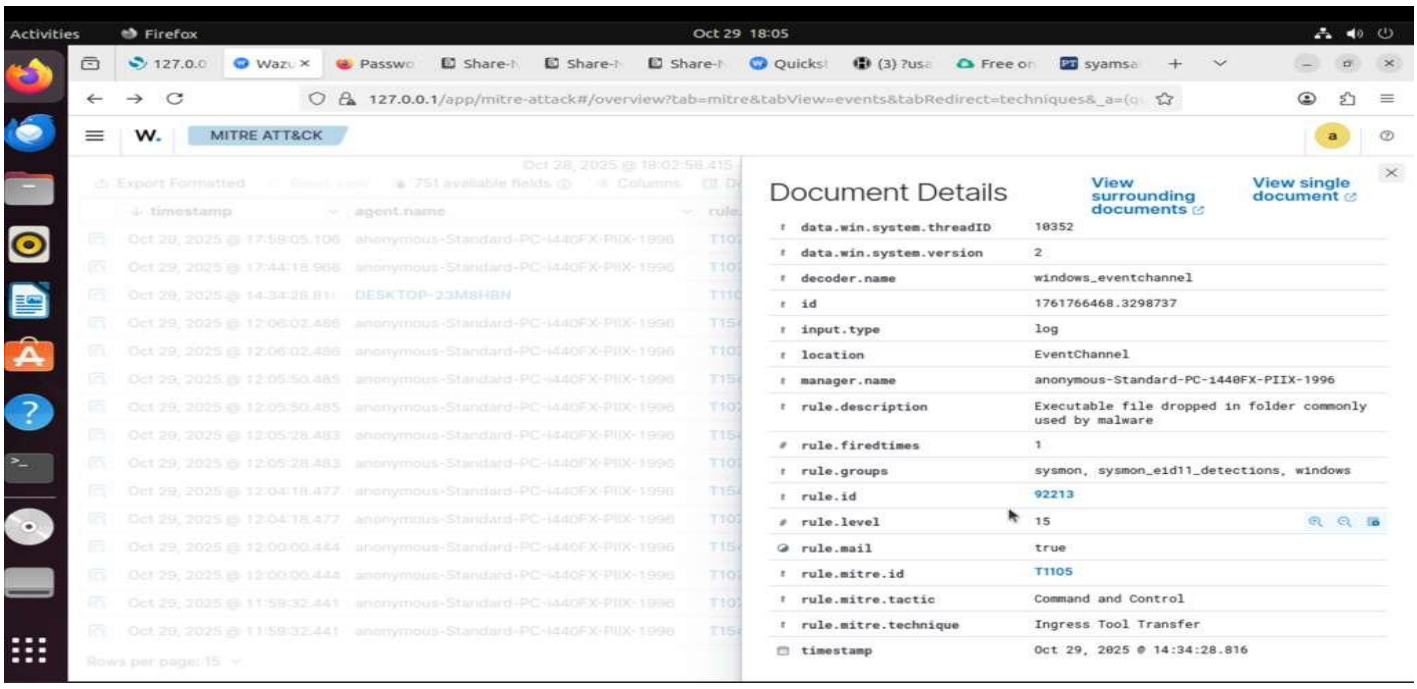
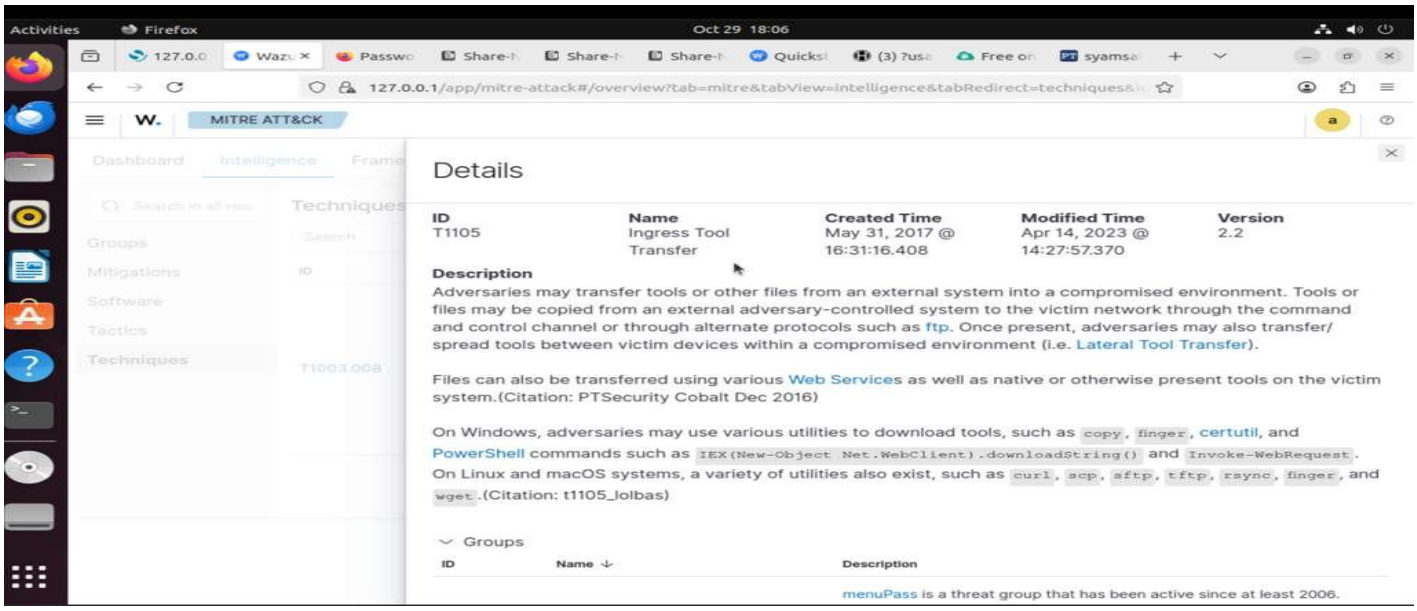


Figure 32

T1105 Details



VPN Configuration and Connectivity

We managed to install WireGuard by downloading the latest installer and launching it in administrator mode and importing the tunnel configuration file that was provided to us. The VPN tunnel was enabled and displayed as an active one in the WireGuard application. PowerShell connectivity tests verified access to the network, pinging 192.168.26.6 was successful and Test-NetConnection to port 80 was successful indicating that a web server is available (Hoxha, 2020).

