

Brute-Force Attack Detection via Wazuh SIEM

Security Monitoring and Alert Correlation on Windows Server 2019

Prepared by

Yiğit Yücel

Cybersecurity Intern | Blue Team Specialization

Institution / Lab

Privia Security – Cybersecurity Internship Program

Virtual SOC Environment | Ubuntu – Windows Server – Kali Linux

Date

April 29, 2025

1. Executive Summary

This project involved deploying a complete SIEM environment from scratch using Wazuh and conducting a brute-force attack simulation against a Windows Server target.

Objectives:

- Detect attacks through centralized monitoring
- Generate relevant security alerts
- Document incidents according to professional SOC standards

Scope:

- Wazuh SIEM installation and configuration
- Brute-force attack simulation against Windows Server
- Attack detection and log management

Results:

- Wazuh successfully detected the brute-force attack
- Generated MITRE ATT&CK framework-compliant alerts
- Validated SOC operational processes

Note: All steps are supported with screenshots and technical logs.

Key Highlights:

- ✓ SIEM deployment and integration
- ✓ Realistic attack scenario
- ✓ Professional incident management procedures

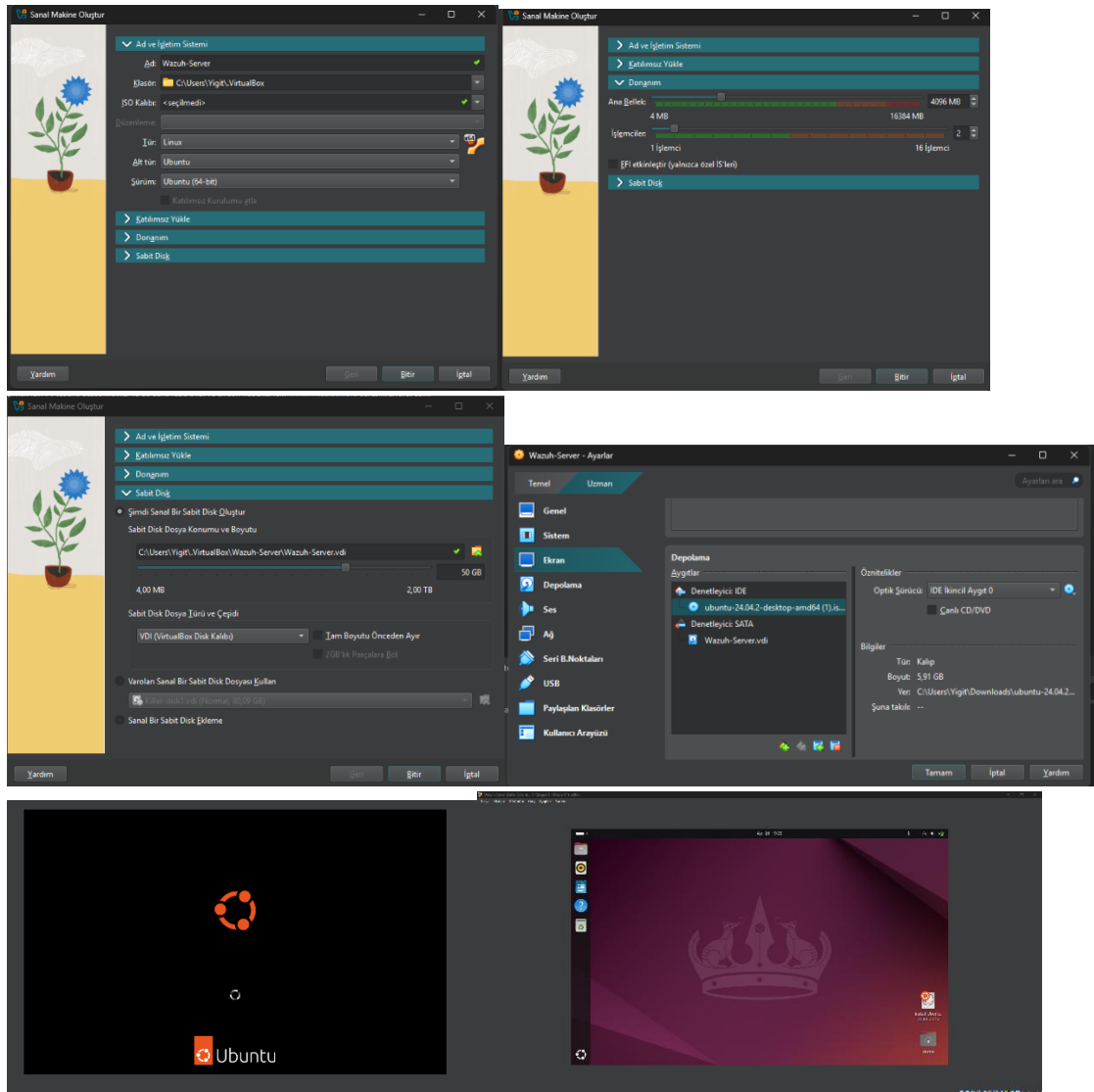
Refer to relevant sections of the report for additional technical details.

