8/10/2025

Final Project

CST8808 – Cyber Incident Report

**Submitted By :**  
Adeep Mani  
Kanhay Thakore

Mitanshi Solanki

Rajat Mani

# Table Of Contents

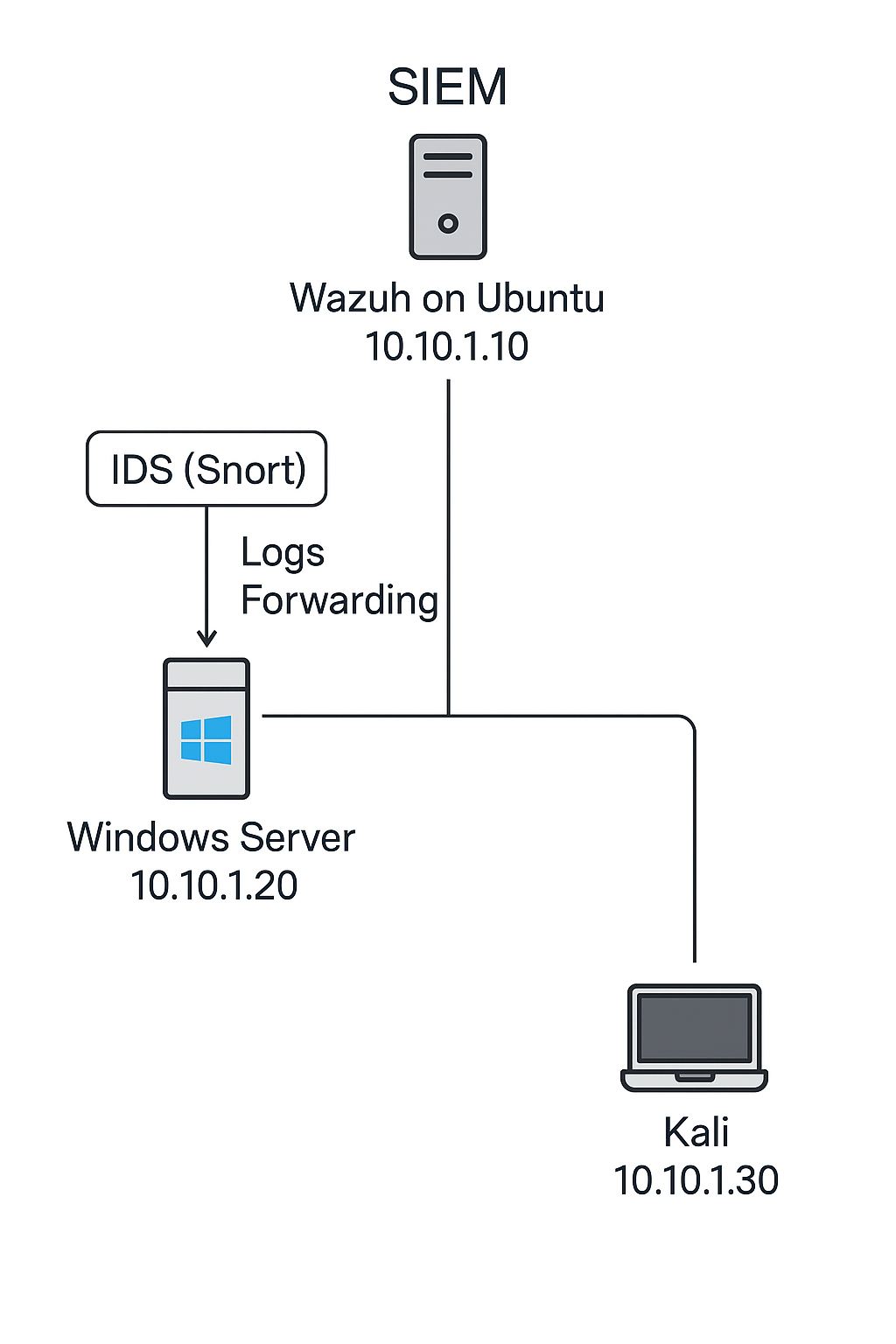
1. Introduction……………………………………………………………….2
2. Network Diagram…………………………………….……………..…..2
3. VMs Configuration ……………………………………………………...3
4. NTP Synchronization ………………………………………………..….6
5. SIEM Configuration……………………………………………….…..…7
6. Attack Simulation and Detection…………………………….…….16
7. File Integrity Monitoring (FIM)…………………………………….....20
8. Memory Analysis with Volatility………………………………..….…25
9. Conclusion………………………………………………………………...29
10. References…………………………………………………………..….…30

**Incident Response Plan Implementation**

**1. Introduction**

This report documents the implementation of an Incident Response (IR) plan for CSA271.com, focusing on log monitoring, threat detection, and attack simulation using a SIEM (Wazuh), Snort IDS, and Volatility for memory analysis. The project involved setting up a virtual lab environment with Kali Linux (attacker), Windows Server (IIS/FTP + Snort), and Ubuntu (Wazuh SIEM).

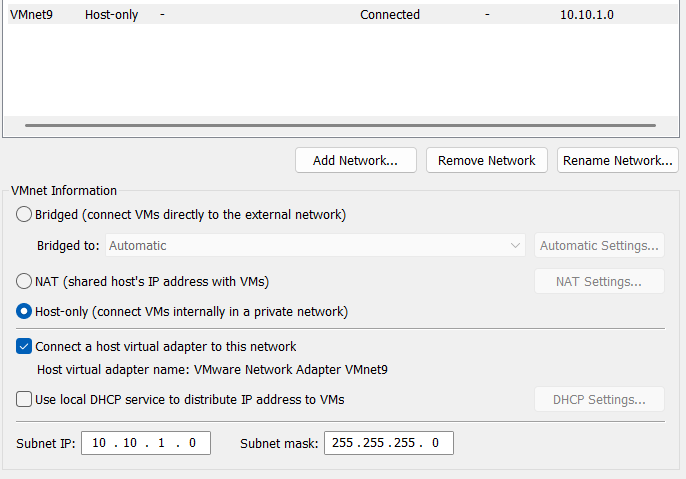
**2. Network Topology & Configuration**

****

**3. VMs Configuration:**

**IP Configuration**

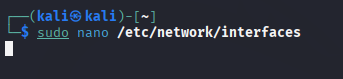
| **Machine** | **IP Address** | **Role** |
| --- | --- | --- |
| **Kali Linux** | 10.10.1.30 | Attack simulation |
| **Windows Server** | 10.10.1.20 | IIS/FTP + Snort IDS |
| **Wazuh (Ubuntu)** | 10.10.1.10 | SIEM (Log collection & alerts) |

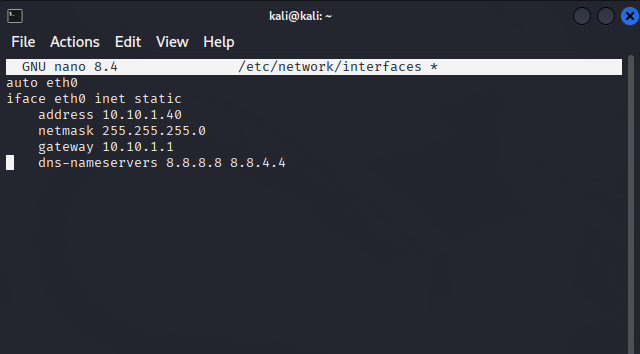
**Configuring Vmware** : Setting Subnet IP and Subnet Mask on Vmnet9

**Key Configurations**

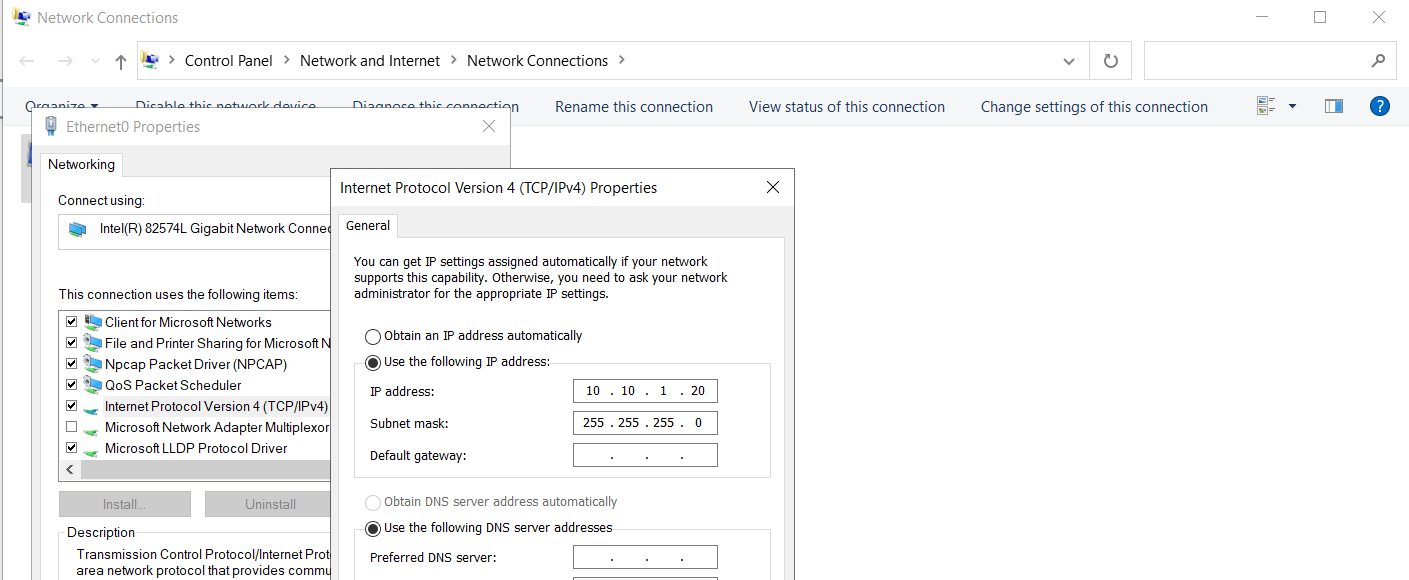
* **Static IPs** configured on all machines

Kali Linux:



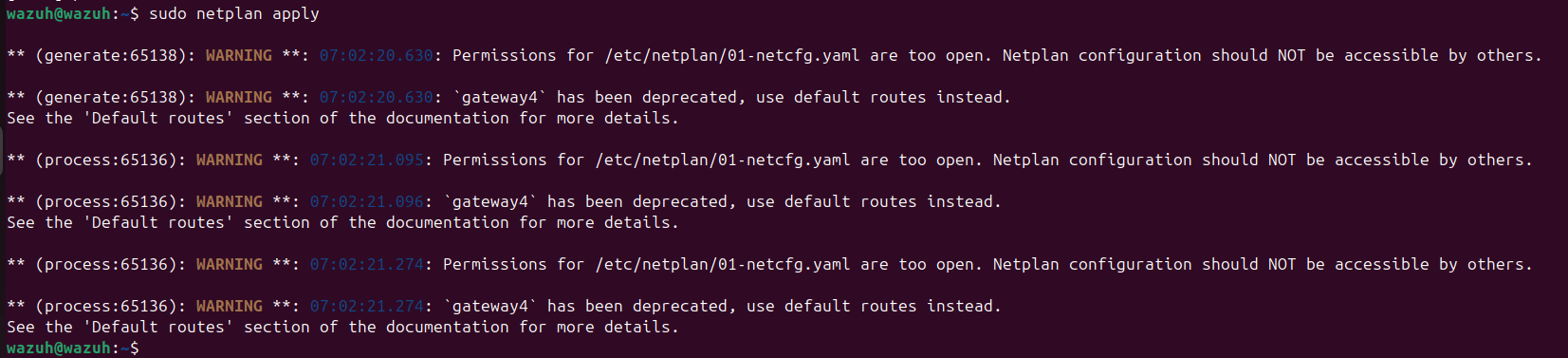


Dns-nameservers can be skipped because we are on host only network. It was only set in case we need to switch to NAT for downloading any tools.