Figure 33

WireGuard

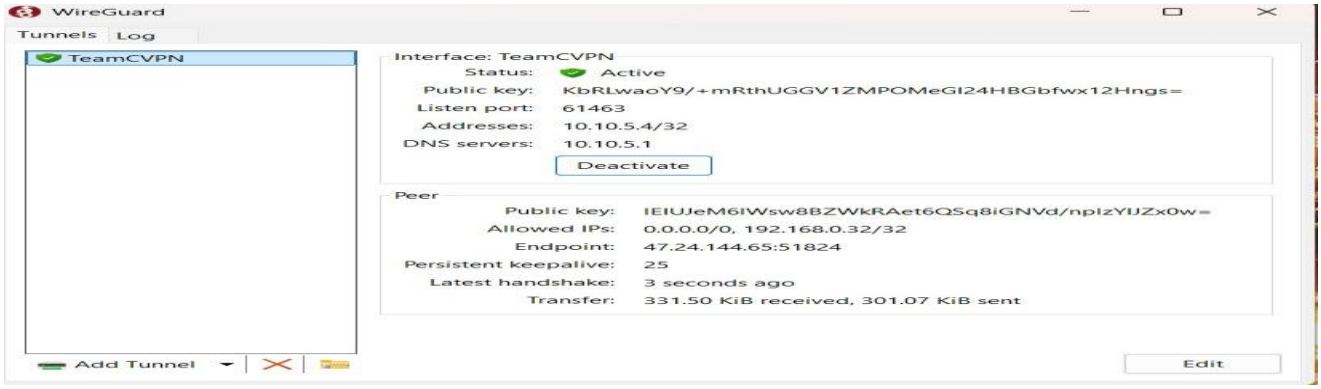


Figure 34

Port 22 Connection Testing

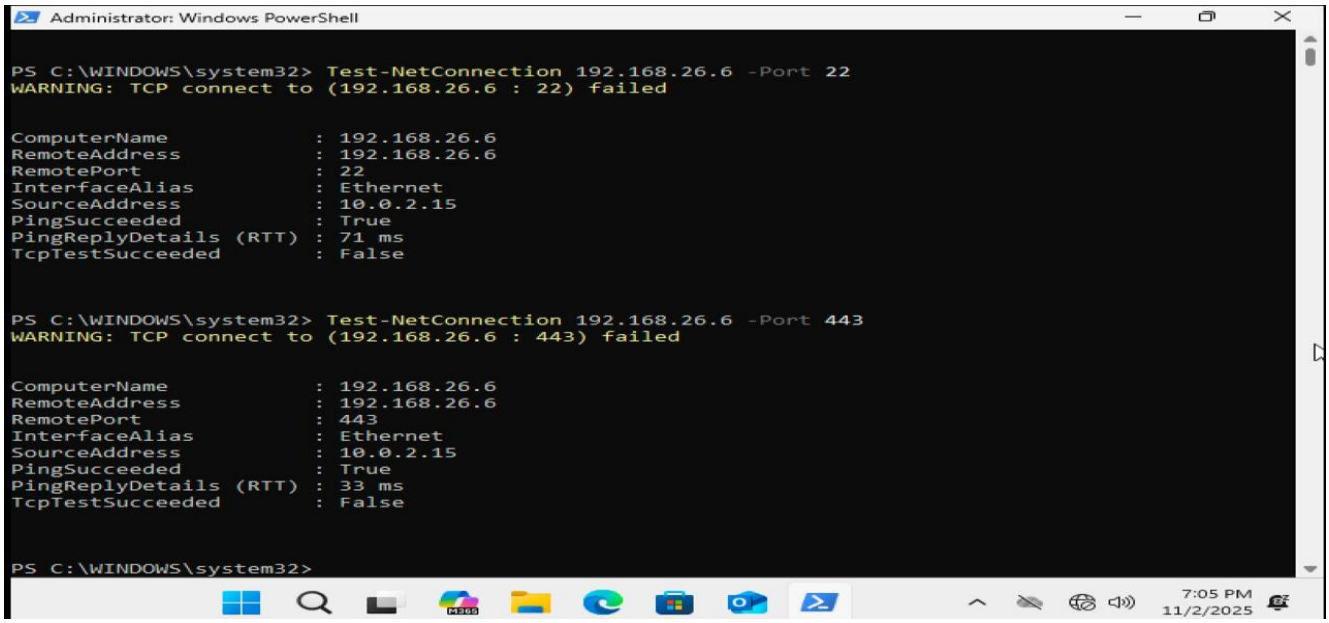
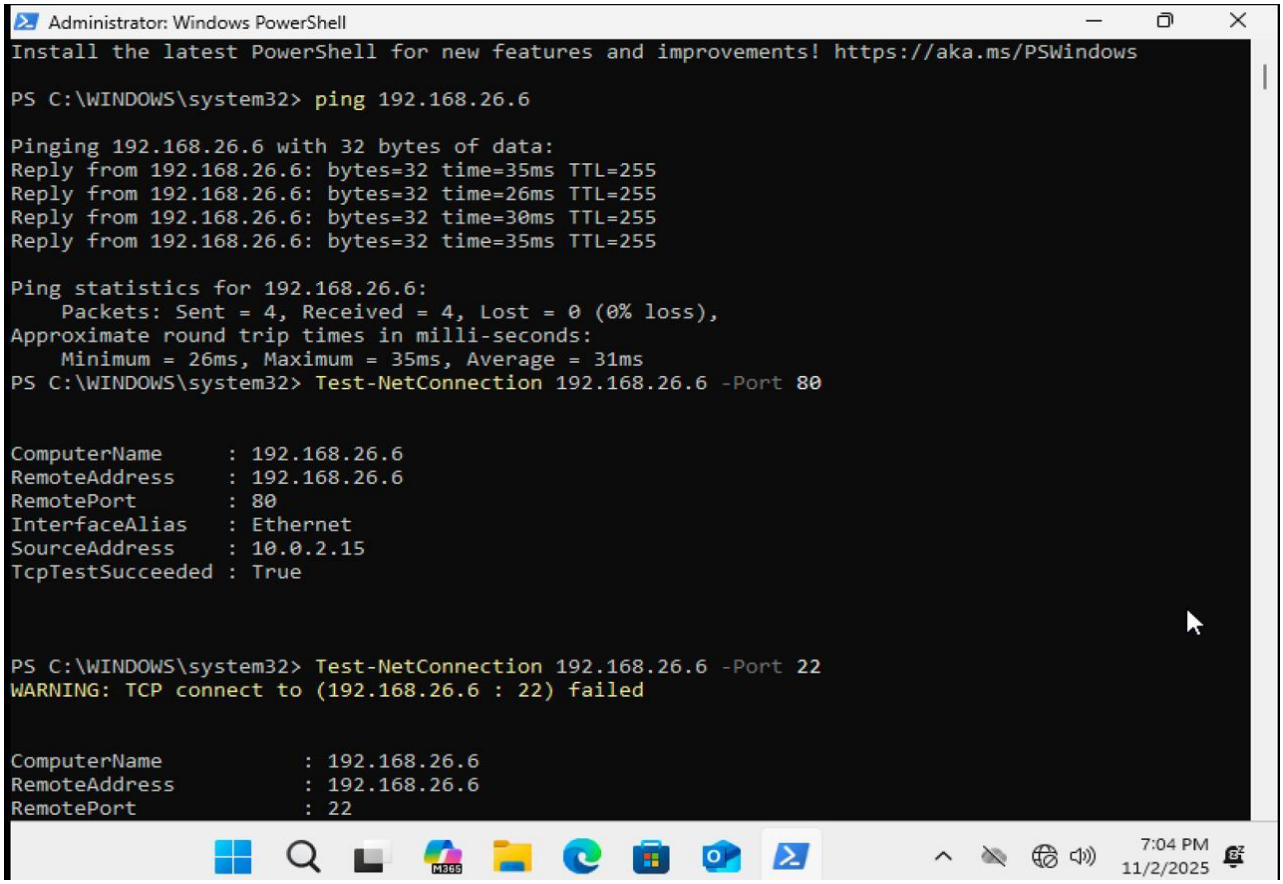


Figure 35

Ping



Red Team Offensive Simulation

Finding 1: SMB Access Achieved

It was possible to gain access to SMB on 192.168.26.6 with authenticated credentials having a chance to enumerate shares including ADMIN$, C$, F$ and IPC$ shares and no guest or null connections allowed. The severity of this concern is a High severity (CVSS 8) one because the compromised credentials can be used to move laterally, and it must be addressed by imposing strong access controls and tracking the SMB activity (Inbal, 2021).

Figure 36

SMB

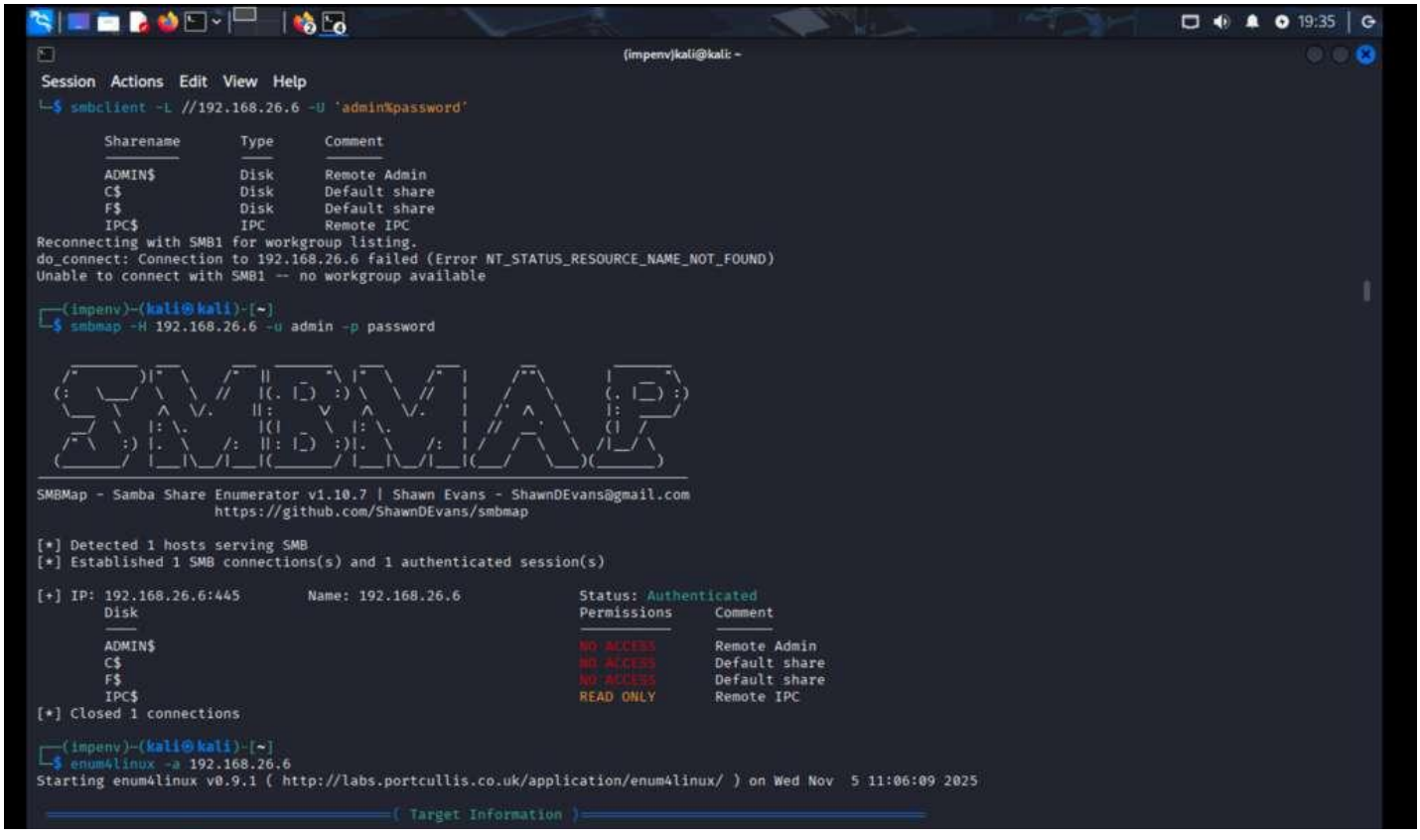
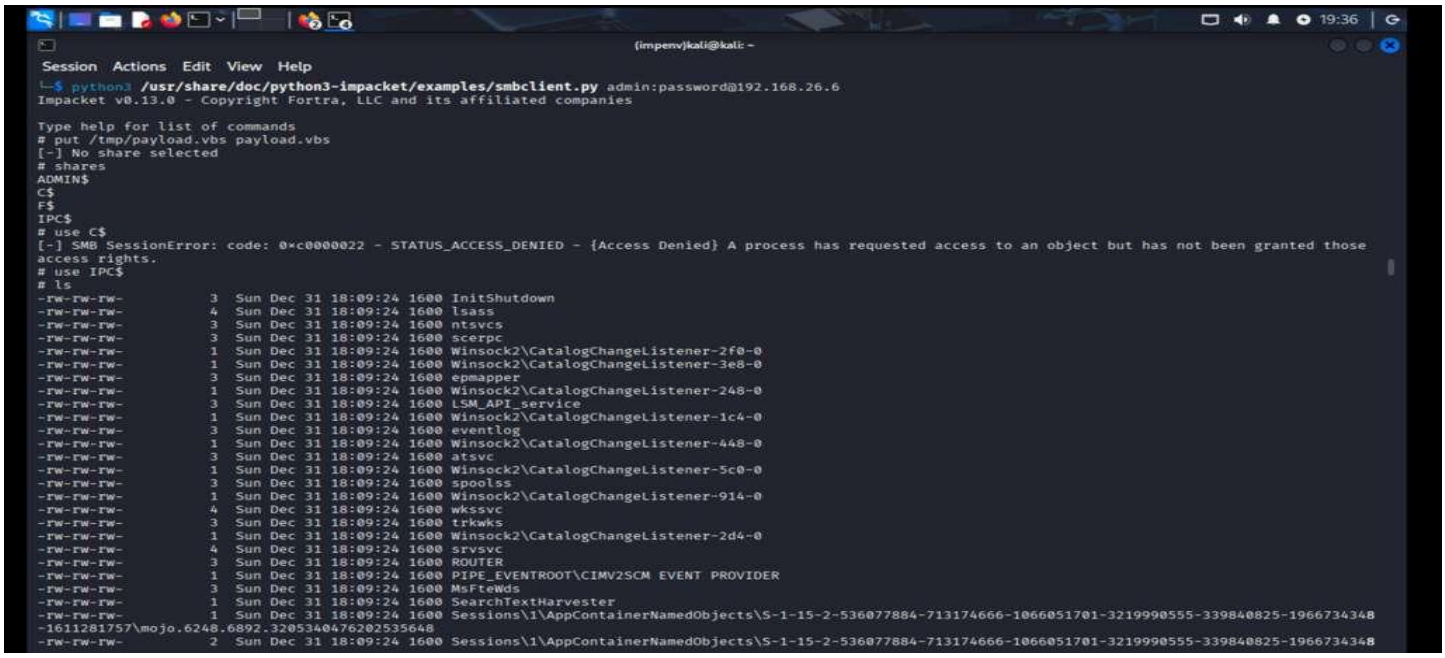