2. Environment Setup

2.1 Virtual Machines and Installation

The test environment consisted of three virtual machines created using Oracle VirtualBox:

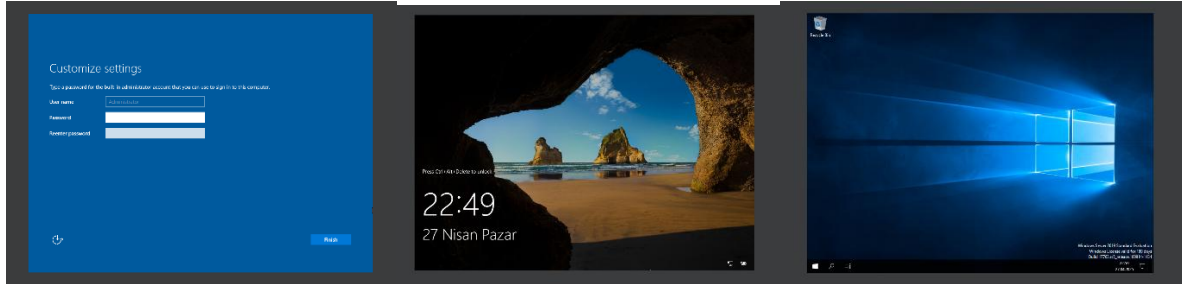
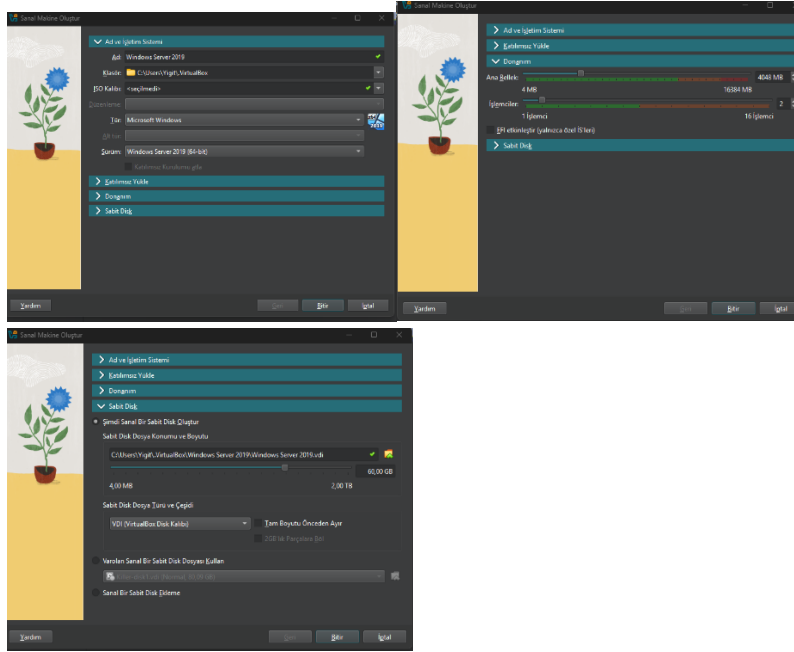
- **Ubuntu Server 22.04 LTS – Used as the Wazuh Manager and Dashboard host.**



Ubuntu Installation Summary:

- ✓ Chose language and keyboard layout.
- ✓ Skipped internet connection.
- ✓ Selected interactive installation with default apps.
- ✓ Did not install third-party software.
- ✓ Erased disk and installed Ubuntu.
- ✓ Created user account and set timezone to Istanbul.
- ✓ Confirmed setup and installed system.
- ✓ Logged in and skipped Ubuntu Pro and data sharing offers.
- ✓ Installation finished, system ready to use.

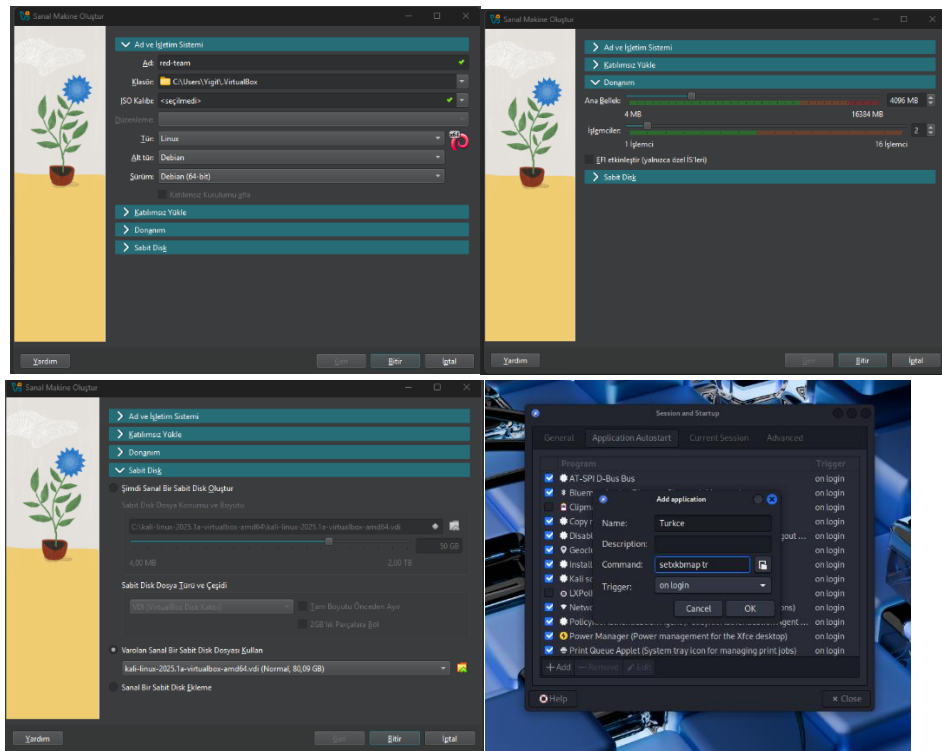
- **Windows Server 2019 – Served as the monitored target with Wazuh Agent installed.**



Windows Server 2019 Installation Summary:

- ✓ Selected language, time, and keyboard settings.
- ✓ Clicked "Install Now."
- ✓ Accepted the license agreement.
- ✓ Chose "Custom: Install Windows only."
- ✓ Selected the disk (unallocated space).
- ✓ Started copying Windows files.
- ✓ Installation process continued and Windows setup began.