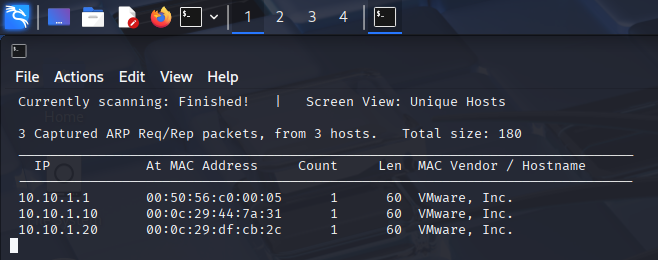
Windows Server (IIS):  


Ubuntu(Wazuh/SIEM):

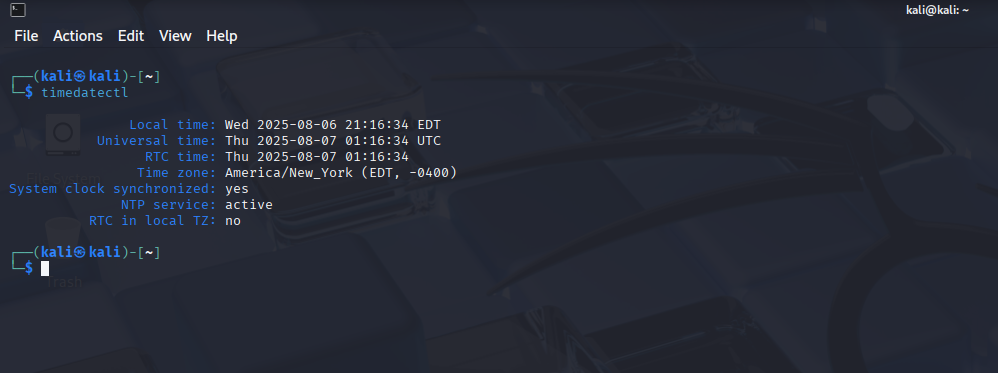
  

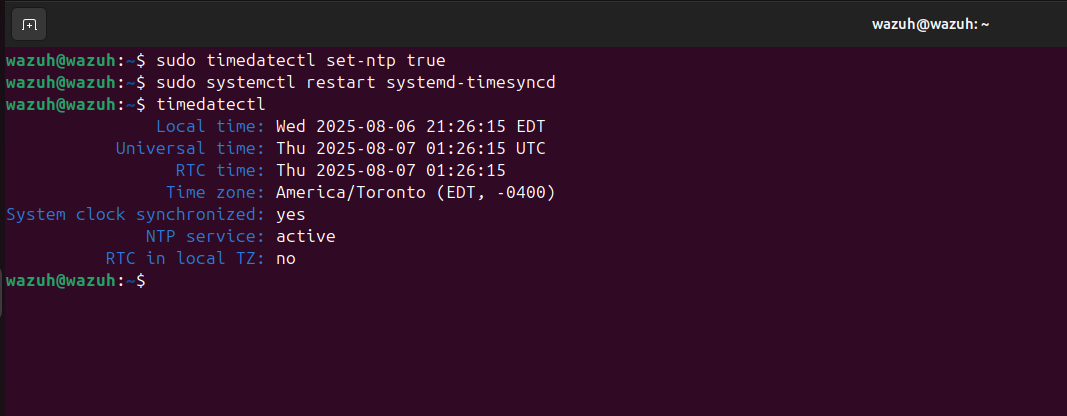
**Confirming if machines are on same network:**



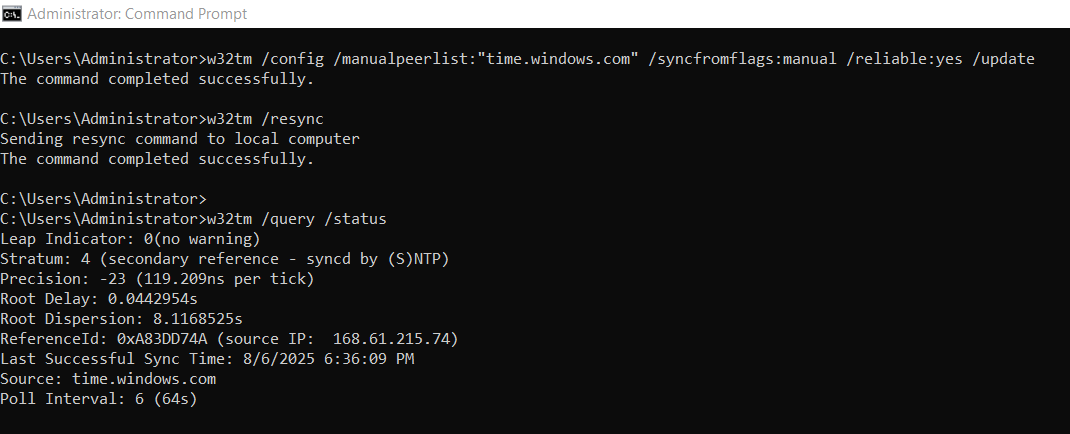
**4. NTP Synchronization on all machines:**

**Kali:**  


**Ubuntu:**



**Windows:**



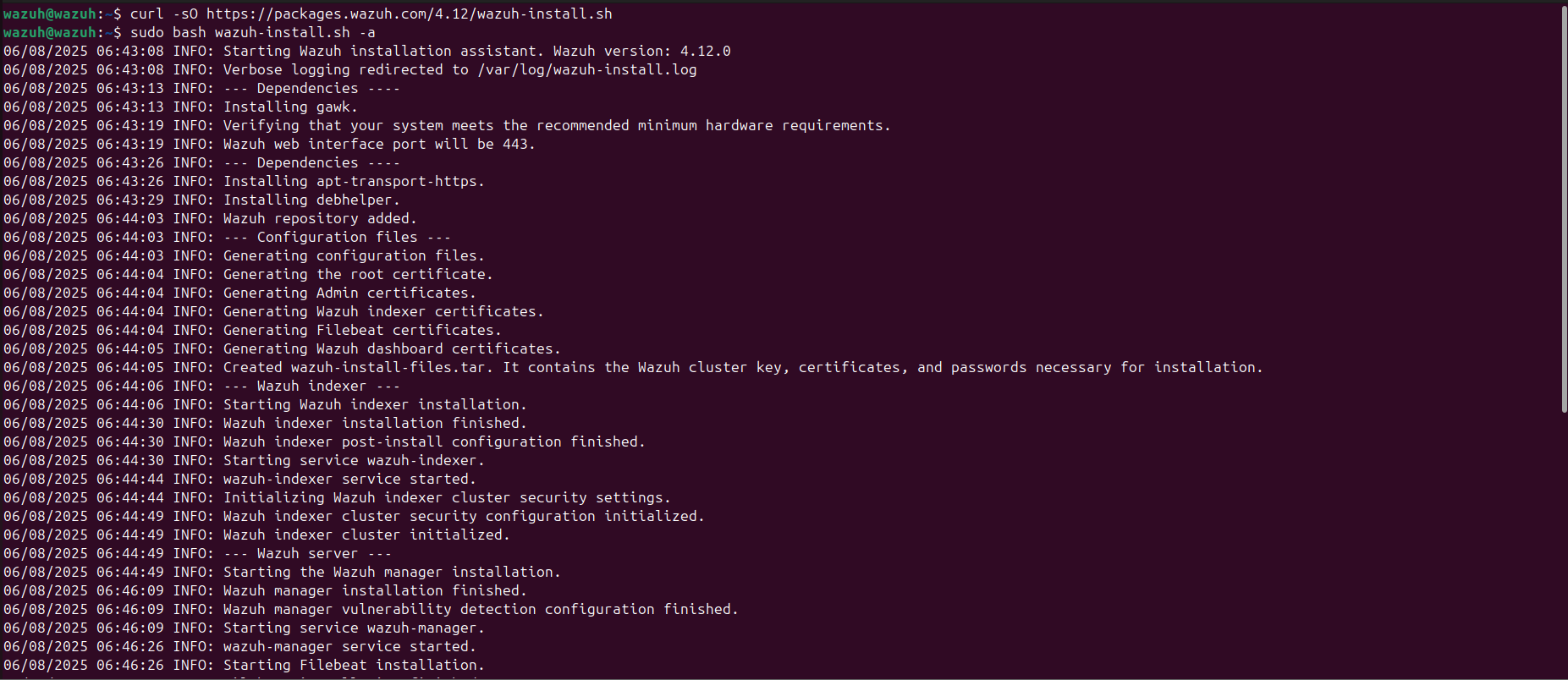
**5. SIEM Setup :**

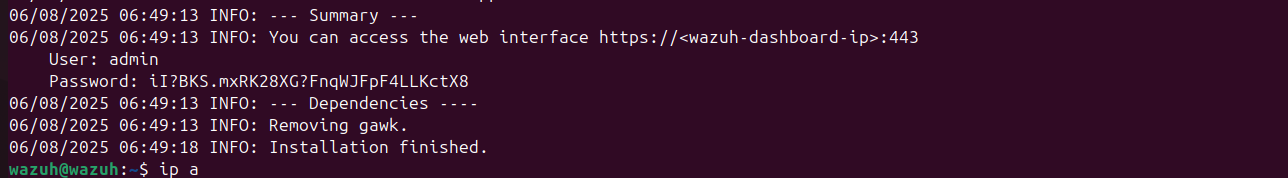
**SIEM Selection Rationale:**

For this project, we selected Wazuh as the SIEM platform instead of alternatives like Splunk, OSSIM, or the ELK Stack. The decision was based on the following factors:

1. **Cost Efficiency** – Wazuh is open-source and free to deploy, avoiding licensing costs associated with Splunk Enterprise or commercial OSSIM implementations, which is ideal for test lab environment.
2. **Feature Set** – Wazuh integrates SIEM, log analysis, and File Integrity Monitoring (FIM) in a single platform. This allowed us to meet both the SIEM and FIM requirements without installing multiple separate tools.
3. **Integration with IDS** – Wazuh can easily ingest alerts from Snort IDS, enabling centralized monitoring of both host-based and network-based events.