Figure 37

SMB

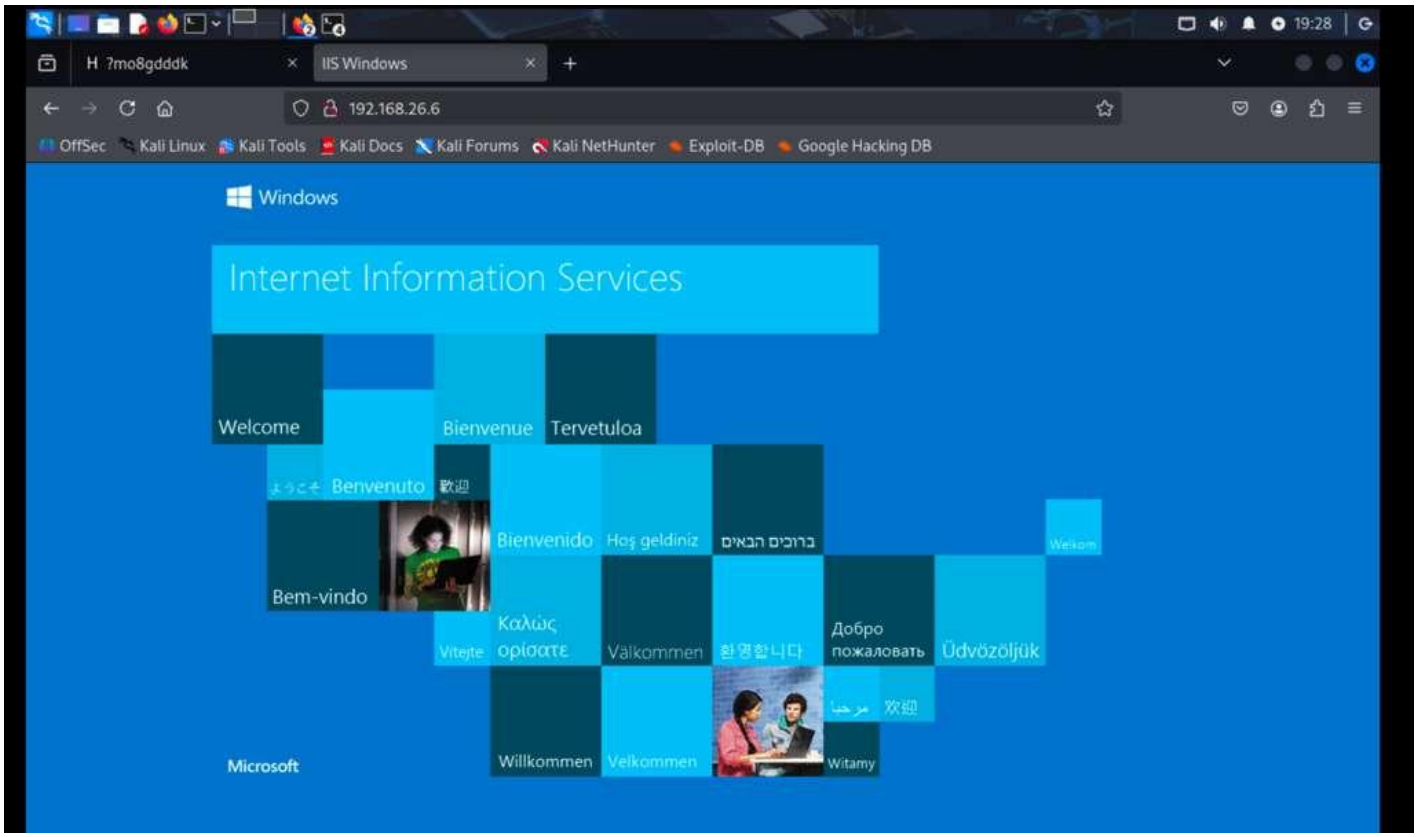


Finding 2: IIS Running with Default Configuration & TRACE Enable

The host 192.168.26.6 has been discovered to be running IIS with its default settings open on a port 80 and TRACE HTTP method is also available, which may be used to facilitate cross-site tracing attacks. The level of this problem is Medium (CVSS 5), and the solution will be disabling TRACE in IIS, enabling HTTPS, and deleting default IIS pages (MDN contributors, 2025).

Figure 38

IIS



Finding 3: Network Reconnaissance Identified Open Services

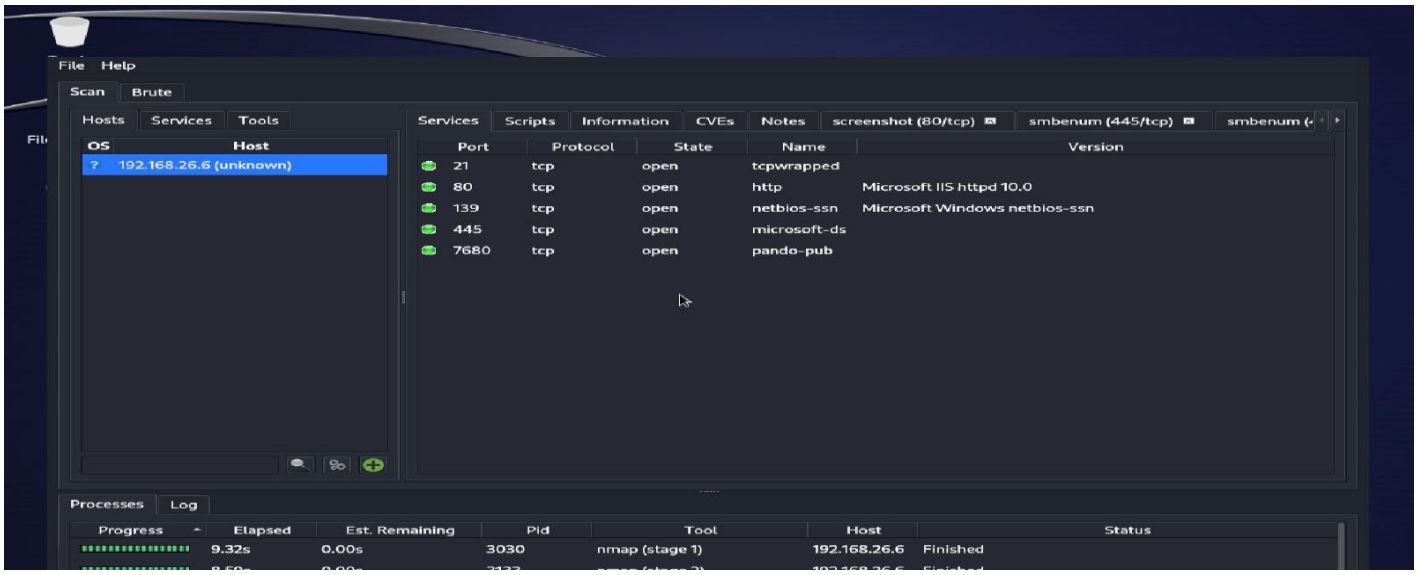
Network reconnaissance identified that host 192.168.26.6 had some open ports and running service information found with Nmap and Legion, but there were no direct vulnerabilities available at the time, as the available service information may be used in targeted attack (GeeksforGeeks, 2020). This is Low severity (CVSS 3), and the remediation would involve a reduction in unnecessary exposed services, periodic scanning of open ports, host-based firewall rules, and detection of suspicious scanning activity (Wattuhewa, 2023).