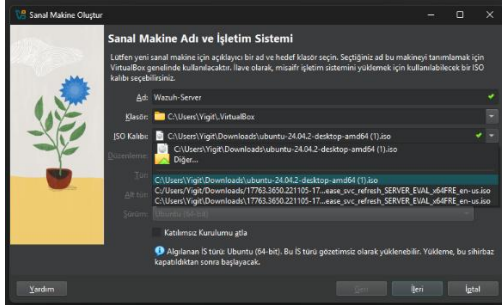
- **Kali Linux – Used to simulate brute-force attacks.**



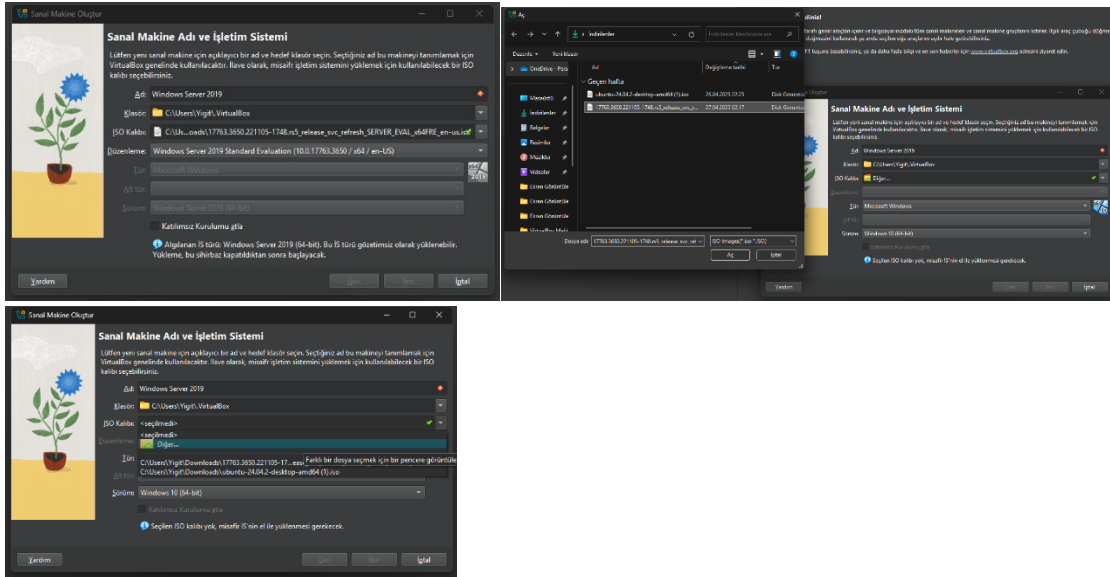
2.2 ISO File Selection

Official ISO images were used for clean and realistic installations:

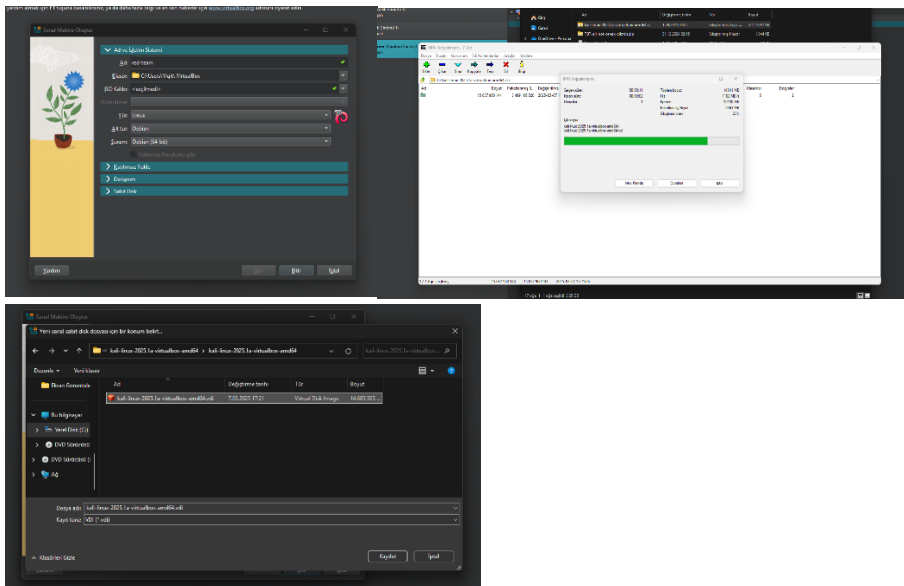
- **Ubuntu Server: Minimal desktop version**



- **Windows Server 2019: Evaluation ISO**



- **Kali Linux: Latest VirtualBox prebuilt image**

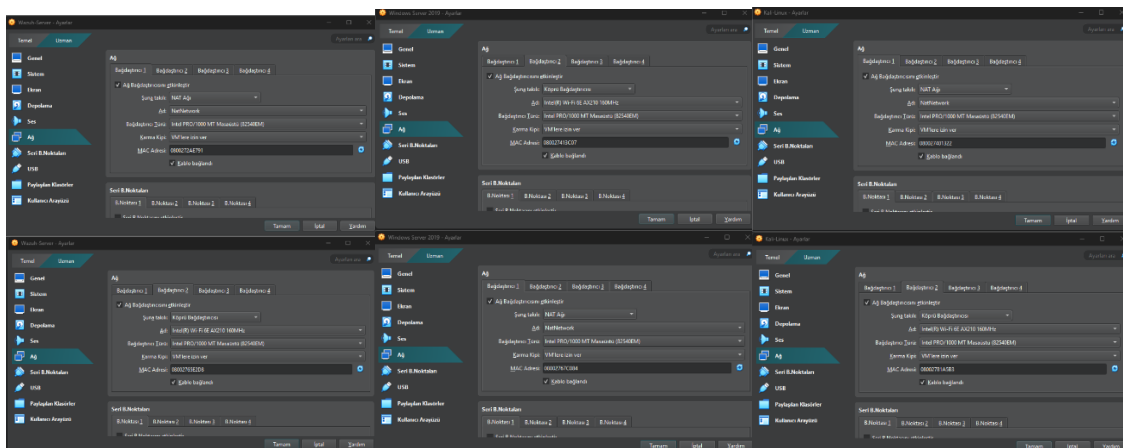


All ISO files were mounted through VirtualBox, and operating systems were installed step-by-step. Screenshots show installation sequences for each VM.