**Wazuh Installation:**

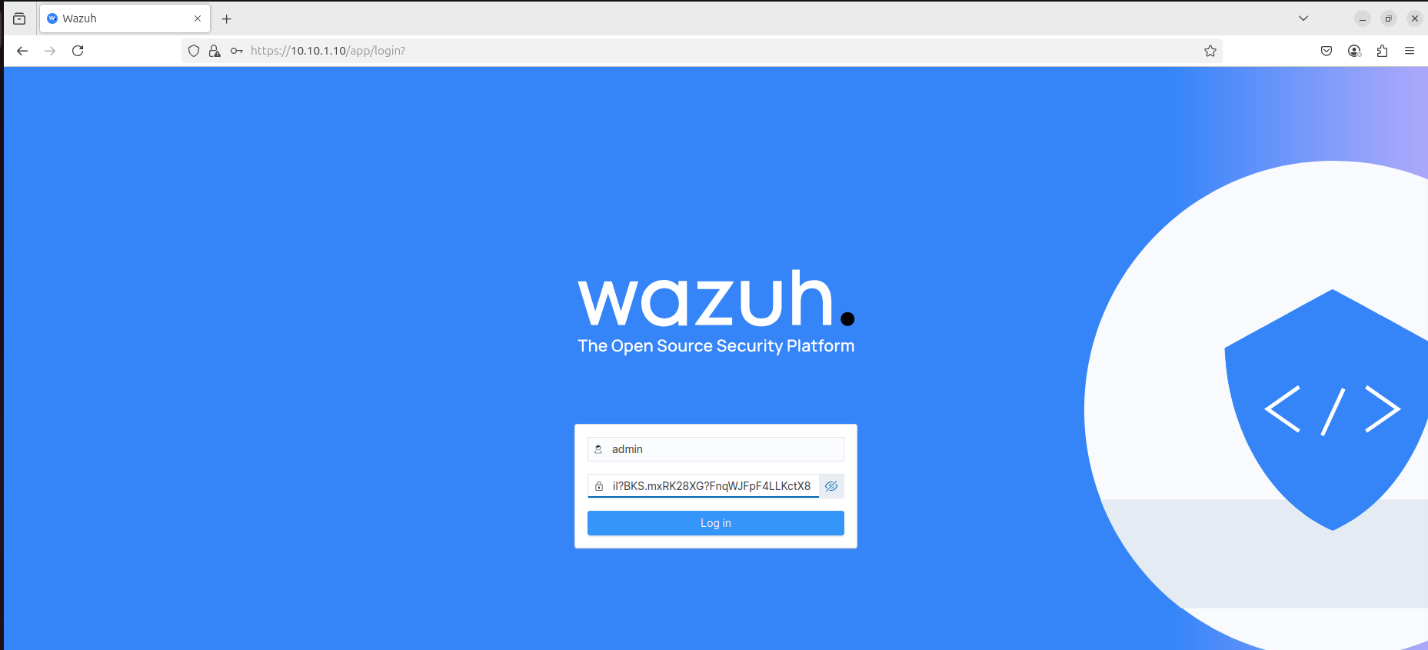


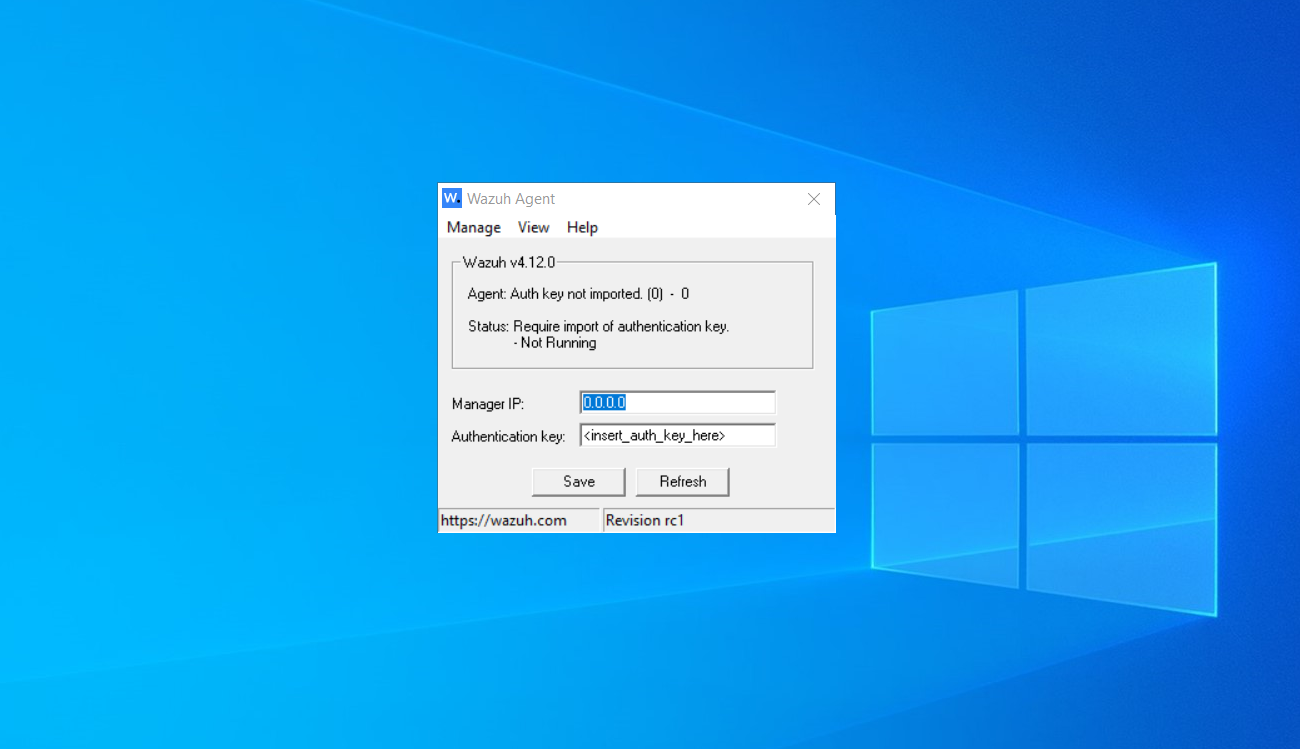


User : admin

Password : iI?BKS.mxRK28XG?FnqWJFpF4LLKctX8

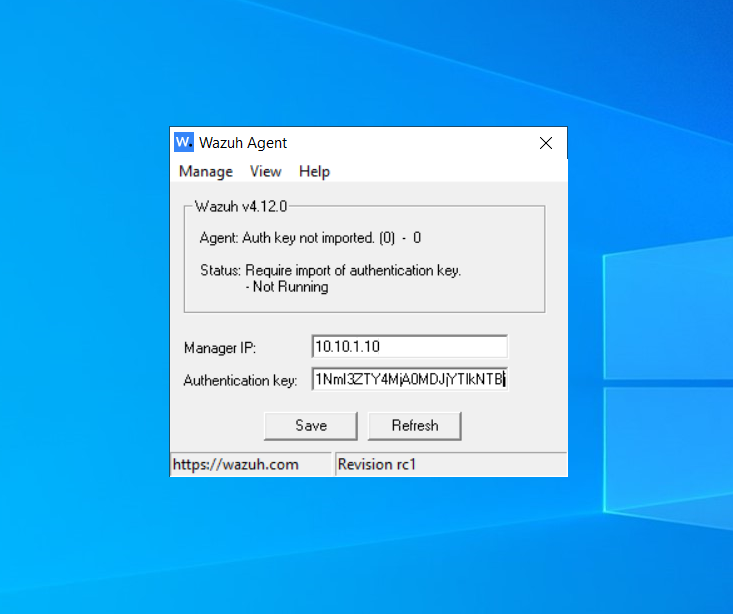
Next, we will open the Wazuh Dashboard on our local ip 10.10.1.10 with the above credentials.

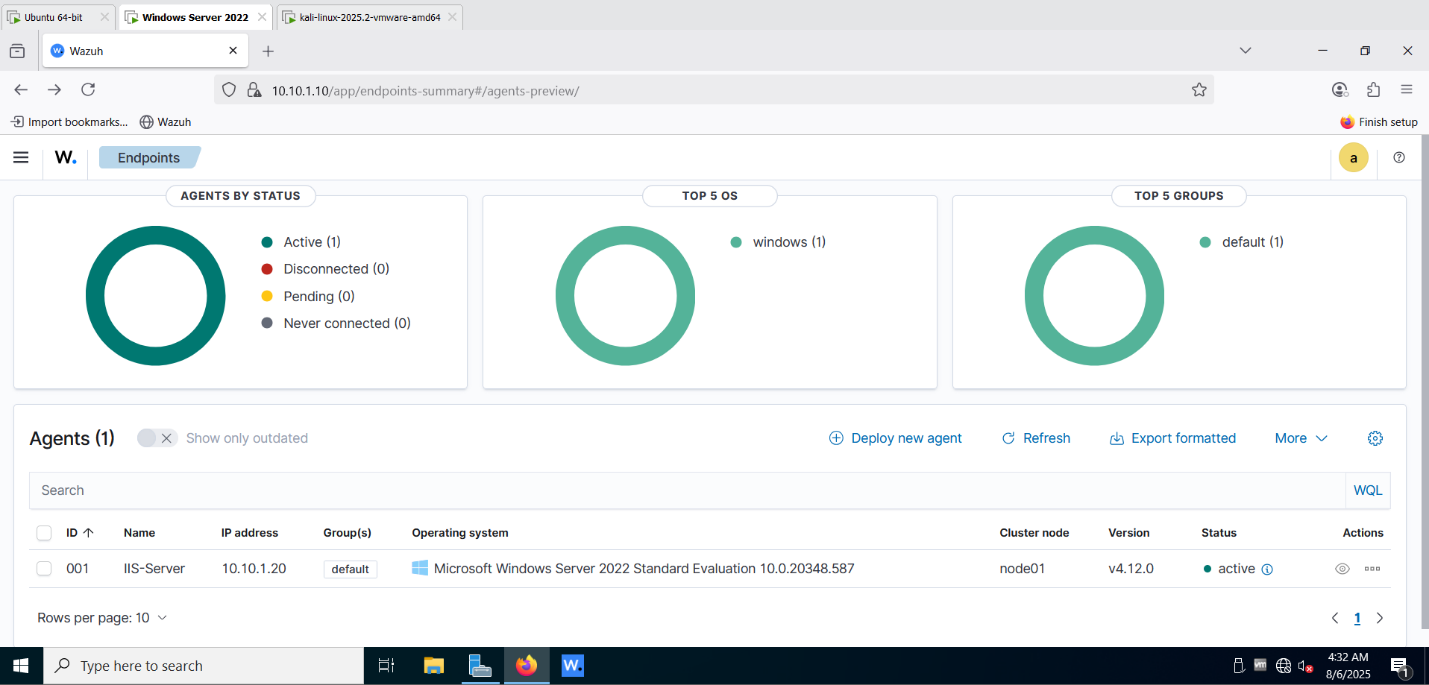
**Dashboard :**  


**Wazuh agent on windows server :**  


Wazuh agent will require manager ip (10.10.1.10) and authentication key which we will generate next.