Figure 39

Nmap

Figure 40

Legion



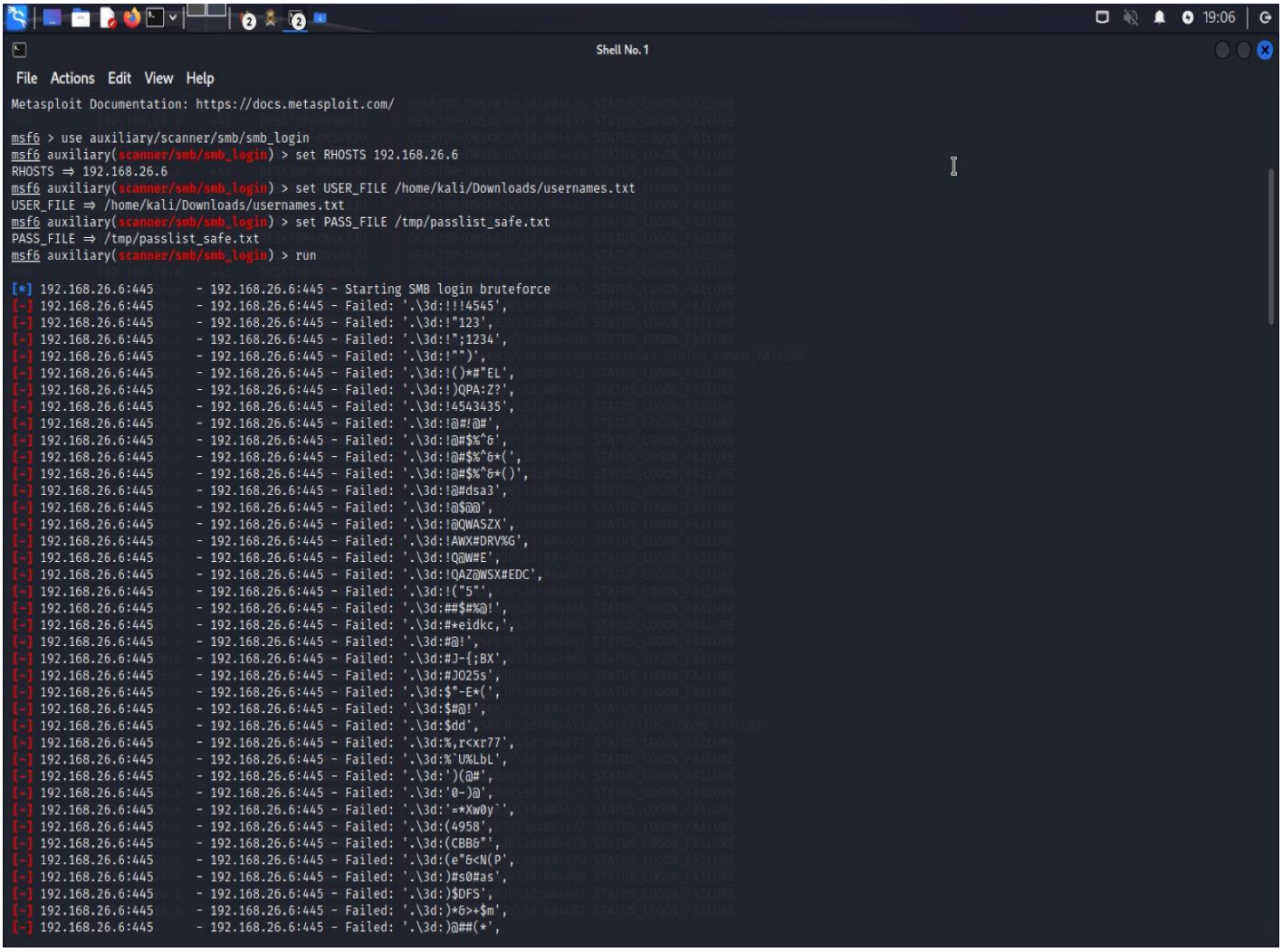
Finding 4: SMB Login Brute-Force Attempts Blocked

A brute-force attack attempt on host 192.168.26 based on the smb_login Metasploit module and using several usernames and passwords to brute-force into host was unsuccessful, indicating account protection measures or lockout policies are in place. It is a Medium-severity vulnerability (CVSS 5), and its mitigation measures involve having policies of strong passwords and lockouts, attending to

recurrent unsuccessful logins, and the possibility of having SMB signing and multifactor authentication of sensitive accounts (Faturrohman et al., 2023).

Figure 41

Brute-Force



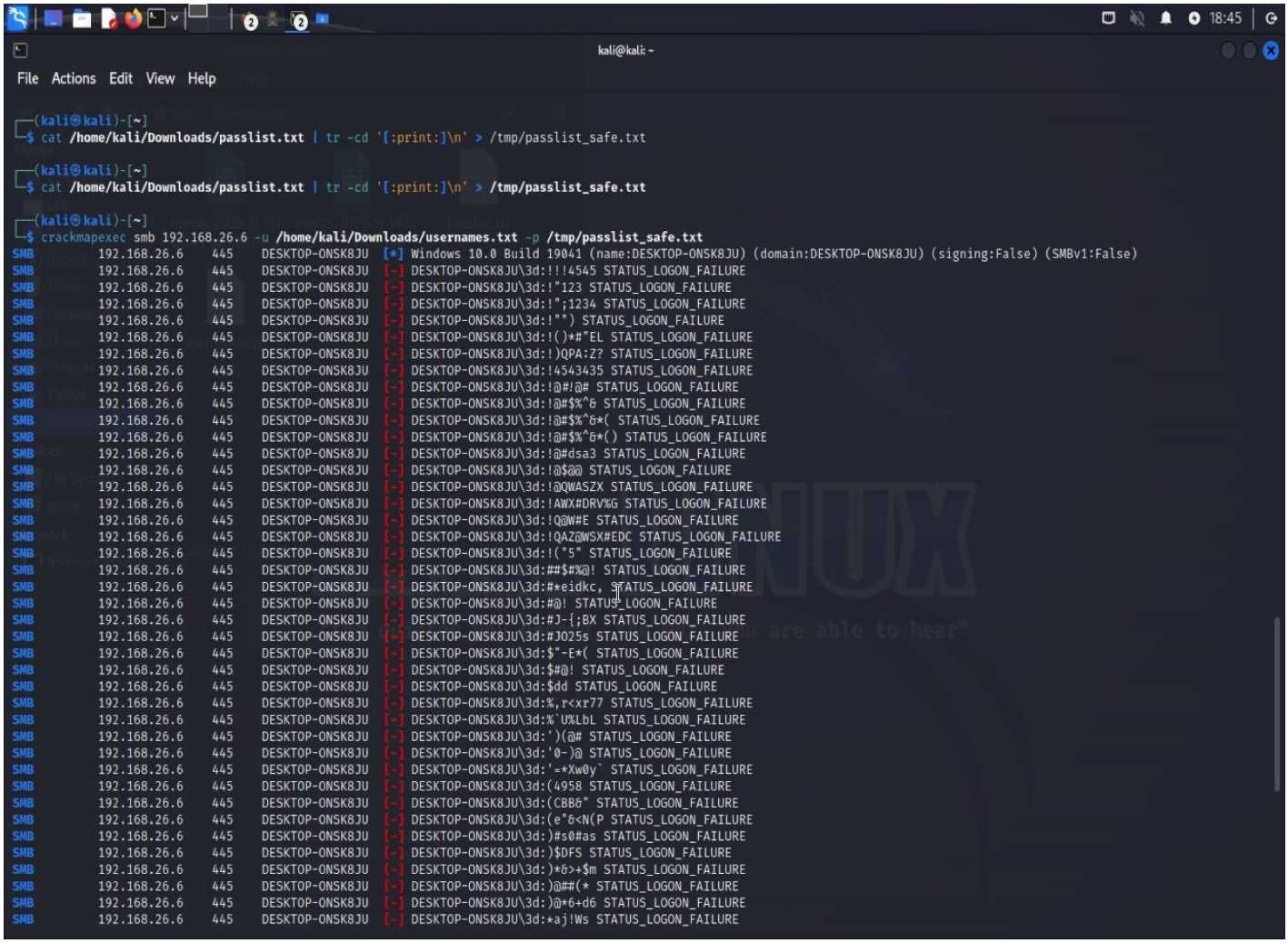
Finding 5: SMB Signing Disabled

The host is using windows SMB service, which does not have SMB signing enabled, which means that no SMB traffic has been validated, and is susceptible to Man-in-the-Middle attacks, message tampering, or authentication relays. Though no credentials were compromised the same undermines

lateral-movement protection. It is a medium-level language problem (CVSS 6), and the remedies involve the ability to use SMB signed communication via Group Policy, limit SMB traffic using host-based firewalls, and proper management of the credential to prevent credential reuse (Nairuz, 2023).