2.3 Network Configuration

Each VM was configured with two adapters:

- **Adapter 1 (NAT):** For internet access.
- **Adapter 2 (Host-Only):** For internal lab communication.



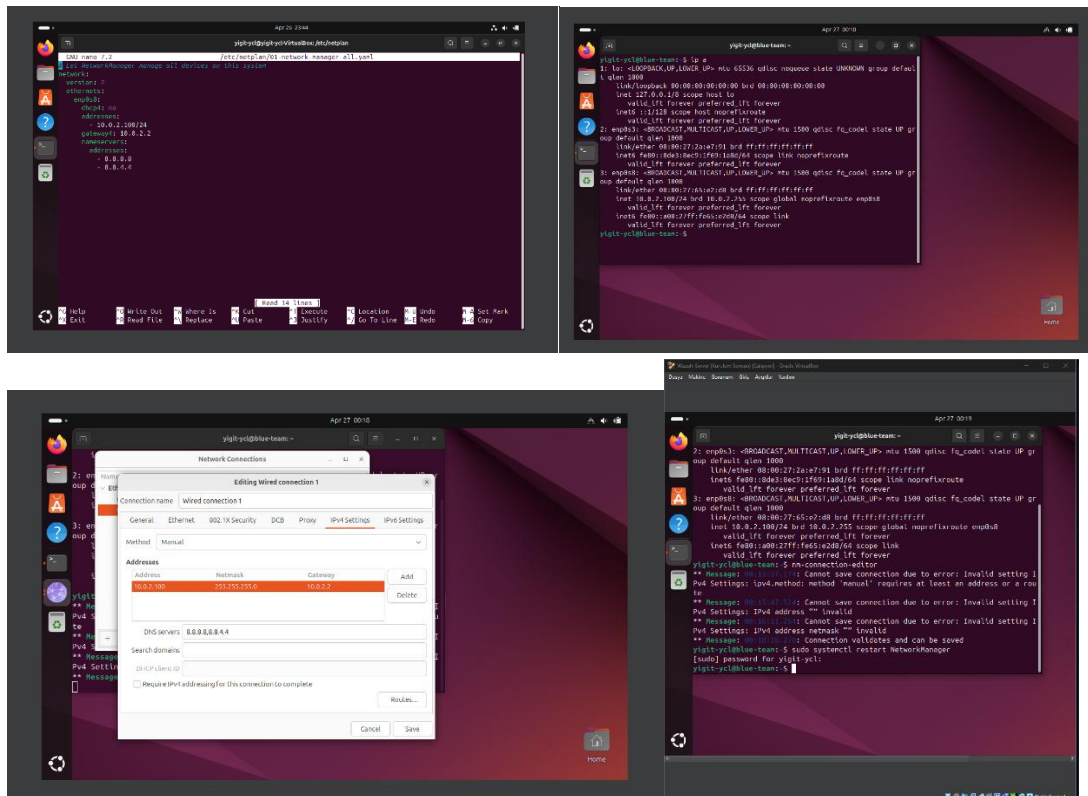
Screenshots illustrate how these adapters were configured for Ubuntu, Windows, and Kali Linux respectively.

2.4 Static IP Assignment

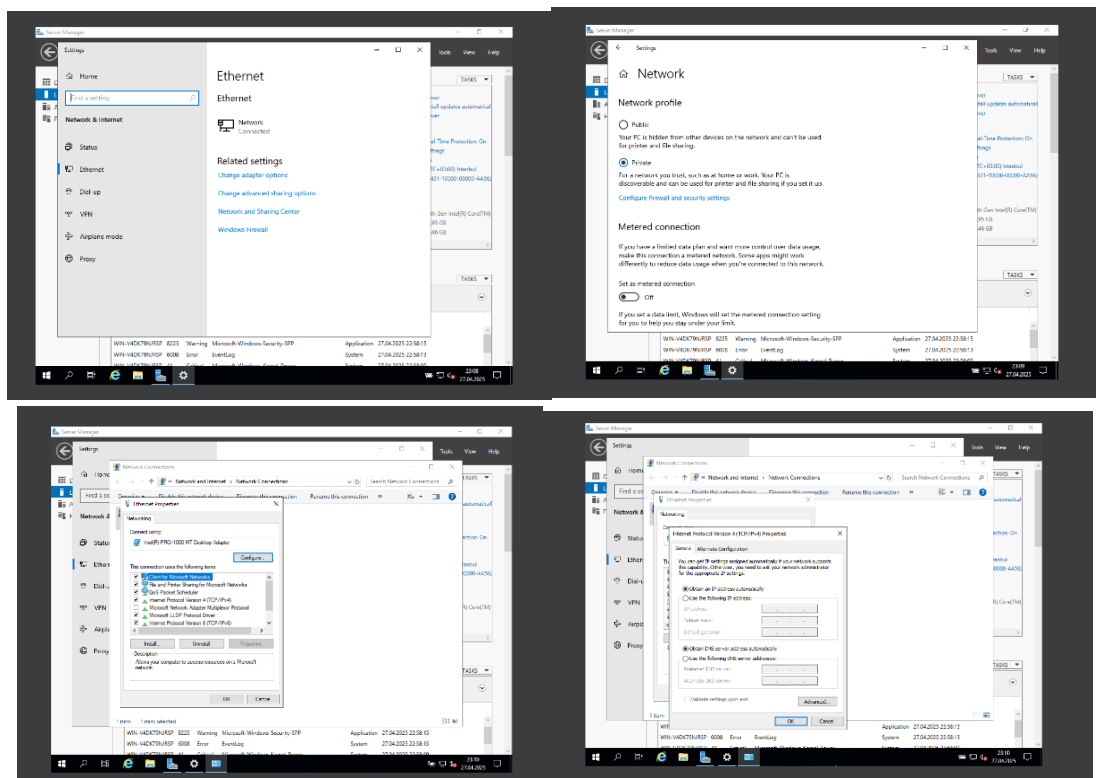
To ensure stability and avoid IP conflicts during simulation, each VM was manually assigned a static IP address:

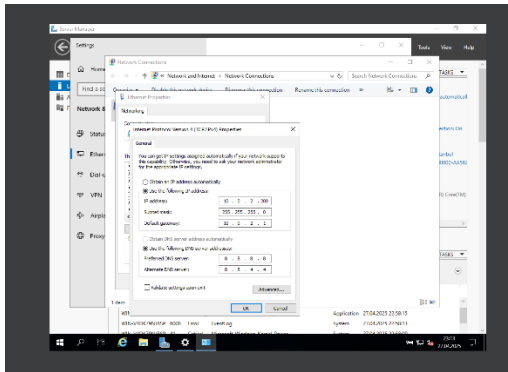
- Ubuntu Server (Wazuh Manager): 10.0.2.100
- Windows Server 2019: 10.0.2.200
- Kali Linux: 10.0.2.101

Ubuntu: IP was configured using the nm-connection-editor GUI tool.

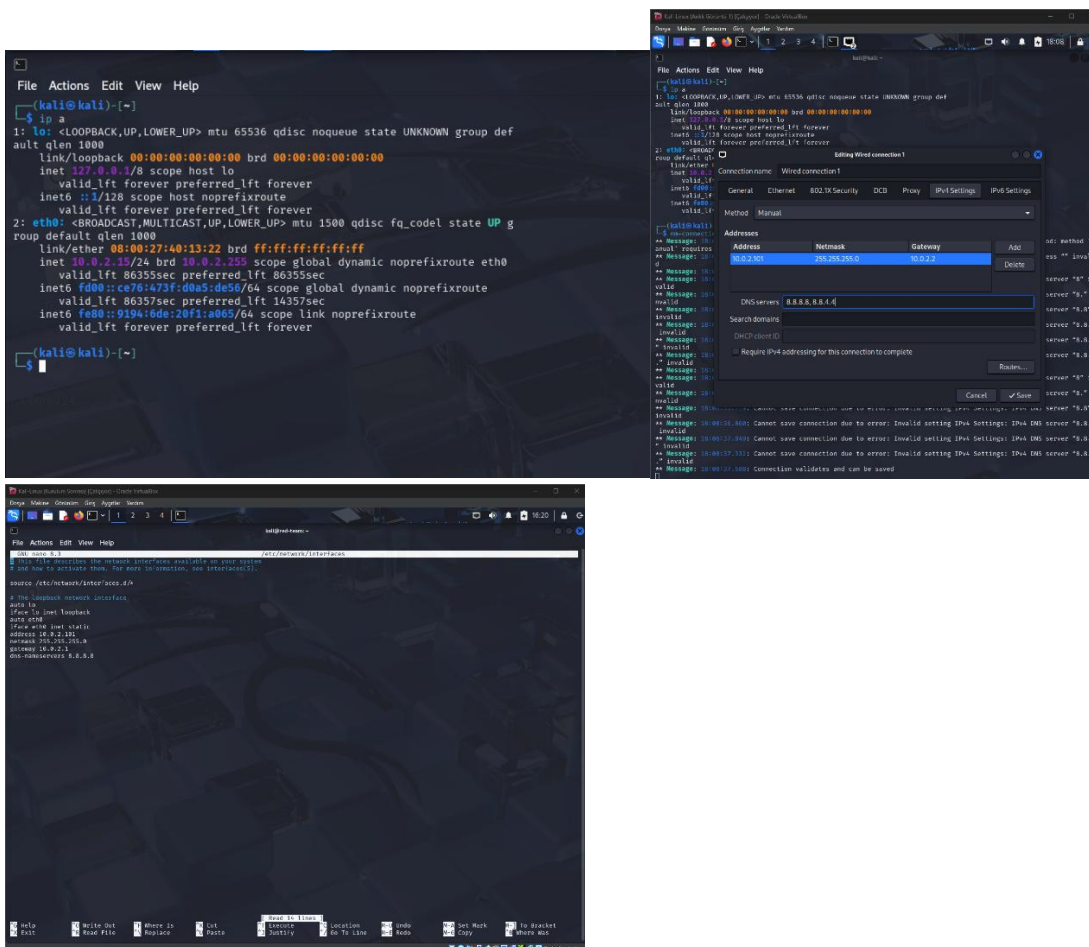


Windows: IP assigned through the Control Panel > Network settings.





Kali Linux: IP was set via /etc/network/interfaces or GUI interface.



Each setting step is supported with screenshots.

2.5 Ping Test Verification

Ping tests were successfully conducted between:

- Ubuntu ↔ Windows
- Ubuntu ↔ Kali
- Kali ↔ Windows

The image displays three screenshots of network connectivity tests. The top-left screenshot shows a Windows PowerShell window where a ping to 10.0.2.100 is successful, with 4 packets received and 0% loss. The top-right screenshot shows a terminal window on a Kali Linux virtual machine (yigit-ycl@blue-team) pinging 10.0.2.200, also successful with 4 packets received and 0% loss. The bottom-left screenshot shows a Kali Linux terminal (kali@red-team) pinging 10.0.2.101, successful with 4 packets received and 0% loss. The bottom-right screenshot shows a Kali Linux terminal (kali@red-team) pinging 10.0.2.200, successful with 4 packets received and 0% loss.