**Generating the key for agent:**

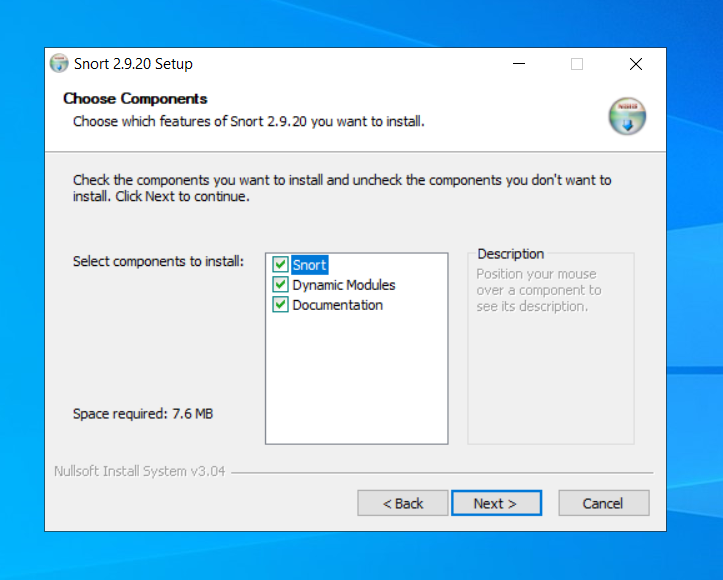
**Wazuh Agent Final Setup:**  


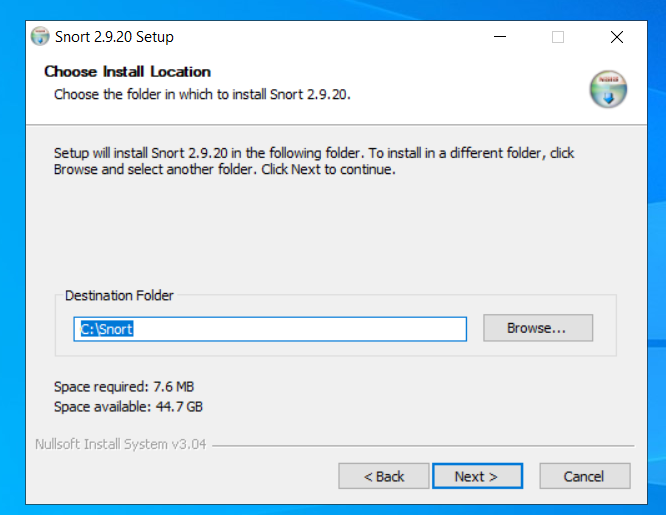
**Confirmed agent connection in Wazuh dashboard** :

**Snort IDS Integration:**

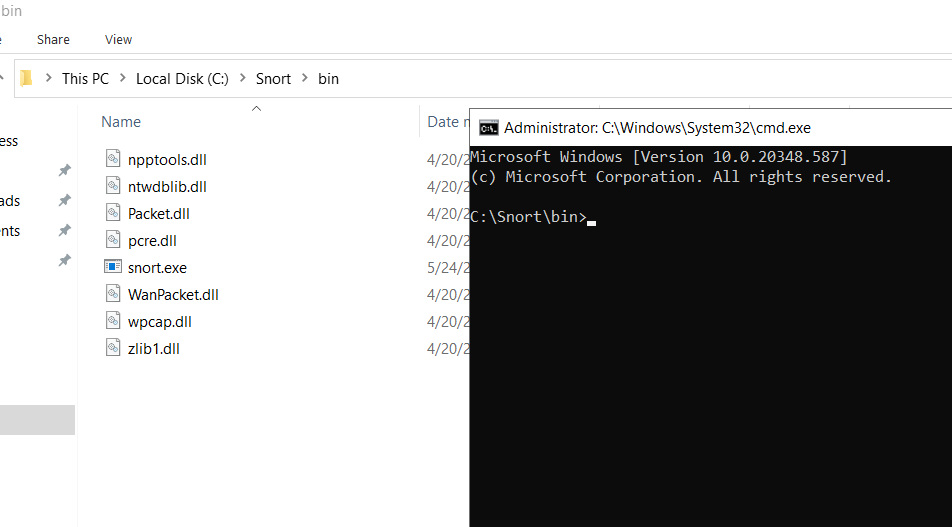
Next, we needed an IDS system to forward network-based alerts from the Windows Server to the Wazuh SIEM. In a real-world deployment, IDS is typically hosted on a dedicated appliance or server to avoid resource contention and ensure optimal performance. However, due to the limited resources available in our lab environment for running multiple VMs, we integrated the IDS directly into the same Windows Server hosting IIS. We selected Snort as our IDS because it is open source, widely used, and offers straightforward integration with Wazuh. Snort’s extensive rule set and community support allowed us to quickly detect simulated network attacks and forward the corresponding alerts to the SIEM for correlation and visualization.

**Snort installation :**

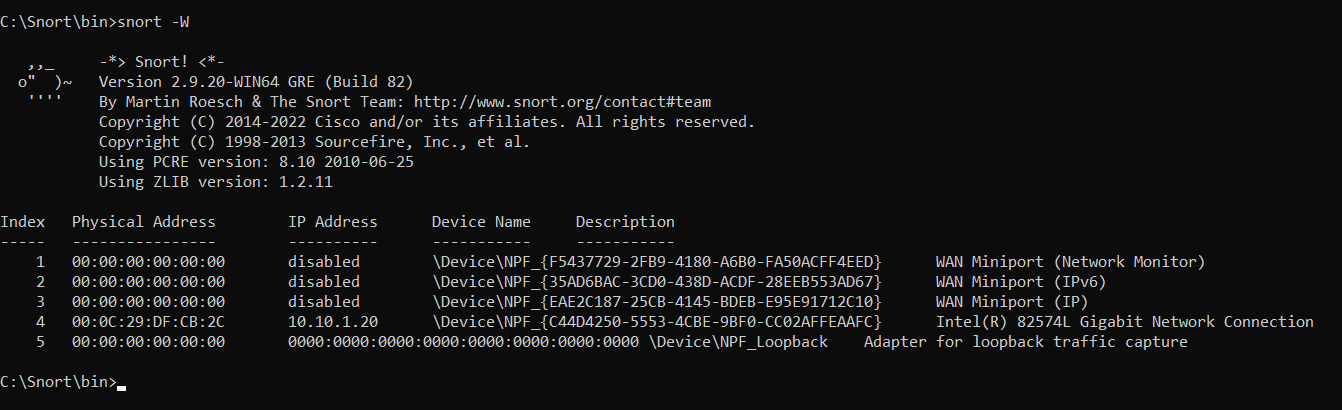




**Running snort :**



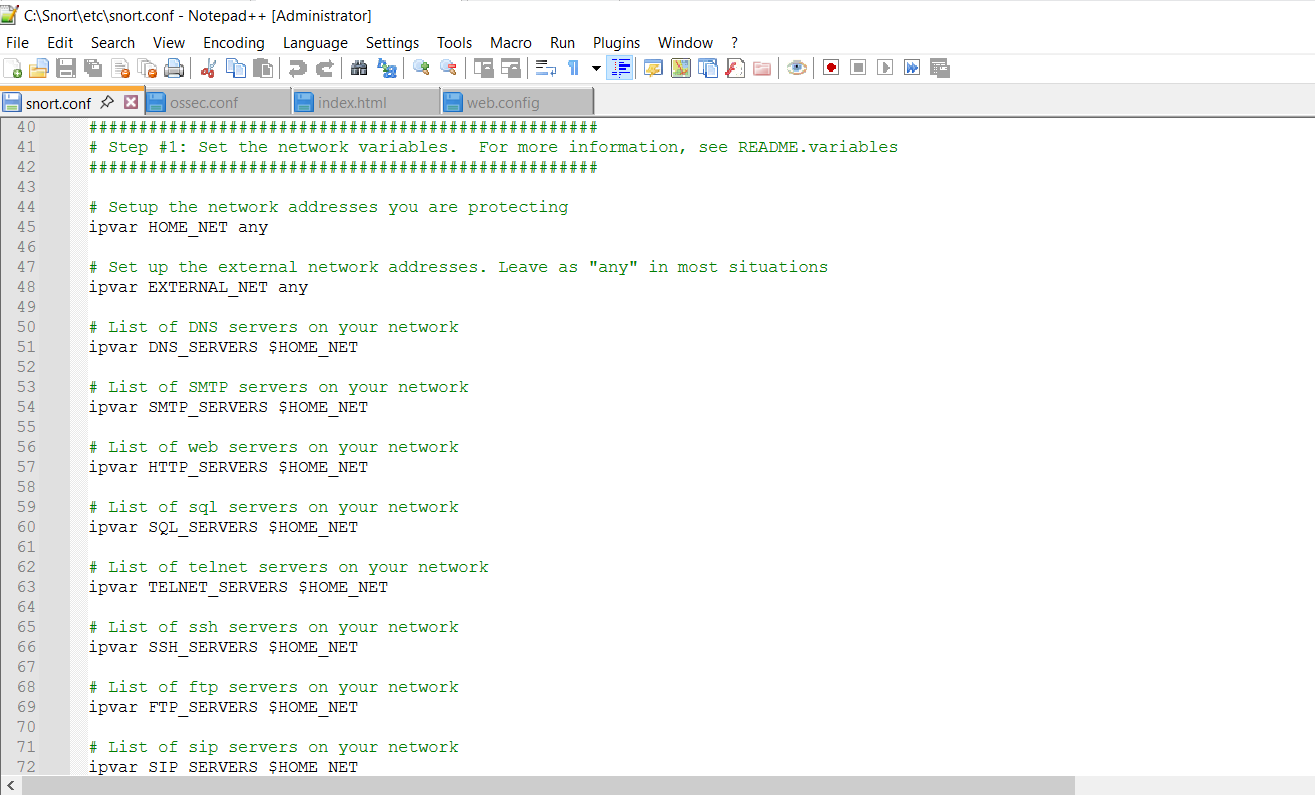
**Checking the index number of interface:**