Figure 42

Crackmap

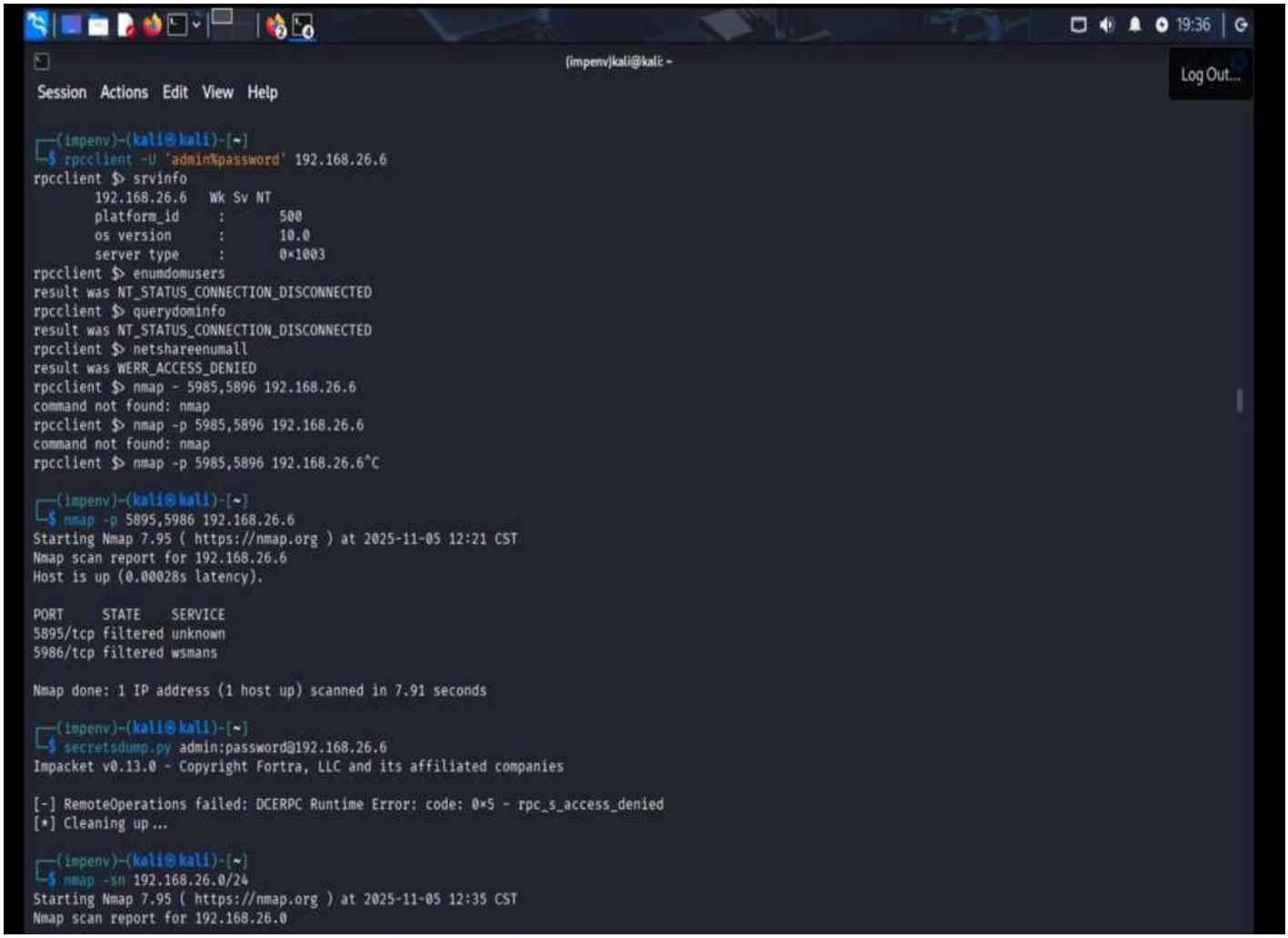


Finding 6: Remote Service Information Disclosure via RPC

Host revealed the operating systems information through the srvinfo RPC command even though most of the enumeration was blocked which meant that attackers could customise exploits or password attacks based on the OS and configuration. It is a Medium-level problem (CVSS 6), and its solution involves disabling the unnecessary SMB/RPC services, permitting SMB/RPC traffic to trusted hosts, using the latest windows security patches, and implementing firewall policies to restrict RPC discovery traffic.

Figure 43

RPC



Finding 7: FTP Access on Port 123

FTP service was found open on the nonstandard port 123 and the search of the FTP directory showed that there were two files: john.txt, which merely had the teasing text line “I am not easy! Chat GPT will crash,” and the user input to Challenge.zip, which decompresses to a main.c program that will XOR-transform the input and compare it with a hard-coded XOR-encoded array, is XOR-transformed and then compared before the program produces the success message.