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> ping 10.0.2.100

Pinging 10.0.2.100 with 32 bytes of data:
Reply from 10.0.2.100: bytes=32 time=1ms TTL=64
Reply from 10.0.2.100: bytes=32 time=1ms TTL=64
Reply from 10.0.2.100: bytes=32 time=1ms TTL=64
Reply from 10.0.2.100: bytes=32 time=1ms TTL=64

Ping statistics for 10.0.2.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
PS C:\Users\Administrator>
```

```
yigit-ycl@blue-team:~$ ping -c 4 10.0.2.200
PING 10.0.2.200 (10.0.2.200) 56(84) bytes of data.
64 bytes from 10.0.2.200: icmp_seq=1 ttl=128 time=3.14 ms
64 bytes from 10.0.2.200: icmp_seq=2 ttl=128 time=1.10 ms
64 bytes from 10.0.2.200: icmp_seq=3 ttl=128 time=0.683 ms
64 bytes from 10.0.2.200: icmp_seq=4 ttl=128 time=0.477 ms

--- 10.0.2.200 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3024ms
rtt min/avg/max/mdev = 0.477/1.350/3.141/1.058 ms
yigit-ycl@blue-team:~$
```

```
kali@red-team:~$ ping -c 4 10.0.2.101
PING 10.0.2.101 (10.0.2.101) 56(84) bytes of data.
64 bytes from 10.0.2.101: icmp_seq=1 ttl=64 time=0.660 ms
64 bytes from 10.0.2.101: icmp_seq=2 ttl=64 time=0.644 ms
64 bytes from 10.0.2.101: icmp_seq=3 ttl=64 time=0.636 ms
64 bytes from 10.0.2.101: icmp_seq=4 ttl=64 time=0.737 ms

--- 10.0.2.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3055ms
rtt min/avg/max/mdev = 0.636/0.666/0.737/0.031 ms
kali@red-team:~$
```

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> ping 10.0.2.101

Pinging 10.0.2.101 with 32 bytes of data:
Reply from 10.0.2.101: bytes=32 time=1ms TTL=64
Reply from 10.0.2.101: bytes=32 time=1ms TTL=64
Reply from 10.0.2.101: bytes=32 time=1ms TTL=64
Reply from 10.0.2.101: bytes=32 time=1ms TTL=64

Ping statistics for 10.0.2.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
PS C:\Users\Administrator>
```

```
kali@red-team:~$ ping -c 4 10.0.2.200
PING 10.0.2.200 (10.0.2.200) 56(84) bytes of data.
64 bytes from 10.0.2.200: icmp_seq=1 ttl=128 time=0.676 ms
64 bytes from 10.0.2.200: icmp_seq=2 ttl=128 time=0.584 ms
64 bytes from 10.0.2.200: icmp_seq=3 ttl=128 time=0.713 ms
64 bytes from 10.0.2.200: icmp_seq=4 ttl=128 time=0.900 ms

--- 10.0.2.200 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3126ms
rtt min/avg/max/mdev = 0.584/0.693/0.900/0.077 ms
kali@red-team:~$
```

The ICMP responses validated the network topology and confirmed internal connectivity.

3. Wazuh Manager Setup

Wazuh Manager and Dashboard were installed on Ubuntu. The steps were as follows:

1. **System Update**
2. **Download Wazuh Installation Script**
3. **Run Installation Script (All-in-One)**
4. **Verify Services**

```
yigit-ycl@blue-team:~$ sudo apt update && sudo apt upgrade -y
[sudo] password for yigit-ycl:
Hit:1 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://packages.wazuh.com/4.x/apt stable InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
13 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following upgrades have been deferred due to phasing:
  apparmor distro-info-data fwupd gir1.2-mutter-14 gnome-control-center
  gnome-control-center-data gnome-control-center-faces libapparmor1 libfwupd2
  libmutter-14-0 mutter-common mutter-common-bin ubuntu-drivers-common
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
```

```
yigit-ycl@blue-team:~$ curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | su
do gpg --dearmor -o /usr/share/keyrings/wazuh-archive-keyring.gpg
File '/usr/share/keyrings/wazuh-archive-keyring.gpg' exists. Overwrite? (y/N) y
yigit-ycl@blue-team:~$
```

```
yigit-ycl@blue-team:~$ echo "deb [signed-by=/usr/share/keyrings/wazuh-archive-ke
ying.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | sudo tee /etc/apt/so
urces.list.d/wazuh.list
deb [signed-by=/usr/share/keyrings/wazuh-archive-keyring.gpg] https://packages.w
azuh.com/4.x/apt/ stable main
yigit-ycl@blue-team:~$
```

```
yigit-ycl@blue-team:~$ sudo systemctl status wazuh-manager.service
wazuh-manager.service - Wazuh manager
Loaded: loaded (/usr/lib/systemd/system/wazuh-manager.service; disabled; vendor p
referred)
Active: active (running) since Mon 2025-04-28 19:44:15 +03; 35s ago
Process: 73690 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start ($?)
Tasks: 149 (limit: 4611)
Memory: 1.4G (peak: 1.5G)
CPU: 55.713s
CGroup: /system.slice/wazuh-manager.service
┌─73152 /var/ossec/framework/python/bin/python3 /var/ossec/apl/sc
├─73192 /var/ossec/bin/wazuh-authd
├─73207 /var/ossec/bin/wazuh-db
├─73217 /var/ossec/bin/wazuh-execd
├─73224 /var/ossec/framework/python/bin/python3 /var/ossec/apl/sc
├─73225 /var/ossec/framework/python/bin/python3 /var/ossec/apl/sc
├─73229 /var/ossec/framework/python/bin/python3 /var/ossec/apl/sc
├─73233 /var/ossec/framework/python/bin/python3 /var/ossec/apl/sc
├─73239 /var/ossec/bin/wazuh-analysisd
├─73294 /var/ossec/bin/wazuh-syscheckd
├─73310 /var/ossec/bin/wazuh-remoted
├─73348 /var/ossec/bin/wazuh-logcollector
├─73364 /var/ossec/bin/wazuh-monitord
└─73377 /var/ossec/bin/wazuh-modulesd
```

```
yigit-ycl@blue-team:~$ sudo /var/ossec/bin/manage_agents

*****
* Wazuh v4.11.2 Agent manager. *
* The following options are available: *
*****

(A)dd an agent (A).
(E)xtract key for an agent (E).
(L)ist already added agents (L).
(R)emove an agent (R).
(Q)uit.
Choose your action: A,E,L,R or Q: Q
```

```
(E)xtract key for an agent (E).
(L)ist already added agents (L).
(R)emove an agent (R).
(Q)uit.
Choose your action: A,E,L,R or Q: A

- Adding a new agent (use 'q' to return to the main menu).
Please provide the following:
  * A name for the new agent: Windows-Server
  * The IP Address of the new agent: 10.0.2.200
Confirm adding it?(y/n): y
Agent added with ID 001.

*****
* Wazuh v4.11.2 Agent manager. *
* The following options are available: *
*****

(A)dd an agent (A).
(E)xtract key for an agent (E).
(L)ist already added agents (L).
(R)emove an agent (R).
(Q)uit.
Choose your action: A,E,L,R or Q: E

Available agents:
ID: 001, Name: Windows-Server, IP: 10.0.2.200
Provide the ID of the agent to extract the key (or 'q' to quit): 001

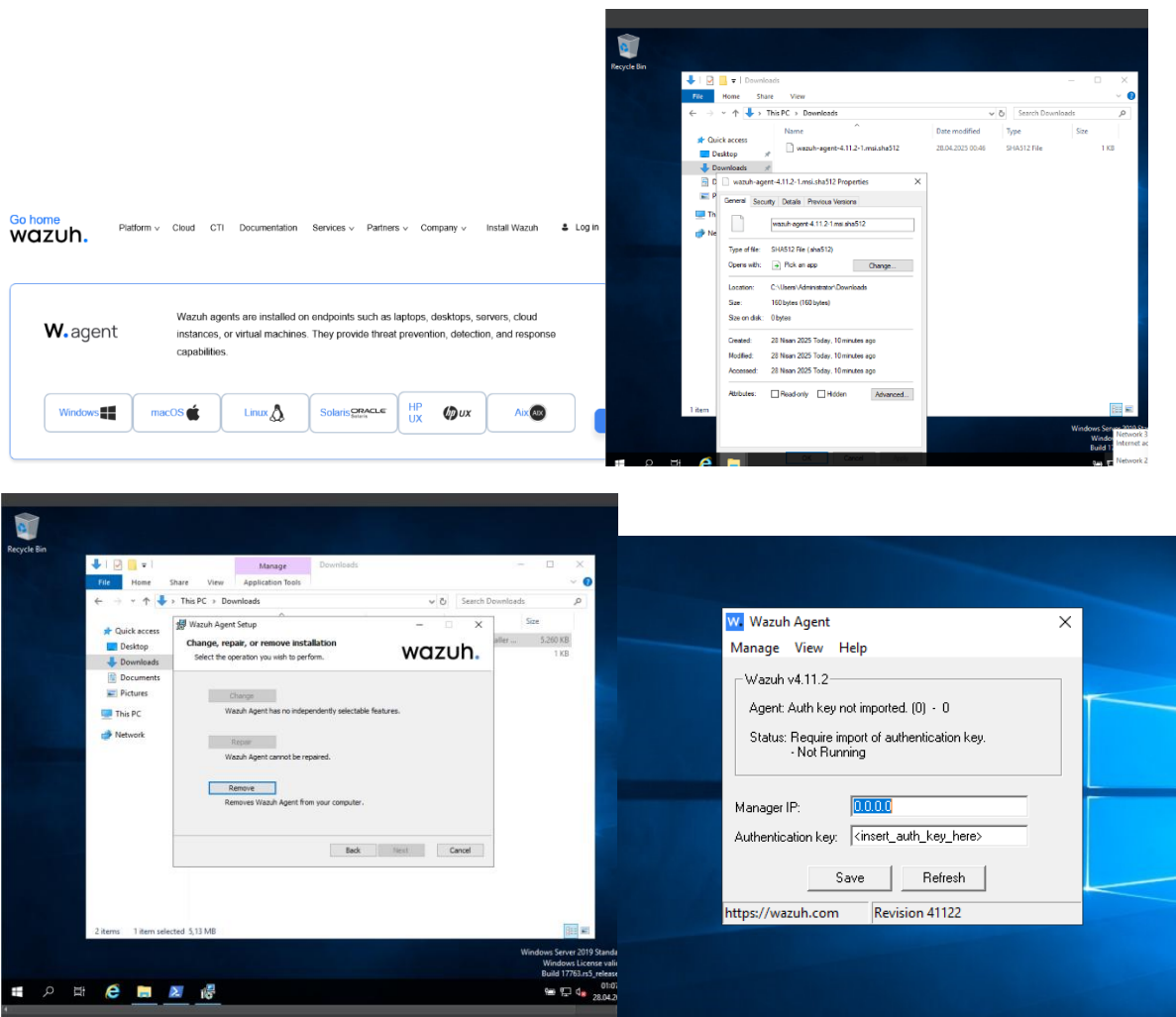
Agent key information for '001' is:
MDAxIFdpbnRvdjM0tUydWVYDEwLJAUIM4YMDAGYWMIMWU4OTgzZDRlMnM2FNGNjMGQ5NWYxMWUS
ZjBhODY2MzQwTg0ZjAzMTESYjdjZjBkYzVzMGMhNmZISzQ==

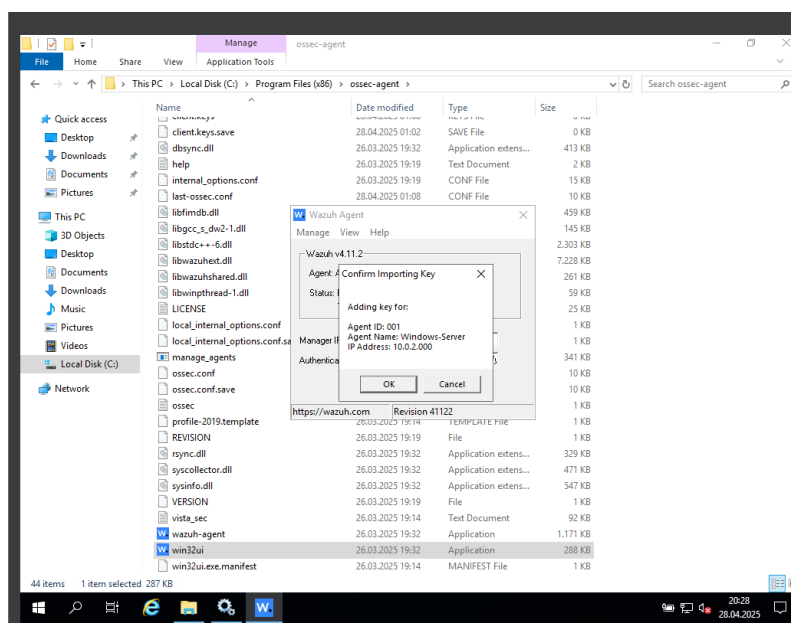
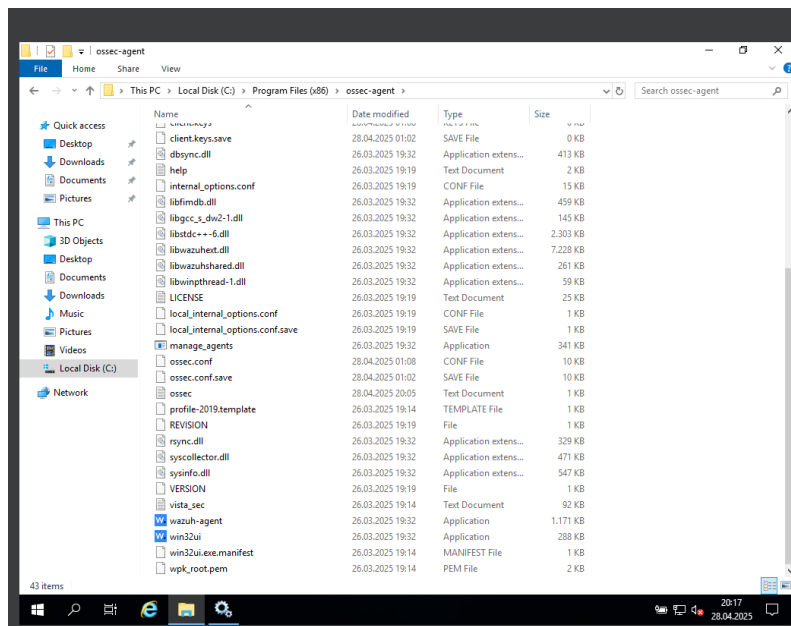
** Press ENTER to return to the main menu.
```