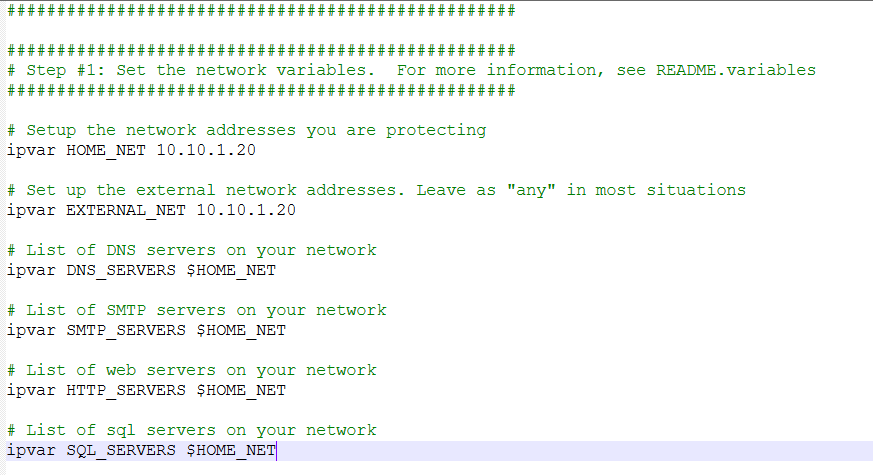


**Snort command :**

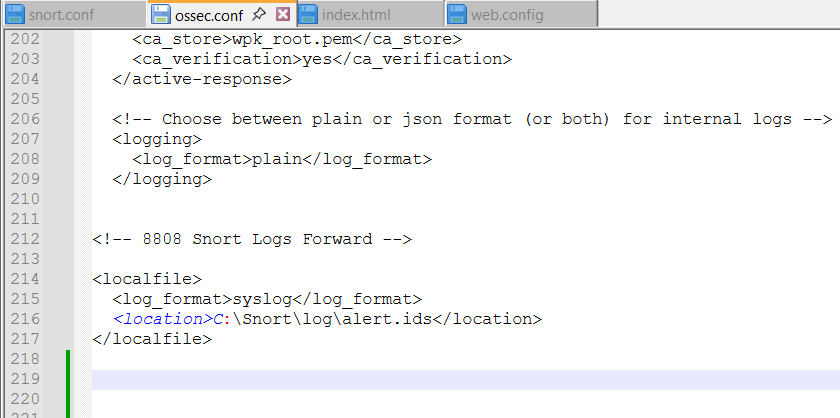


**Snort default config:**

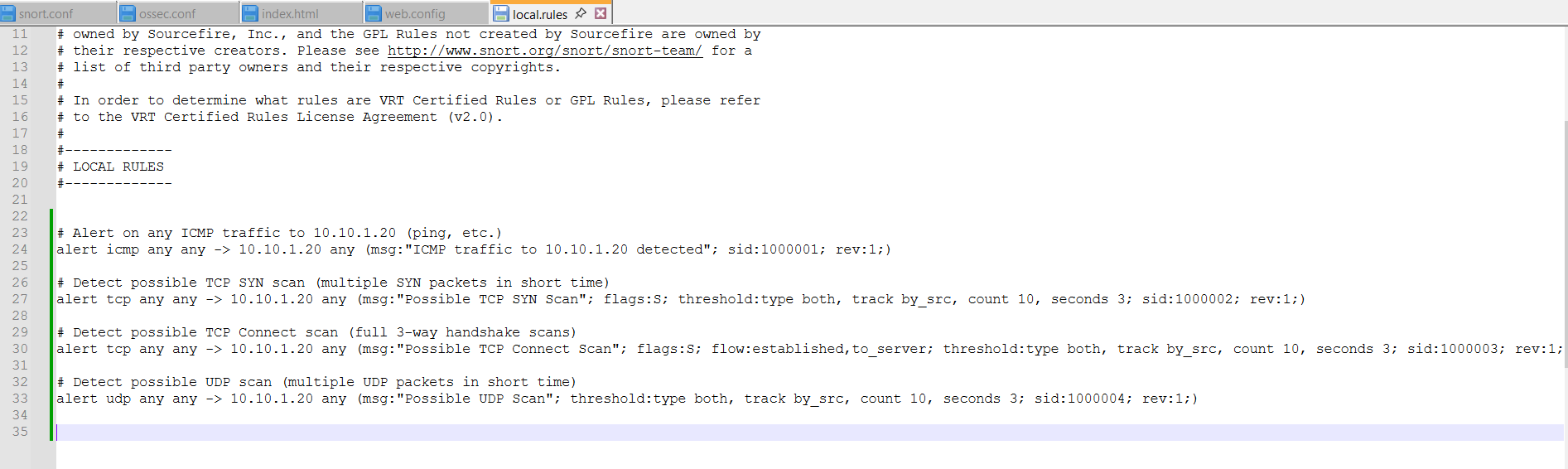


**Configured Snort to monitor home\_net = 10.10.1.20** :

Config can we kept default with “home\_net any” as well , but we did it 10.10.1.20 to keep Kali (which is our attacker) out of scope.

**Forwarded Snort logs to Wazuh via ossec.conf :**  


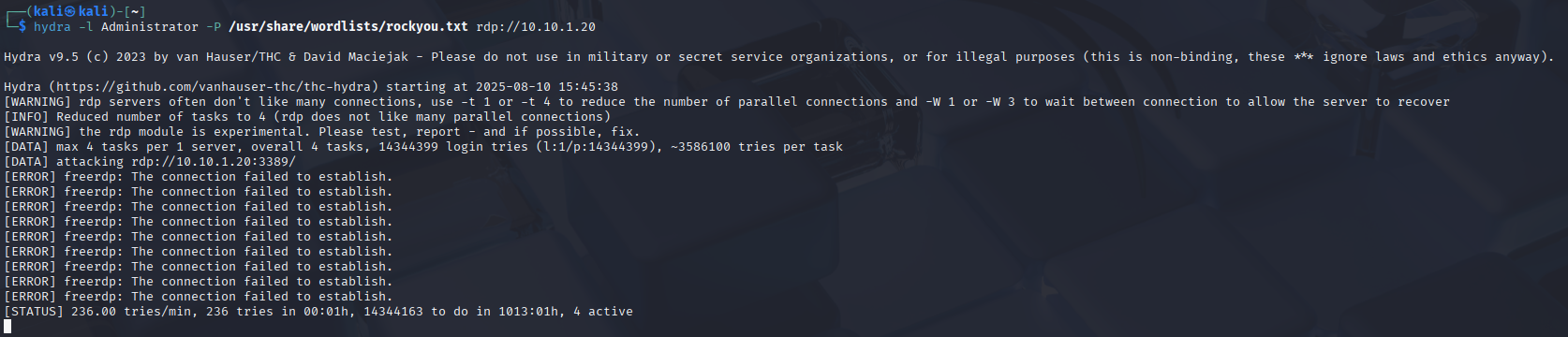
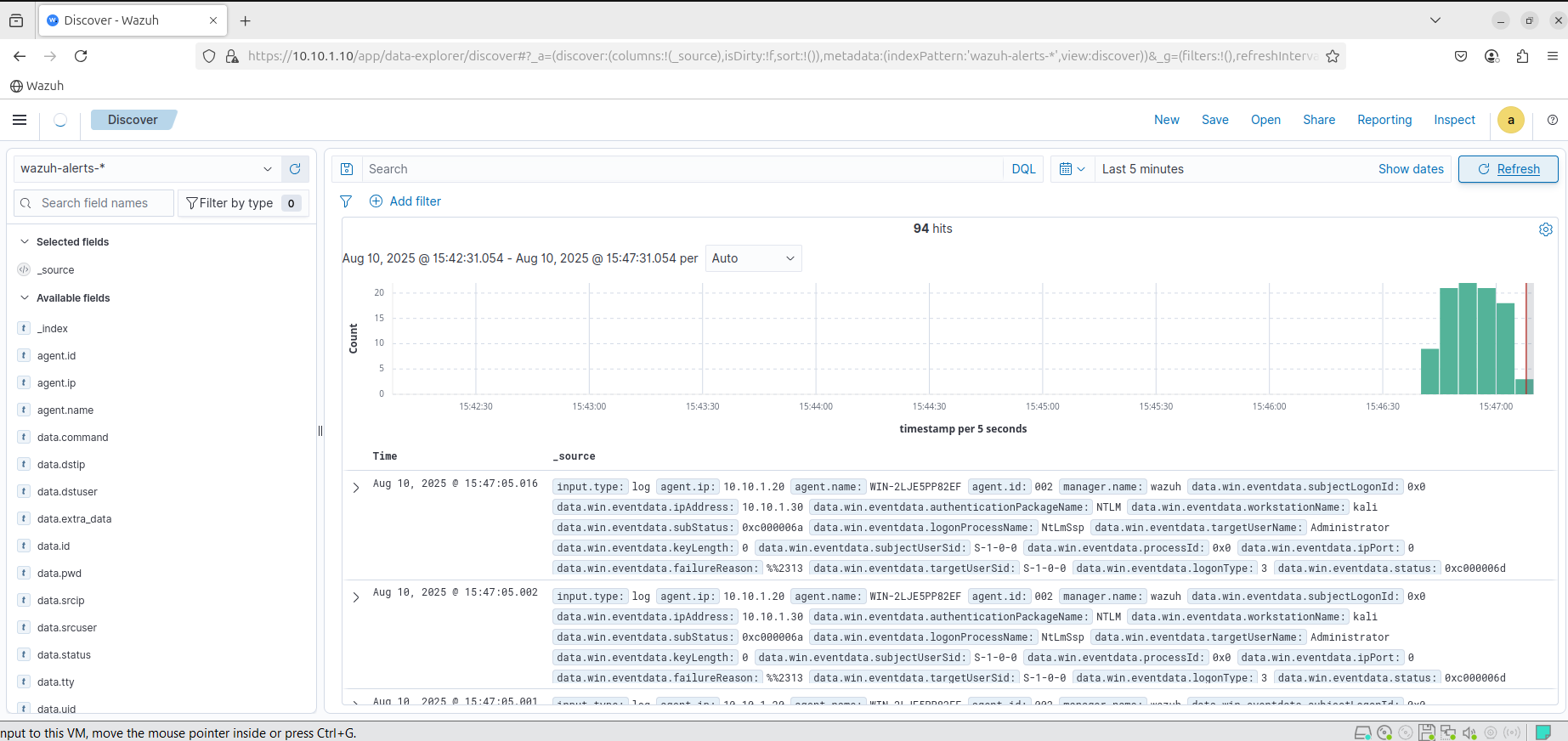
**Updating local.rules:**

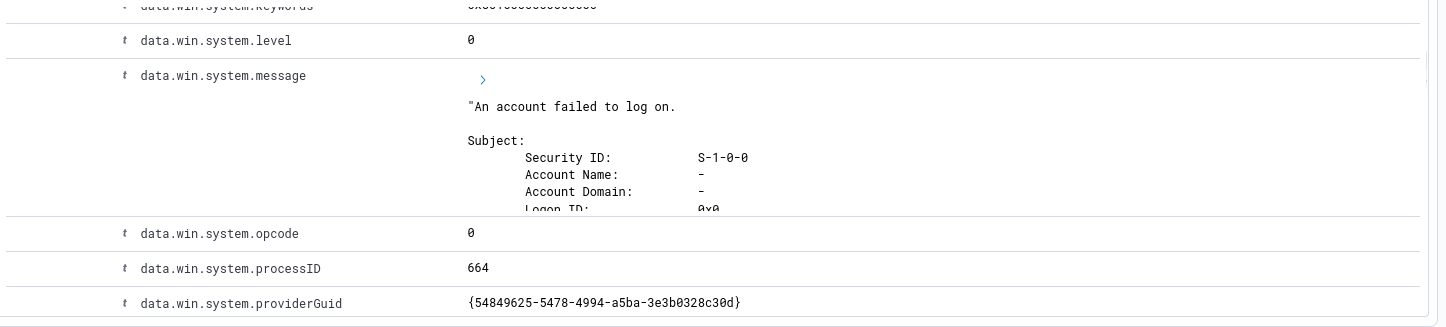


By default, Snort can generate alerts using its preconfigured rule sets without any modifications to the local.rules file. However, we decided to update and customize the local.rules file to ensure that we received precise and relevant alerts for our simulated attacks. This customization allowed us to focus on specific Indicators of Compromise (IoCs) that were part of our project requirements, reducing unnecessary noise and making the alerts in Wazuh more actionable and easier to validate during testing.