Figure 44

FTP

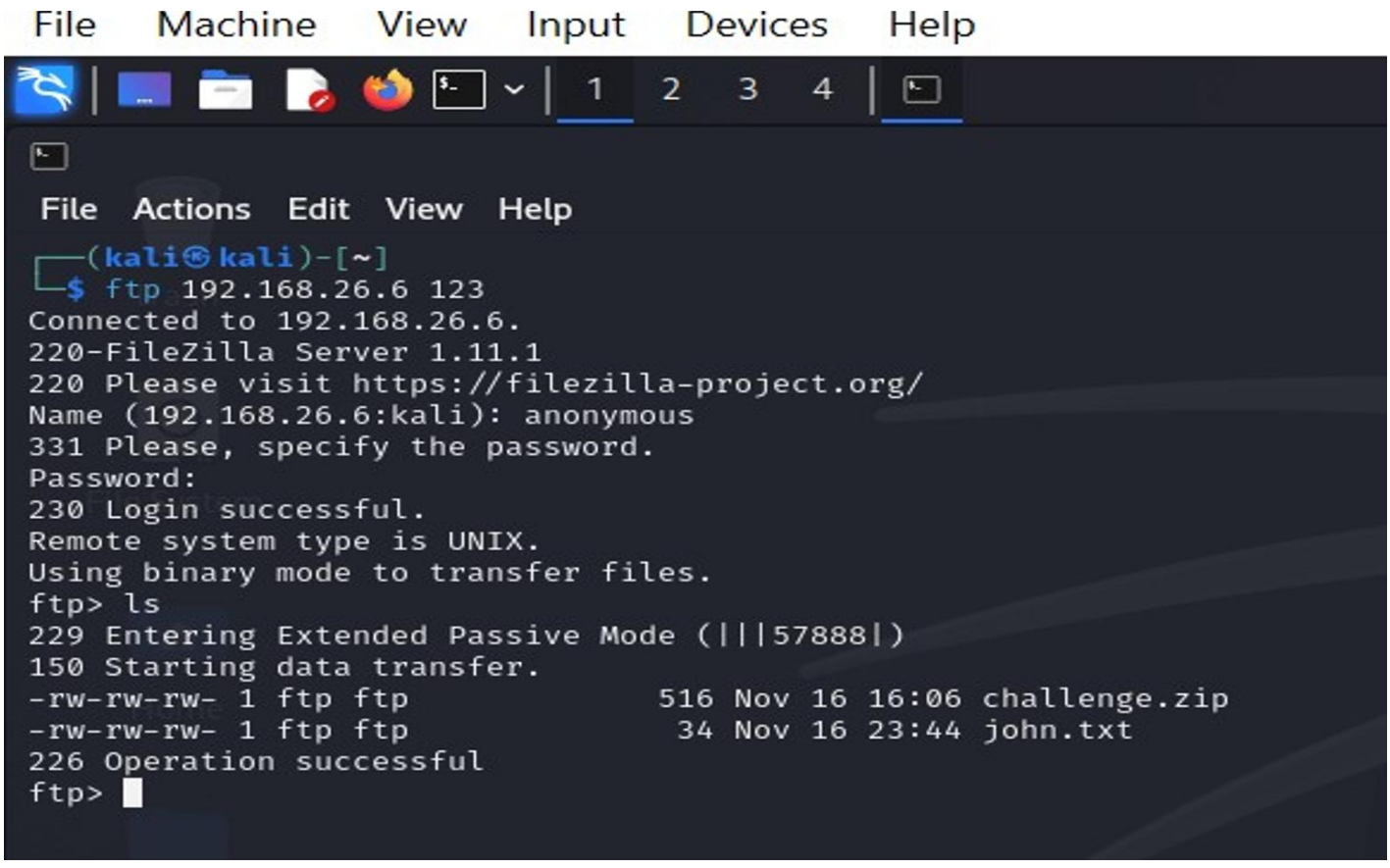


Figure 45

Contents of john.txt

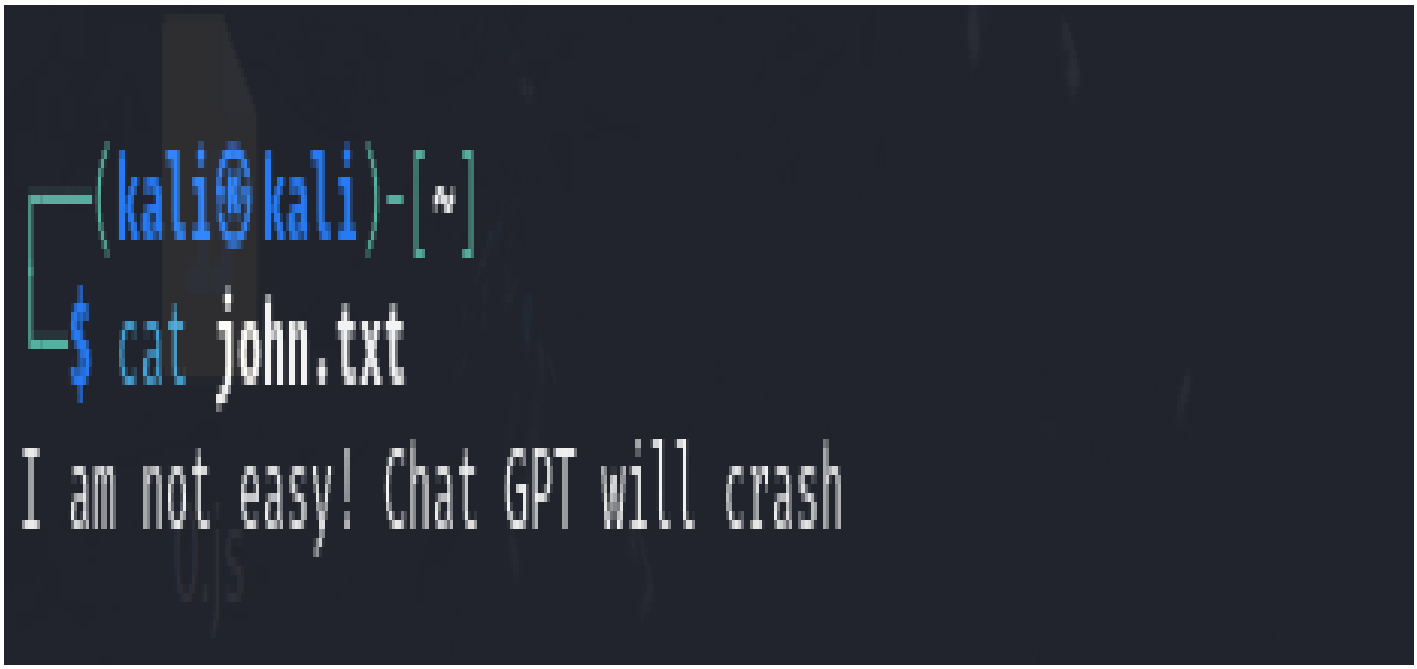
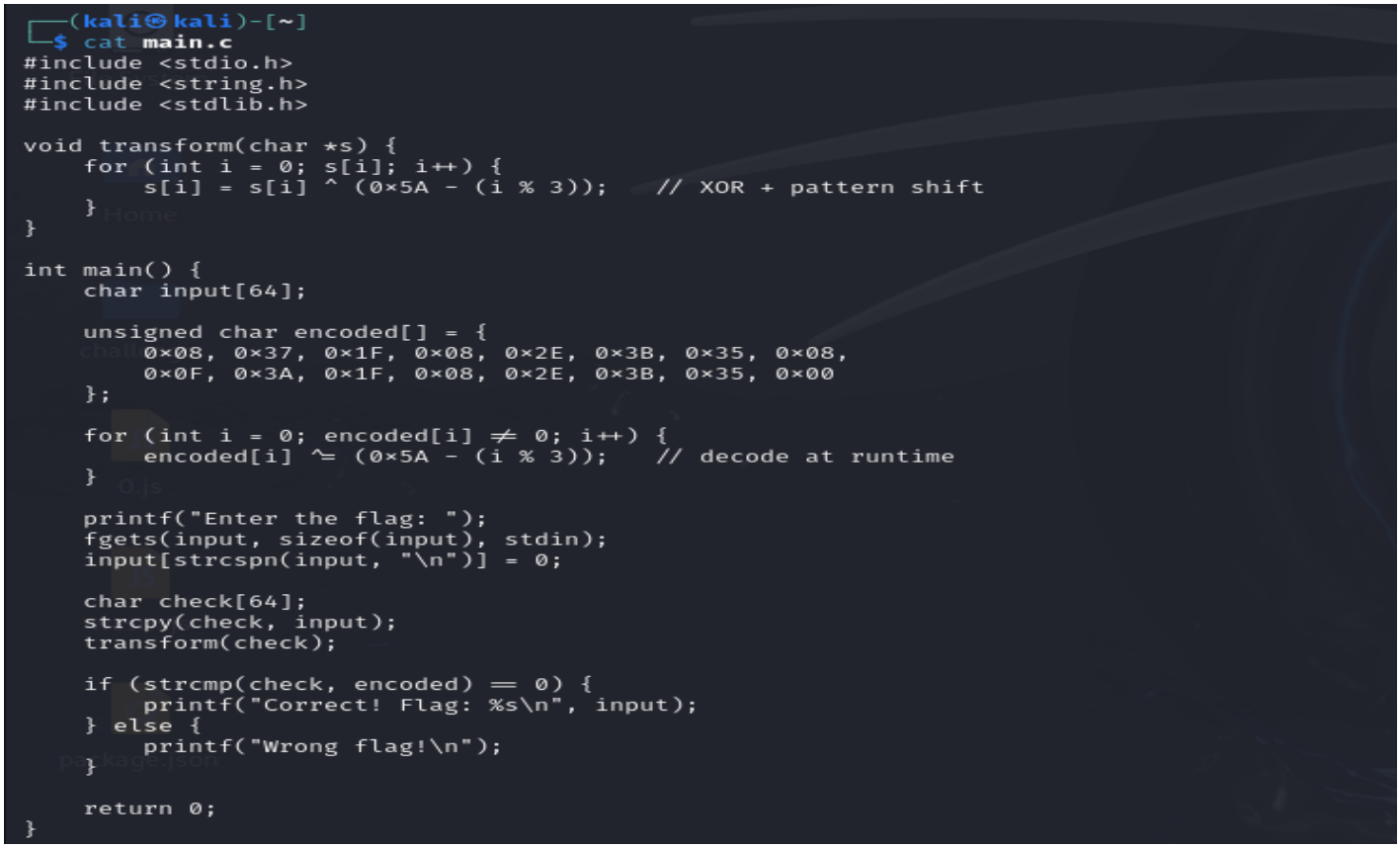


Figure 46

main.c



```
(kali㉿kali)-[~]
$ cat main.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void transform(char *s) {
    for (int i = 0; s[i]; i++) {
        s[i] = s[i] ^ (0x5A - (i % 3));   // XOR + pattern shift
    }
}

int main() {
    char input[64];

    unsigned char encoded[] = {
        0x08, 0x37, 0x1F, 0x08, 0x2E, 0x3B, 0x35, 0x08,
        0x0F, 0x3A, 0x1F, 0x08, 0x2E, 0x3B, 0x35, 0x00
    };

    for (int i = 0; encoded[i] != 0; i++) {
        encoded[i] ^= (0x5A - (i % 3));   // decode at runtime
    }

    printf("Enter the flag: ");
    fgets(input, sizeof(input), stdin);
    input[strcspn(input, "\n")] = 0;

    char check[64];
    strcpy(check, input);
    transform(check);

    if (strcmp(check, encoded) == 0) {
        printf("Correct! Flag: %s\n", input);
    } else {
        printf("Wrong flag!\n");
    }

    return 0;
}
```

Blue Team Defensive Simulation

The Blue Team used the Wazuh threat-hunting dashboards that have agent-level granularity, where the registry, Sysmon, and command-execution activity were combined to reveal the following key indicators identified: suspicious service-creation event (rule ID 92307), Sysmon-detected outbound network connections to external IPs out of C:Program Files (rule ID 61605), and reconnaissance or account-discovery attempts forwarded with net.exe (rule ID 92039). They were set to moderate and high (level 3-5) alert thresholds to facilitate active triage and dashboards provided timestamped and correlated visibility by multiple kill-chain phases such as persistence, C2 communication and discovery so that they can easily detect and respond effectively.

Figure 47

Wazuh Dashboard

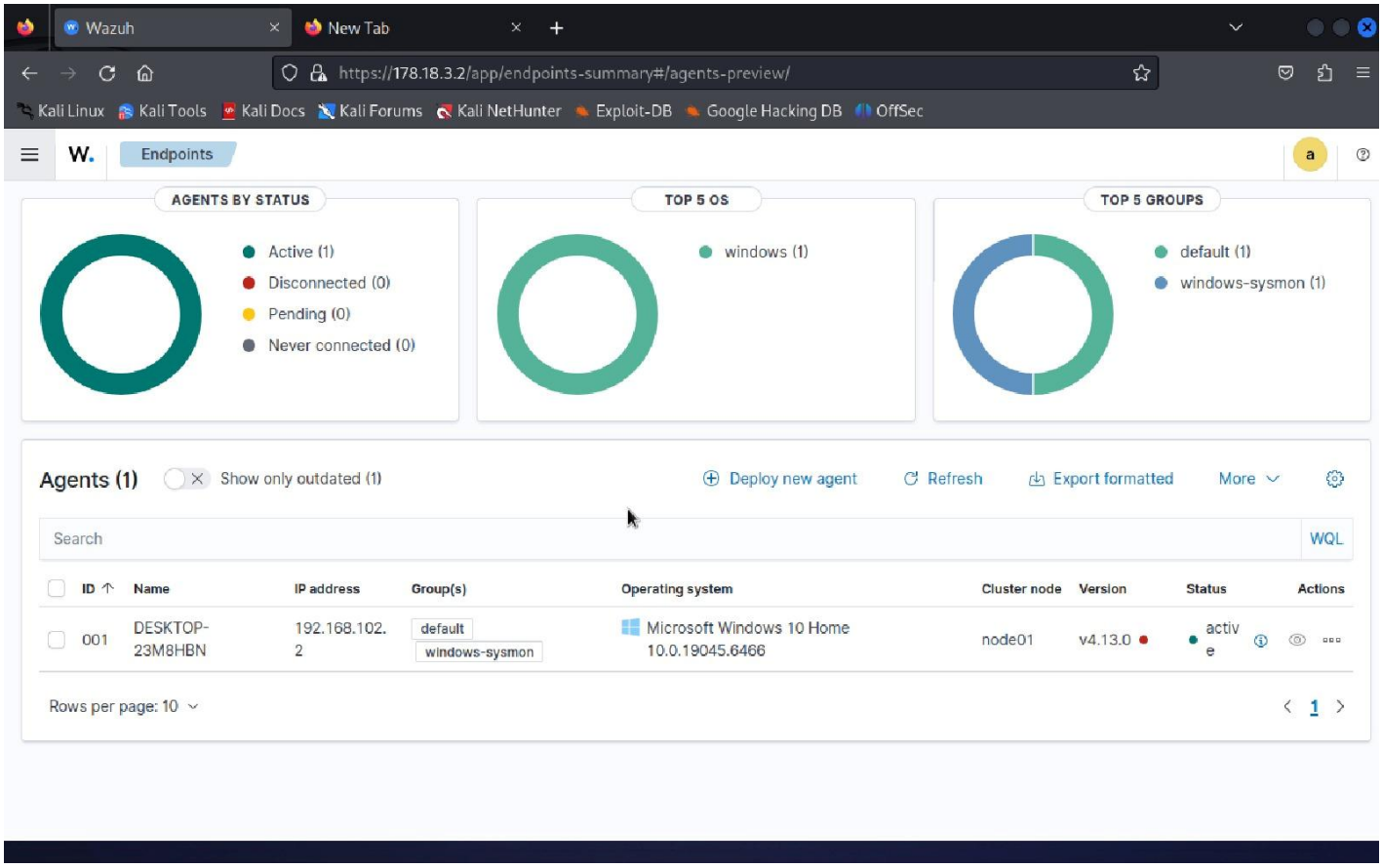
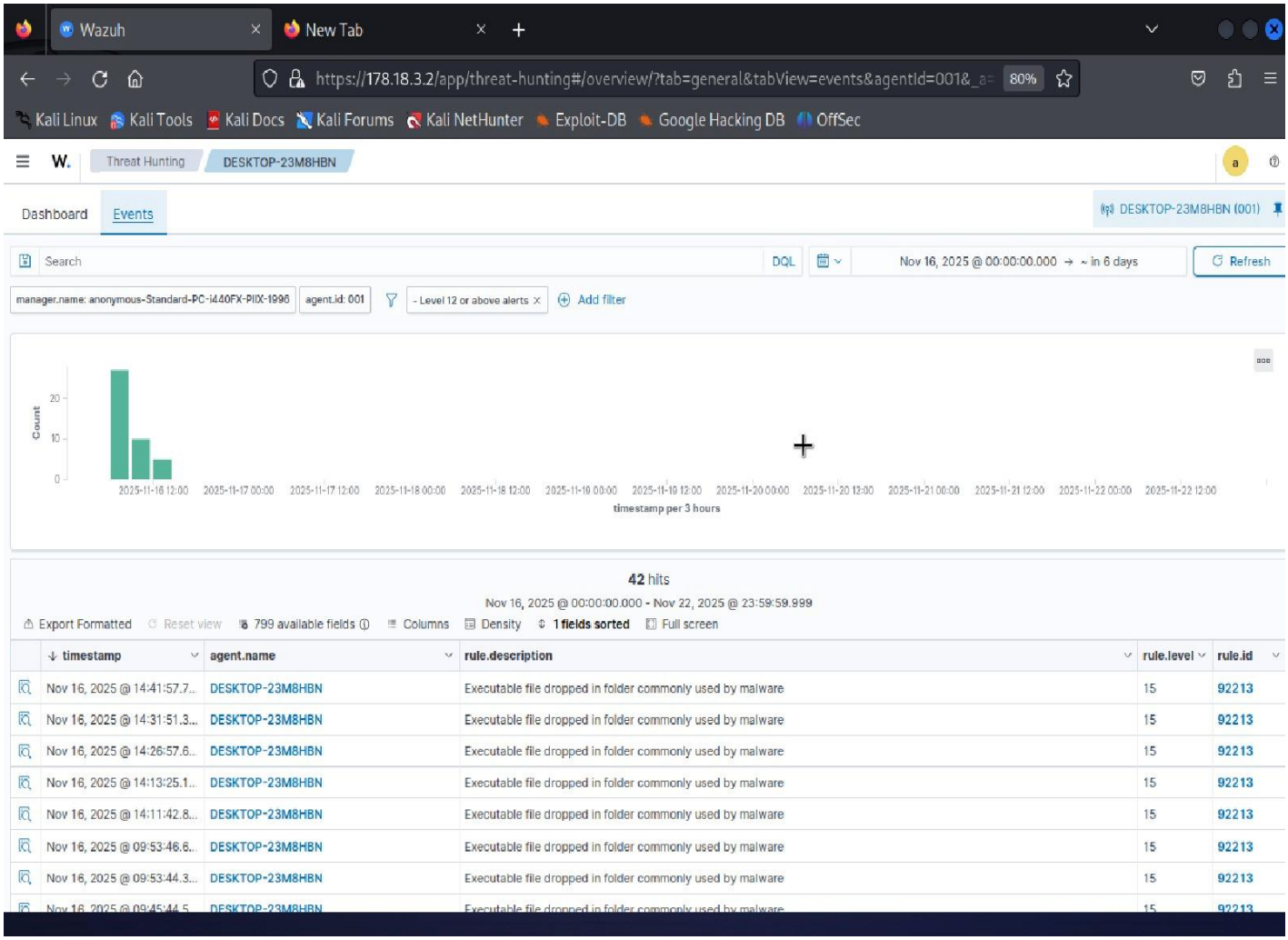