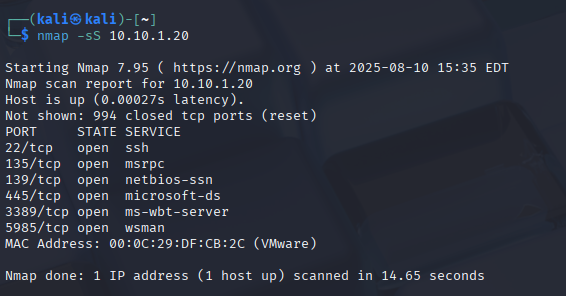
**6. Attack Simulation & Detection**

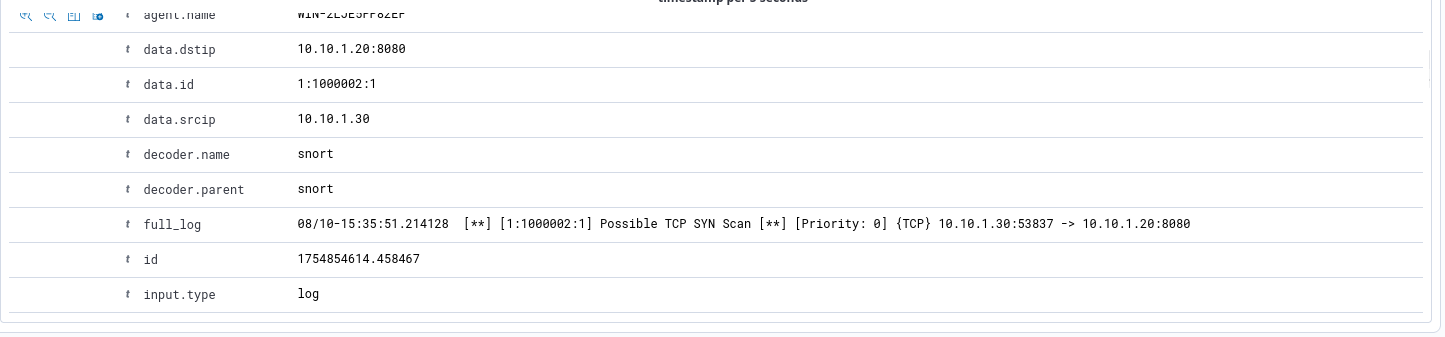
**Brute Force Scan Using RDP:**

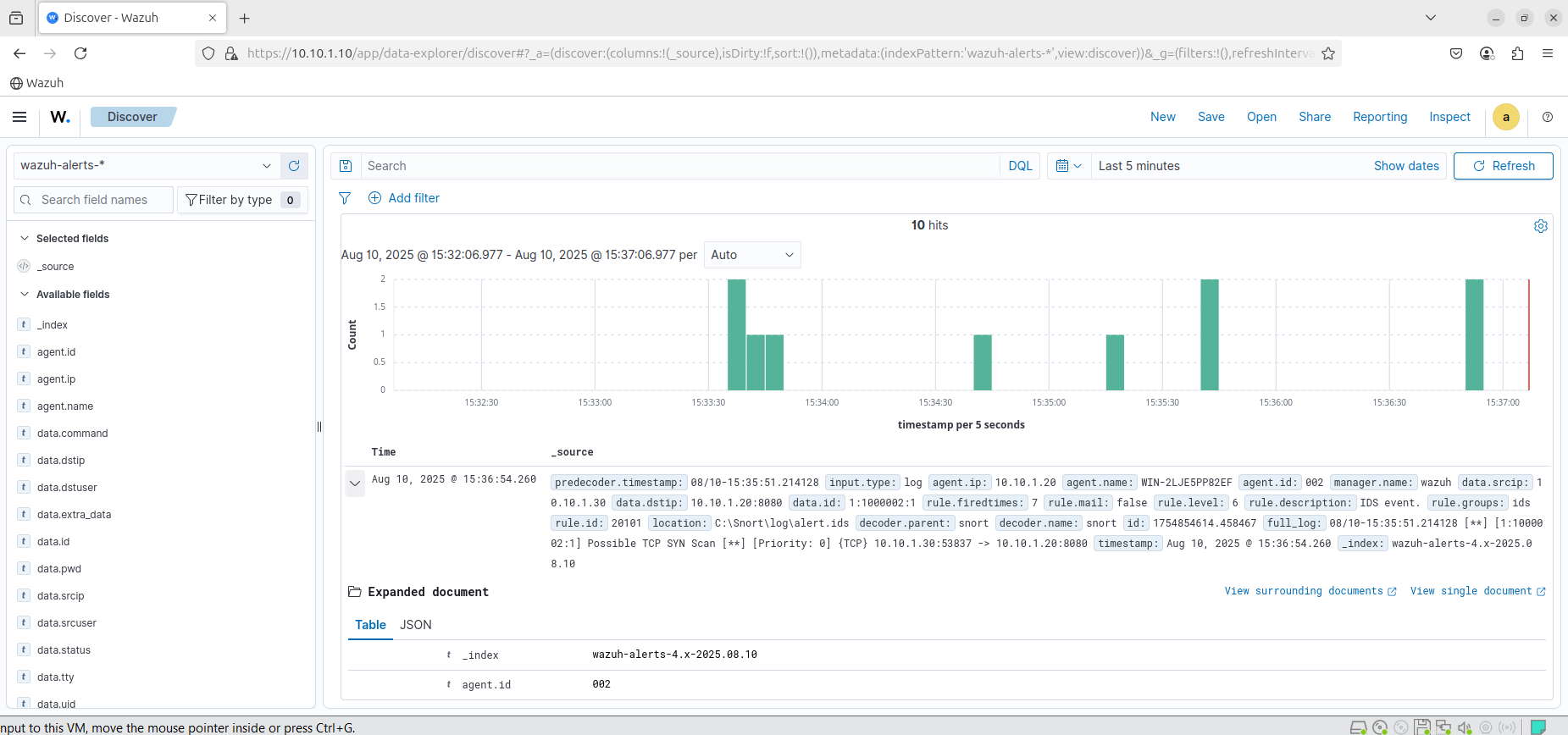
  
  




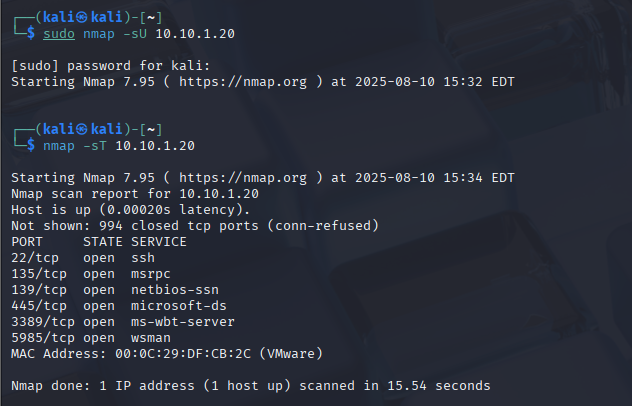
**SYN Scan** :



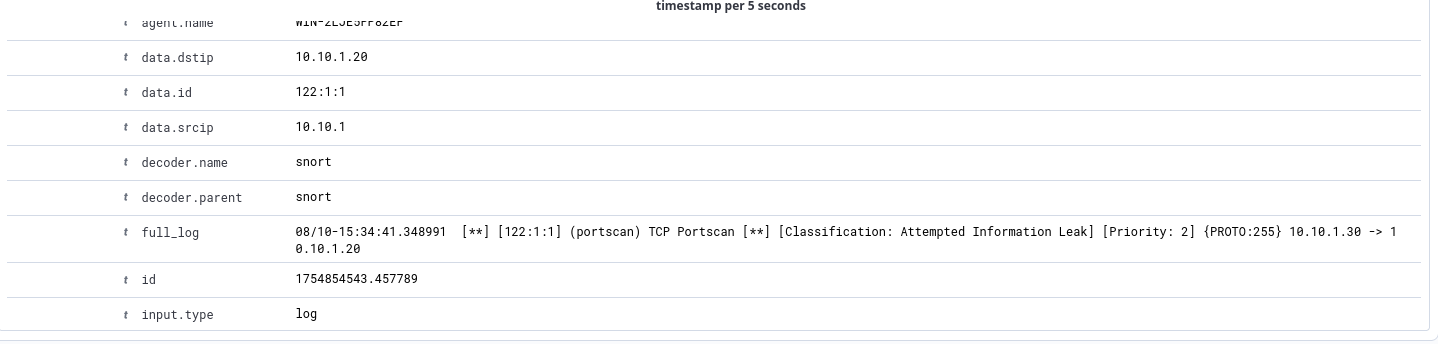




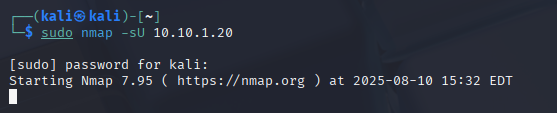
**TCP Scan :**

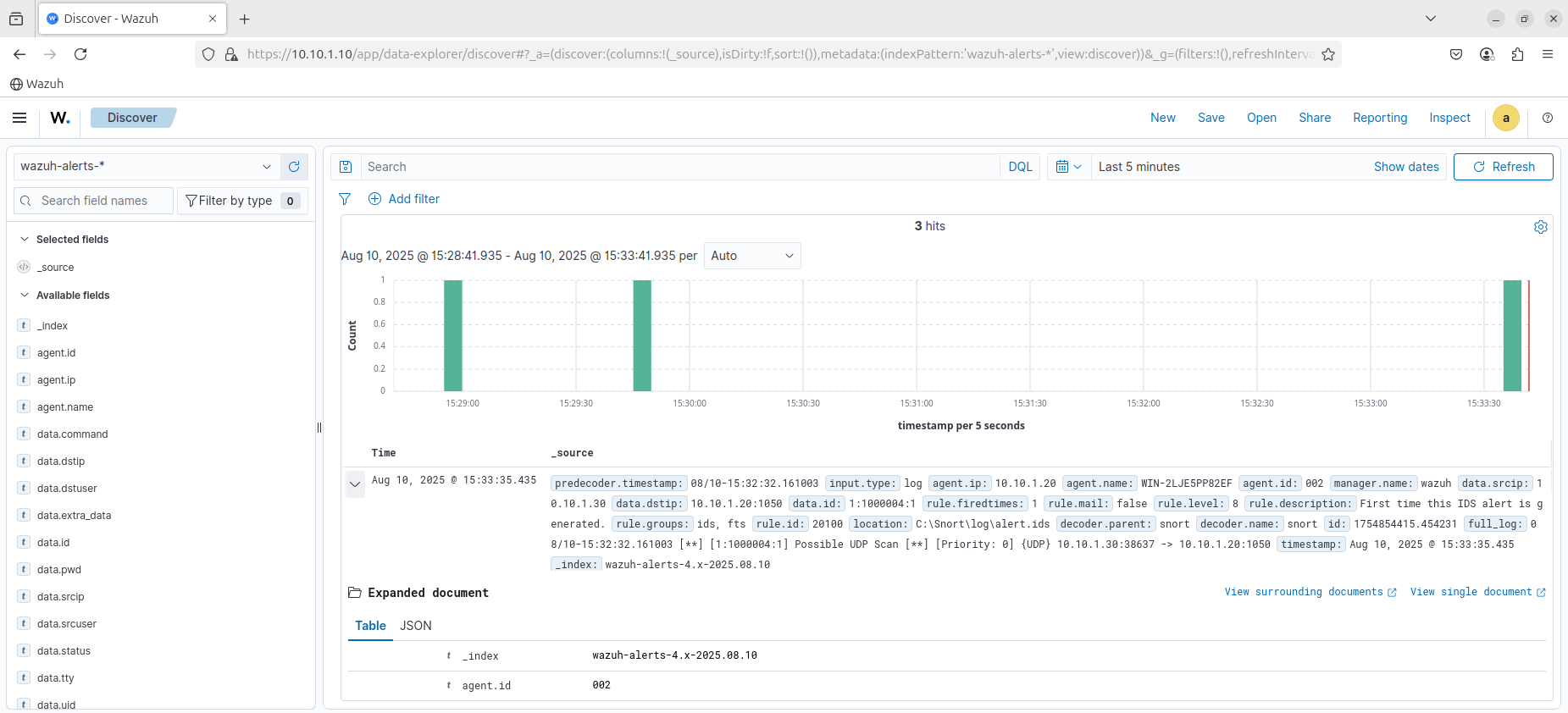
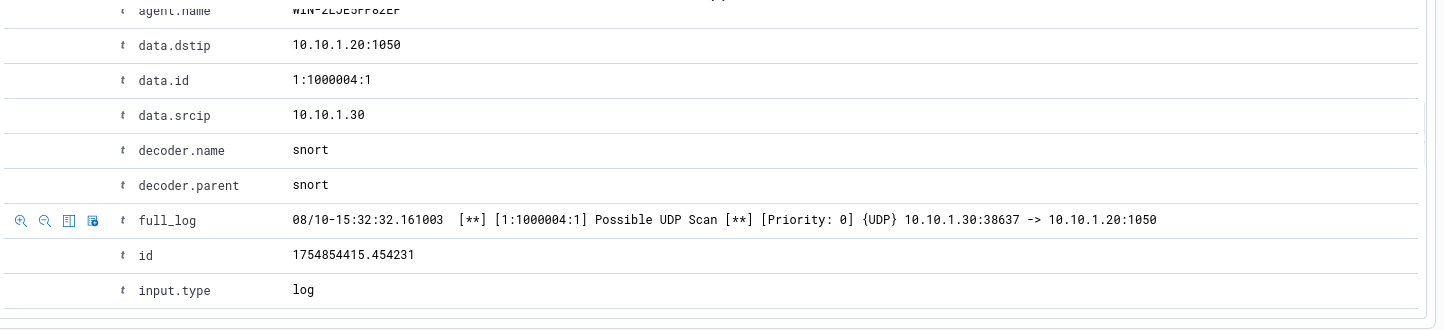






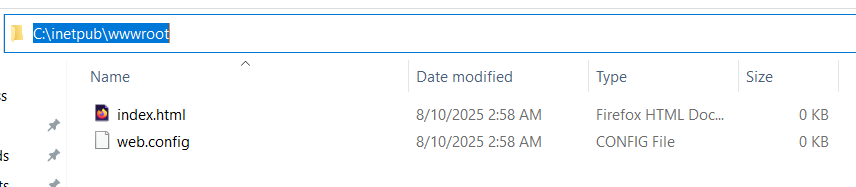
**UDP Scan:**

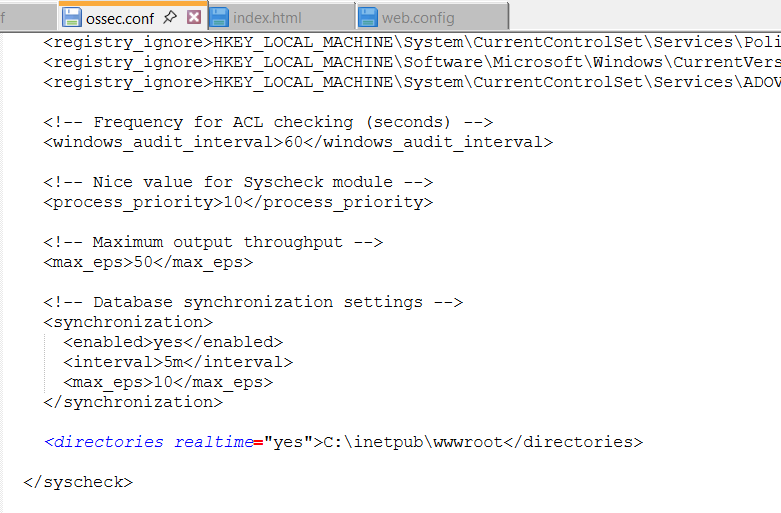


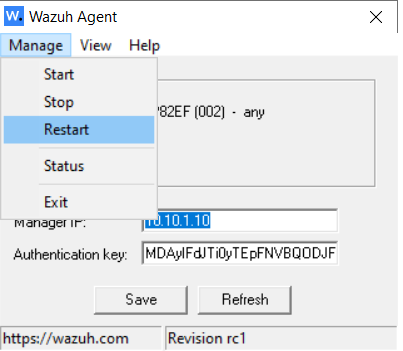
**7. File Integrity Monitoring (FIM)**

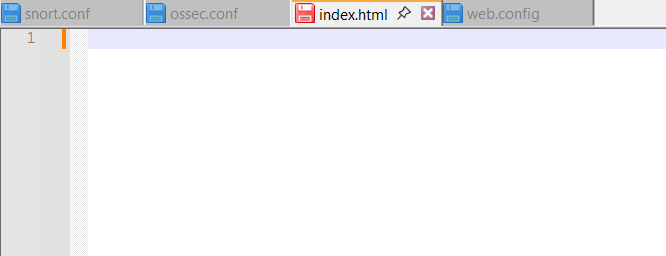
**Monitored files:** index.html & web.config

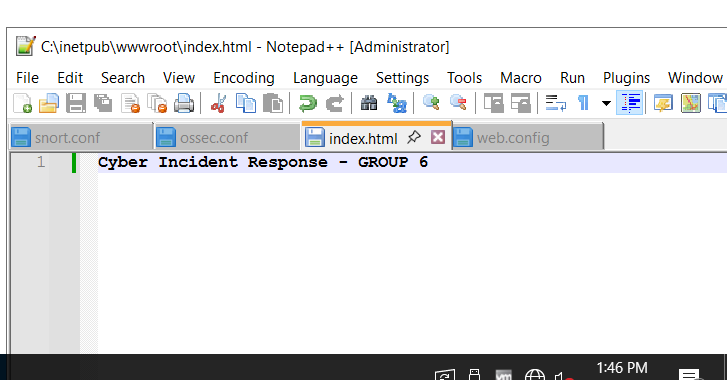


**Configuration:** Modified ossec.conf  

Restarting Wazuh Agent on Server to take changes:

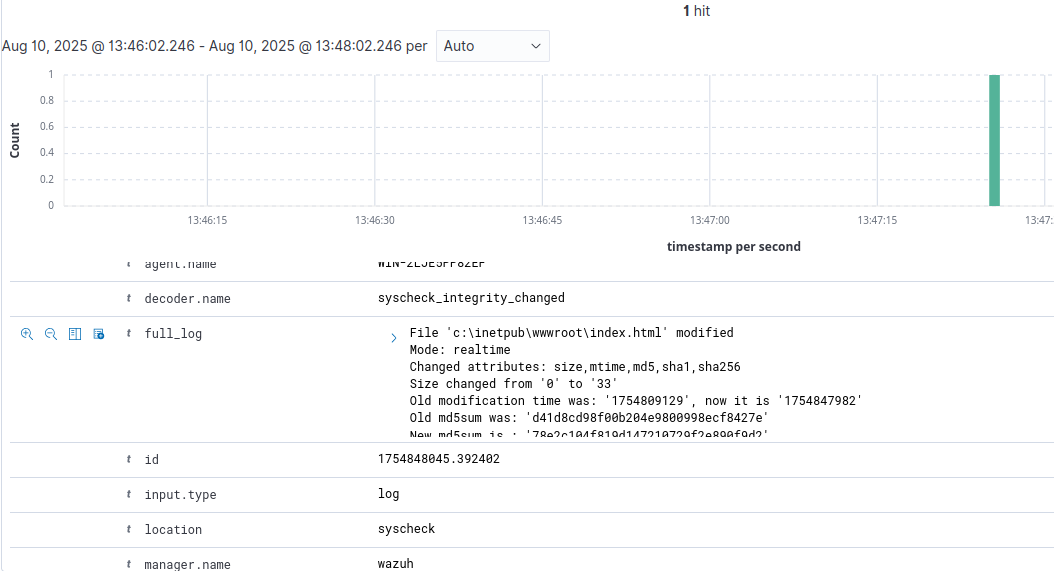


**Test:** Currently the index.html is empty

After editing the file :  


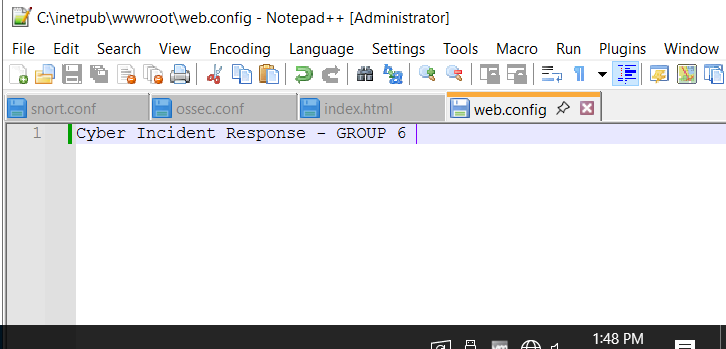
We can see the generated log on Wazuh Dashboard:





full\_log shows the location where the modification was done. Which is the location of index.html in wwwroot folder. Other information like changed attributes, size(old and new) etc. is also visible.

We will do the same test with web.config file :



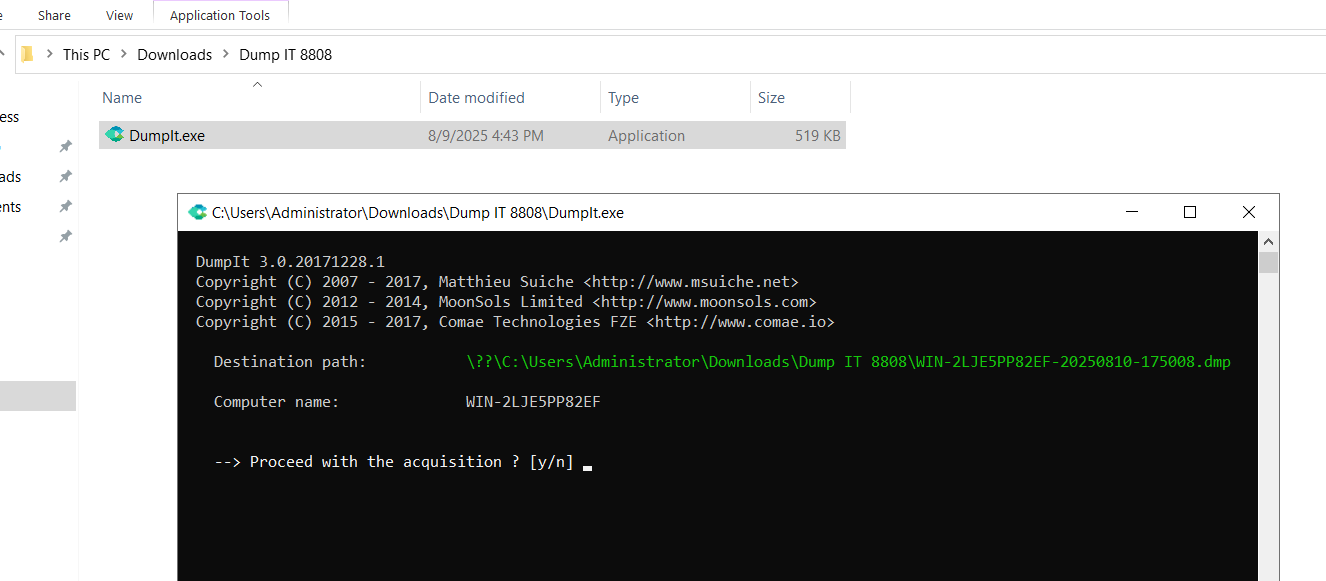




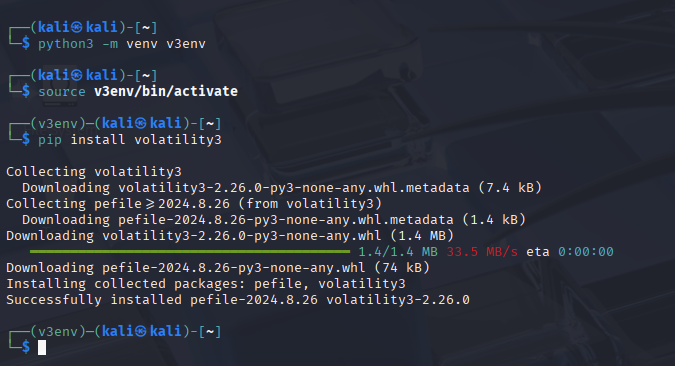
Successful log generated.