Figure 48

Wazuh Events



We can observe repetitive activity of command shells and auto /scheduled processes, malware drops of high severity (level 15) on common folders, 689 SSH logon failures and 110 SSH connections by 192.168.78.9 indicating a possibility of brute-force attack or unauthorized access. Other anomalies are system time, service startup as well as registry integrity. Most network activity is safe (Microsoft Defender endpoints), but 192.168.78.9:22 is unusual, and the CVEs 59505 (occasional) suggested the need to patch. These trends lead to potential enduring threats, signs of malware and target attacks during the period under investigation.

Figure 49

Wazuh rule.id 92307

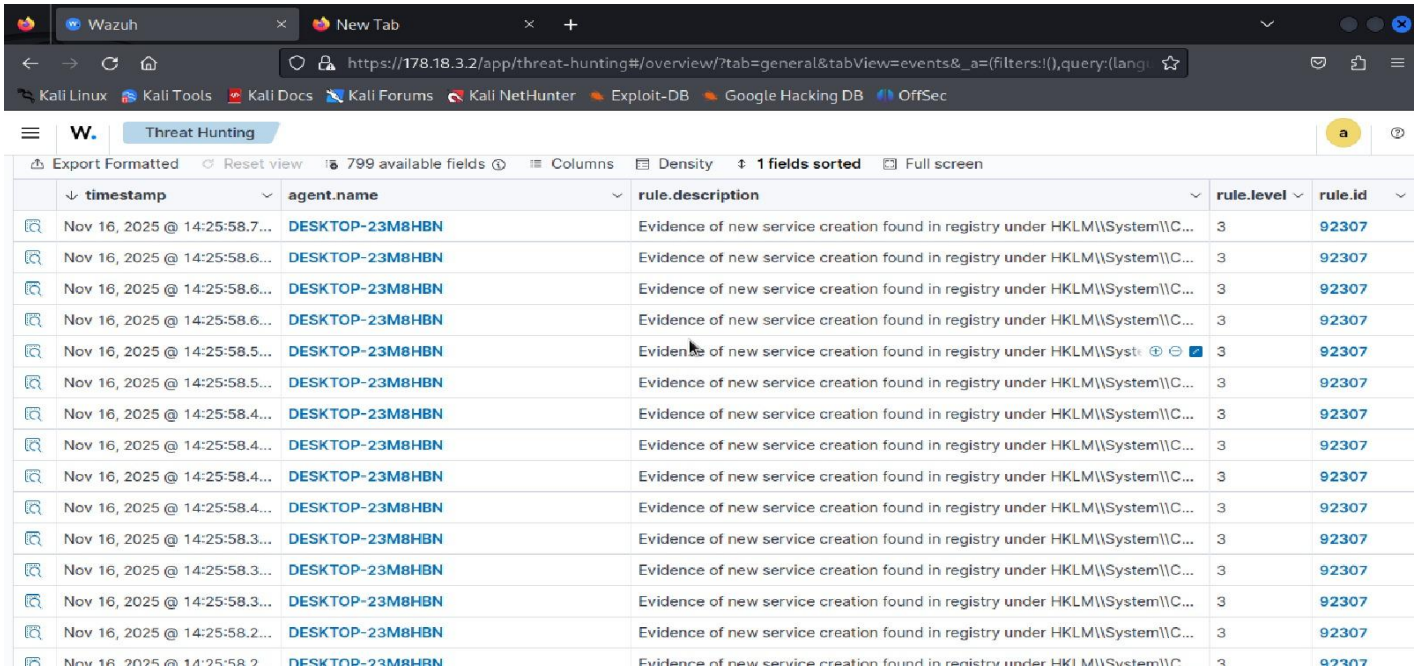


Figure 50

Wazuh rule.id 61605

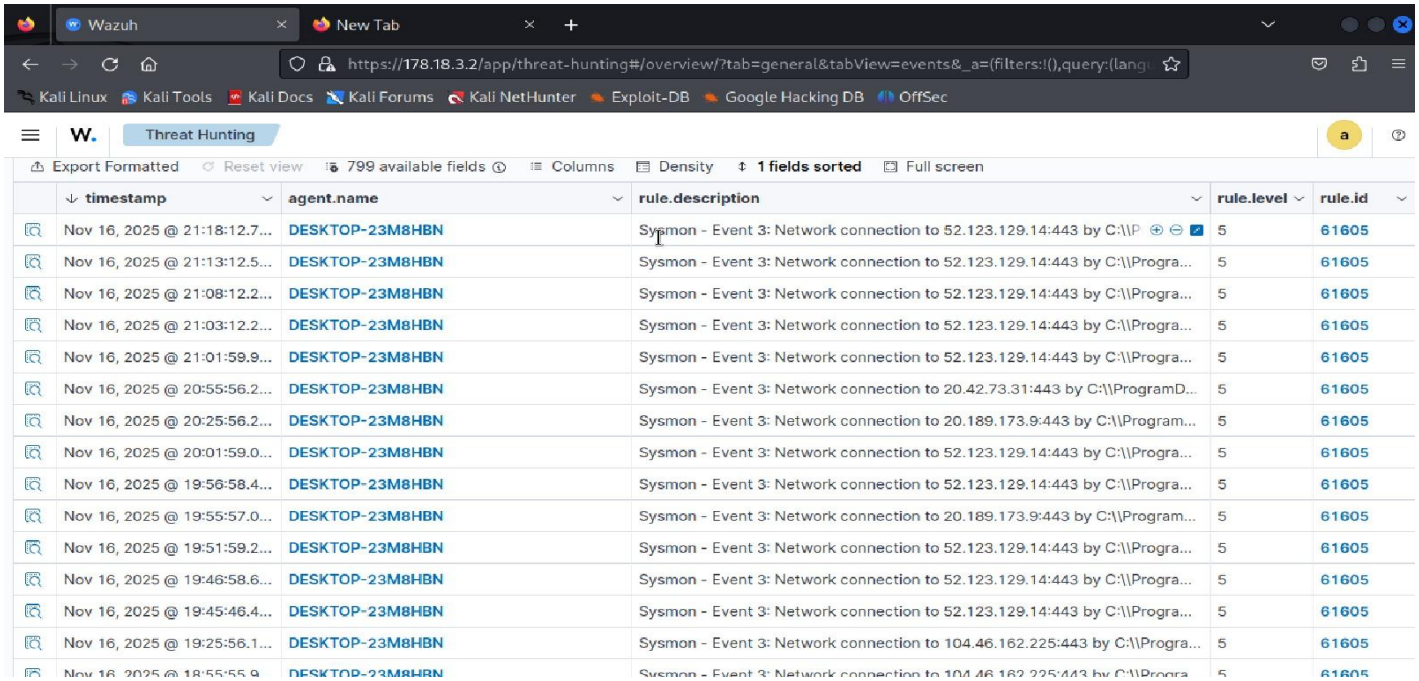
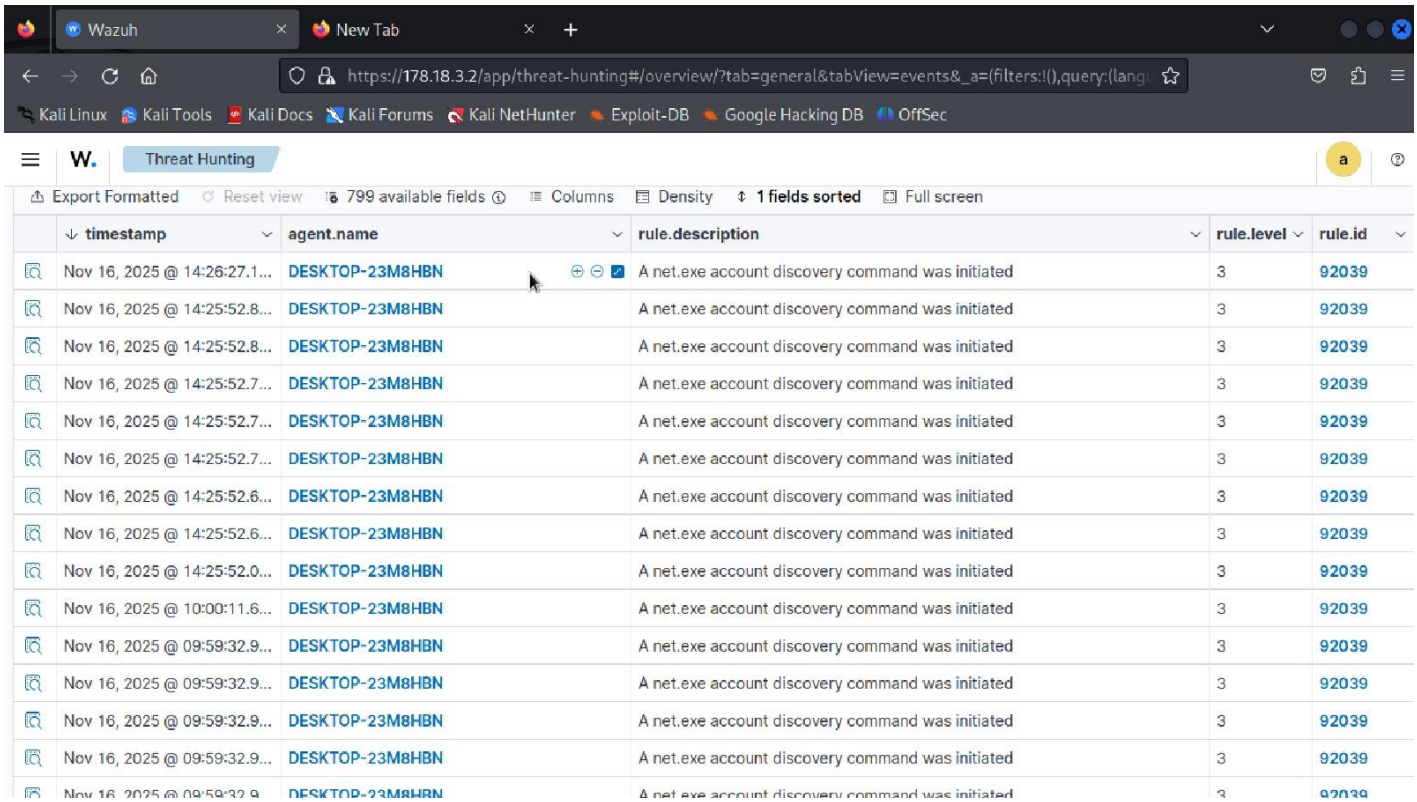