**8. Memory Analysis with Volatility**

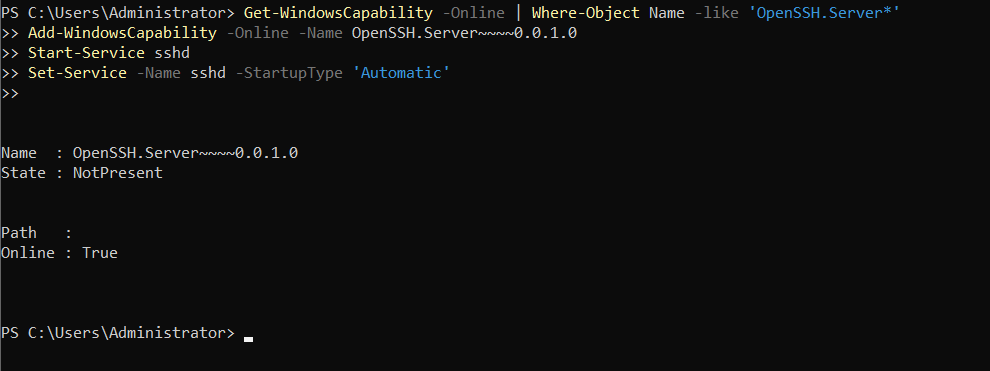
**Dumped memory using DumpIt:**

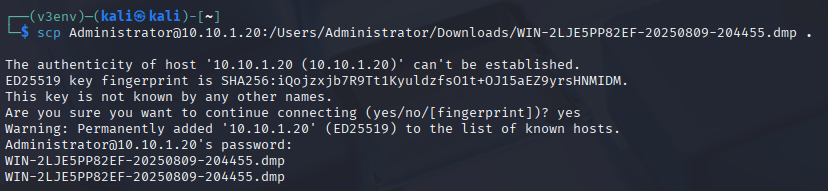


**Setting up Volatility:**



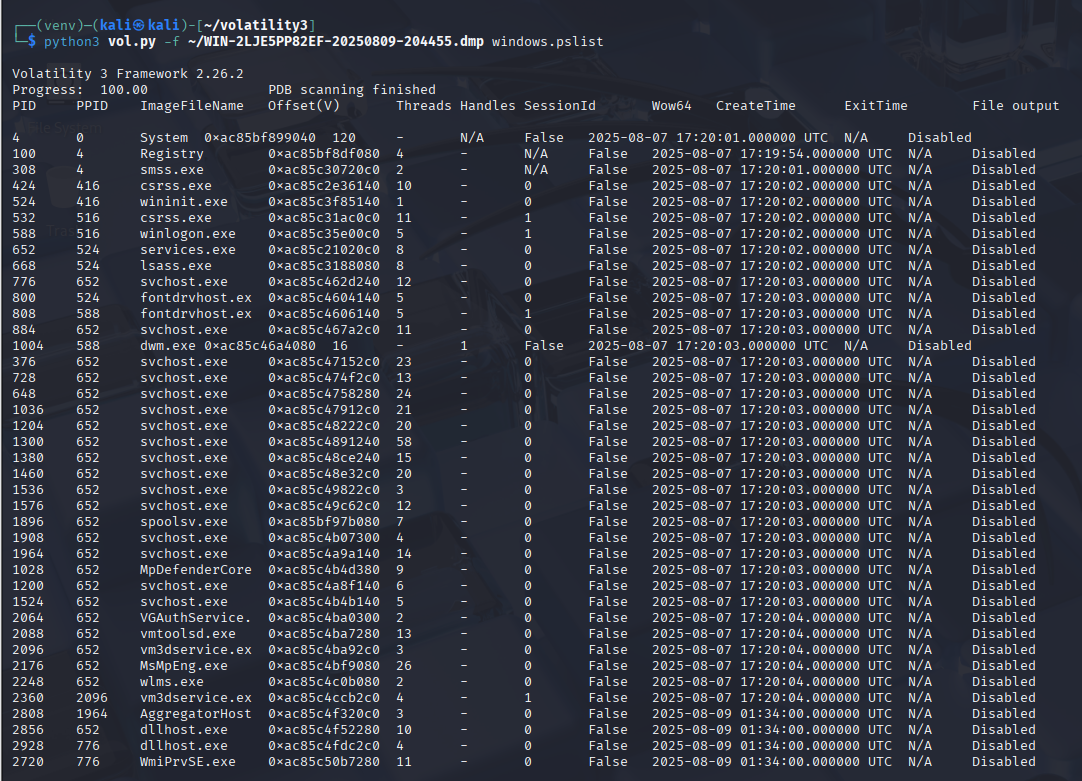
**Installing OpenSSH on Windows Server so SCP can work:**



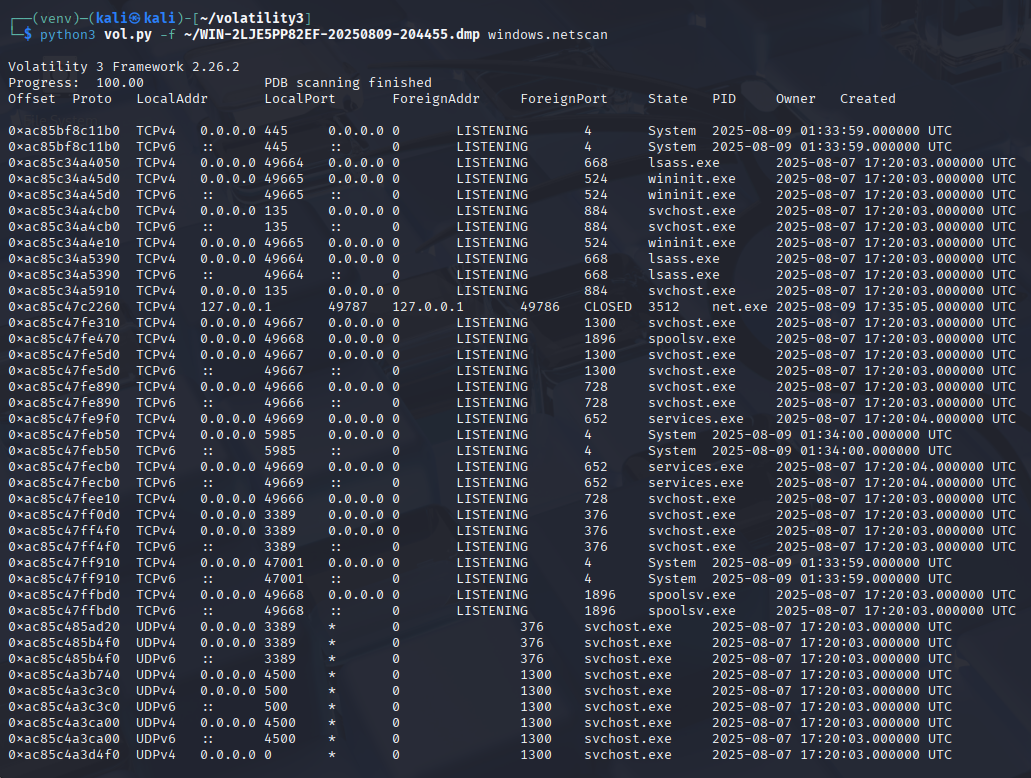
**Transferred .dmp file to Kali via SCP :**

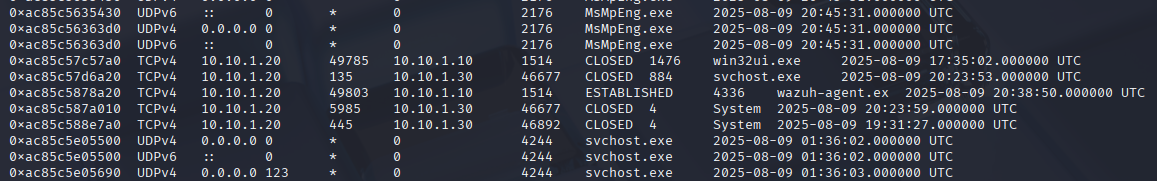
**Command used to inspect .dmp file using volatility:**  


**Windows.pslist dump:**



**Windows.netscan dump:**





We can see the tcp scan done by 10.10.1.30 (kali) to our windows server (10.10.1.20).

Furthermore, we can also notice that a connection was made by wazuh-agent.ex to 10.10.1.10 (SIEM) , confirming logs we sent in real-time to wazuh server.

**9. Conclusion**

This project successfully met all the objectives of the CST8808 Final Project by building a functional incident detection and response environment for CSA271.com. Using Wazuh SIEM integrated with Snort IDS, we detected and logged all four required Indicators of Compromise. Brute force login attempts, SYN scans, TCP scans, and UDP scans.

File Integrity Monitoring ensured that unauthorized changes to key web files were immediately flagged, and memory analysis with Volatility confirmed in-memory evidence of the attacks.

Despite resource constraints that prevented us from running a fully dedicated machine for each role, the lab environment was carefully configured to mimic real-world operations while maintaining performance. This allowed us to validate log forwarding, alert generation, and correlation within the SIEM dashboard under realistic attack conditions.

In the end, this project proved that with the right planning and configuration, open-source tools like Wazuh and Snort can deliver robust, enterprise-level security monitoring and incident response without the cost of commercial licenses. Making them both practical and powerful for organizations with limited budgets.

**10. References**

1. **Wazuh Documentation** – *Installation, configuration, and integration guides*  
   Wazuh, Inc. (2025). *Wazuh documentation*. Retrieved from:  
   <https://documentation.wazuh.com/>
2. **Snort Official User Manual** – *Snort configuration, rule writing, and best practices*  
   Cisco Systems, Inc. (2025). *Snort 2.x User Manual*. Retrieved from:  
   <https://www.snort.org/documents>
3. **Volatility 3 Framework** – *Memory forensics tool usage*  
   Volatility Foundation. (2025). *Volatility 3 documentation*. Retrieved from:  
   <https://volatility3.readthedocs.io/>
4. **Microsoft Windows Server 2016 Documentation** – *Networking, IIS, and NTP setup*  
   Microsoft Corporation. (2025). *Windows Server documentation*. Retrieved from:  
   <https://learn.microsoft.com/en-us/windows-server/>
5. **National Institute of Standards and Technology (NIST)** – *Incident Response best practices*  
   Cichonski, P., Millar, T., Grance, T., & Scarfone, K. (2012). *Computer Security Incident Handling Guide* (NIST SP 800-61 Rev. 2).  
   <https://doi.org/10.6028/NIST.SP.800-61r2>
6. **Kali Linux Official Documentation** – *Penetration testing and network scanning tools*  
   Offensive Security. (2025). *Kali Linux documentation*. Retrieved from:  
   <https://www.kali.org/docs/>
7. **Open Source Security (OSSEC)** – *Log-based intrusion detection concepts*  
   Trend Micro, Inc. (2025). *OSSEC documentation*. Retrieved from:  
   <https://www.ossec.net/docs/>