Figure 51

Wazuh rule.id 92039



The detection rules were registry service-create and registry service-modify messages (rule ID 92307) due to new entries under HKLM\System\CurrentControlSet\Services, Sysmon-based network connection messages (rule ID 61605) based on traffic to external IPs over port 443 and account-discovery-based net.exe command messages (rule ID 92039). The Dashboards facilitated triage using Events tab which was filtered by agent name and high severity rule and sorted by timestamp and rule description to enable easy investigation.

Suggestions are to investigate the 228 malware-drop events with the immediate scanning of suspicious folders and isolating the host, hardening the security with multi-factor authentication of SSH/RDP, block or monitor 192.168.78.9, and restrict the use of cmd.exe through policy. CVEs such as

2025-59505 should be patched using Windows and Defender. It is recommended to perform continuous monitoring and issue notifications in case of repeated executions of cmds and logon failures and monitor the logs to identify parent processes that triggered an abnormal cmd activity. There is a possibility of further action that would include correlation with other logs or a complete system scan during incident response.

Learning Outcomes

The project required me to build an entire enterprise environment from scratch while learning its design and security configuration methods. The combination of Proxmox and VyOS and pfSense and Ubuntu and Windows 10 systems demonstrated how each infrastructure level depends on the previous one. The entire network collapses when you make a single mistake between incorrect gateway settings and absent NAT rules. The process of resolving these problems enabled me to understand actual network operations better while learning about the critical infrastructure configuration requirements.

The offensive and defensive simulation activities provided me with essential data which helped me learn new information. The combination of Legion and Hydra and CrackMapExec and Nmap and Metasploit tools provided me with hands-on experience of attacker tools used for enumeration and brute-force attacks and vulnerability testing. The failed attempts demonstrated to me how secure systems function when attackers try to access them while showing me the value of proper system configurations and secure passwords and network segmentation. The defensive training demonstrated to me how Wazuh and Sysmon systems operate together to identify security threats. The Wazuh dashboard displayed process creation events and login failures and registry changes and network activity and brute-force attempts which helped me understand how to track threats in real-time.

Troubleshooting became one of my biggest learning areas. I spent time fixing Wazuh manager issues, improving Sysmon logging, validating pfSense rules, and resolving connectivity errors. This taught me to slow down and confirm assumptions before moving forward. I also learned how important teamwork is in a project like this. Every change one person made affected the entire environment, so we had to communicate constantly, verify each other's work, and solve problems together. By the end, I felt more confident not just technically, but also in project management, collaboration, and staying organized under pressure.

Reflection

This project was the most challenging and rewarding thing I've done in the entire program. There were times when nothing worked - routing didn't pass traffic, pfSense blocked connections we needed, Wazuh agents would not register, and Sysmon logs were missing. I learned not to panic and instead break the problem down layer by layer: check the IPs, check the routes, check the firewall rules, check the agent, check the logs. Every time I solved an issue, I understood the environment a little better.

The Wazuh package mistake alone cost us almost two weeks of repeated uninstalls and cleanups - frustrating, but it taught me how one tiny command can derail everything and why attention to detail is everything.

My experience with Red Team and Blue Team activities provided me with a complete purple team perspective. Running offensive tools like Hydra, CrackMapExec, Legion, Nmap, and Metasploit made me realize how difficult it is to successfully attack a well-configured system. The best moment was seeing another team's real attacks instantly trigger high-severity alerts in our Wazuh dashboard with their source IPs. It proved our setup worked under live pressure. Attacking another lab while defending

our own showed me how quickly misconfigurations get exploited and how powerful proper logging and detection are.

The project required me to acquire essential skills which I needed to learn. I gained technical expertise through working with SIEM tools and learning to manage firewalls and routes and monitor endpoints and solve technical problems. The project allowed me to build my abilities in working with teams and developing my skills in professional communication. The group members encountered multiple instances of incorrect setup which affected everyone, so we needed to stay connected for fast technical problem resolution. It felt like working in a real SOC where collaboration is critical.

If I were to do this project again, I would verify “wazuh-one” installation, automate more of the Windows hardening and Sysmon setup, and keep a shared change-log to avoid accidental misconfigurations. But overall, I’m proud of how much I learned and how well our team worked together. This project gave me real hands-on experience with the same tools, challenges, and workflows used by cybersecurity professionals every day - and it made me even more confident in the path I’m pursuing.

References

Faturrohman, M., Salsabila, A., Mardiah, Z., & Rosadi, A. (2023). Attack in to The Server Message Block (CVE-2020-0796) Vulnerabilities in Windows 10 using Metasploit Framework. JEEMECS (Journal of Electrical Engineering, Mechatronic and Computer Science). 6. 37-44. https://www.researchgate.net/publication/368965977_Attack_in_to_The_Server_Message_Block_CVE-2020-0796_Vulnerabilities_in_Windows_10_using_Metasploit_Framework

Ford, D. (2019, June 30). Proxmox VE – How to home lab part 1. dlford.io. https://www.dlford.io/how-to-home-lab-part-1/

GeeksforGeeks. (2020). Legion Tool in Kali Linux. https://www.geeksforgeeks.org/linux-unix/legion-tool-in-kali-linux/

Hoxha, D. (2020). Using WireGuard VPN. https://www.researchgate.net/publication/345641555_Using_WireGuard_VPN

Inbal, A. (2021). Lateral Movement. https://www.cynet.com/blog/lateral-movement/

Jiang, Y., Meng, Q., Shang, F., Oo, N., Minh, L., Lim, H., & Sikdar, B. (2025). MITRE ATT&CK Applications in Cybersecurity and The Way Forward. https://www.researchgate.net/publication/389090450_MITRE_ATTCK_Applications_in_Cybersecurity_and_The_Way_Forward

MDN contributors. (2025). TRACE request method. https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods/TRACE

Nairuz, A. (2023). CrackMapExec in Action: Enumerating Windows Networks (Part 1). https://medium.com/r3d-buck3t/crackmapexec-in-action-enumerating-windows-networks-part-1-3a6a7e5644e9

Rodríguez, A. (2017, May 30). Using Wazuh to monitor Sysmon events. Wazuh. https://wazuh.com/blog/using-wazuh-to-monitor-sysmon-events/

Samuel, O. (2018, July 11). How to configure Nat the cisco and Vyos Way. Expert Network Consultant. https://www.expertnetworkconsultant.com/configuring/how-to-configure-nat-the-cisco-and-vyos-way/

VyOS maintainers and contributors. (2024). NAT. In VyOS 1.3 documentation. VyOS. https://docs.vyos.io/en/1.3/configuration/nat/index.html

Wattuhewa, S. (2023). Network Scanning with Nmap. https://www.researchgate.net/publication/374135016_Network_Scanning_with_Nmap