4. Wazuh Agent Setup (Windows)

On the Windows Server:

1. The Wazuh Agent .msi installer was downloaded and installed.
2. Configuration file ossec.conf was edited to point to the Wazuh Manager IP.
3. On Ubuntu, the agent key was generated:
4. This key was copied to the Windows Agent interface.
5. The Wazuh Agent service was started and successfully connected to the Manager.





5. Brute-Force Attack Simulation

5.1 Attack Objective

The purpose of this attack simulation was to test the ability of the Wazuh SIEM environment to detect unauthorized access attempts through brute-force attacks on the Windows Server 2019 machine's Remote Desktop Protocol (RDP) service (port 3389).

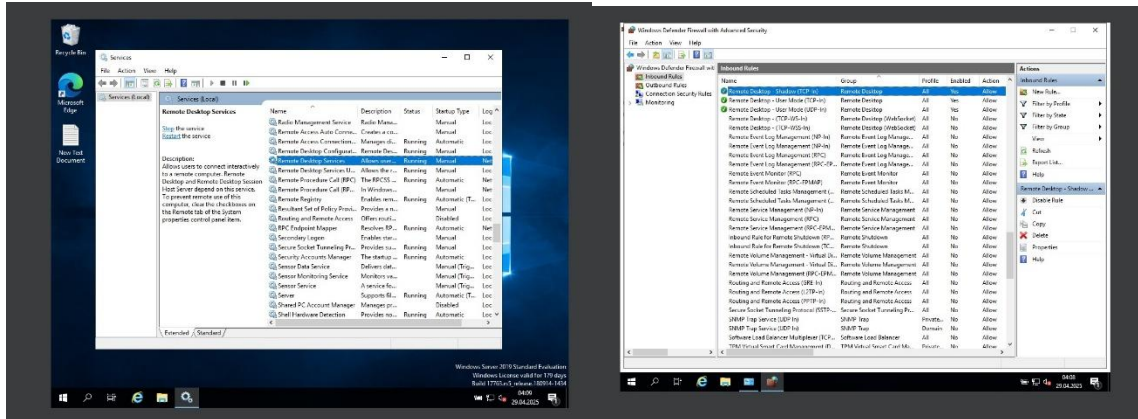
The attacker machine used in this simulation was Kali Linux.

5.2 Preparation Phase

5.2.1 RDP Service Activation

Before launching the attack, it was verified that the RDP service on the Windows Server was enabled and listening on port 3389.

This was confirmed using tools like nmap from the Kali Linux machine, ensuring the target port was open.



5.2.2 Wordlist Preparation

A custom password list named p.txt was created on the Kali Linux machine. This list contained commonly used weak passwords to simulate a realistic brute-force attempt.

File path:

/home/kali/Desktop/p.txt

```
(kali@red-team)-[~]
$ nmap -p 3389 10.0.2.200
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-28 21:10 EDT
Nmap scan report for 10.0.2.200
Host is up (0.00075s latency).

PORT      STATE SERVICE
3389/tcp  open  ms-wbt-server
MAC Address: 08:00:27:67:C0:84 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds

(kali@red-team)-[~]
$
```

5.3 Attack Execution

5.3.1 Hydra Installation

Hydra, the password-cracking tool, was installed on Kali Linux if it was not already present:

```
sudo apt update
sudo apt install hydra -y
```

Hydra is a powerful tool capable of performing fast brute-force attacks against multiple protocols, including RDP.



5.3.2 Launching the Brute-Force Attack

The following Hydra command was executed to initiate the brute-force attack:

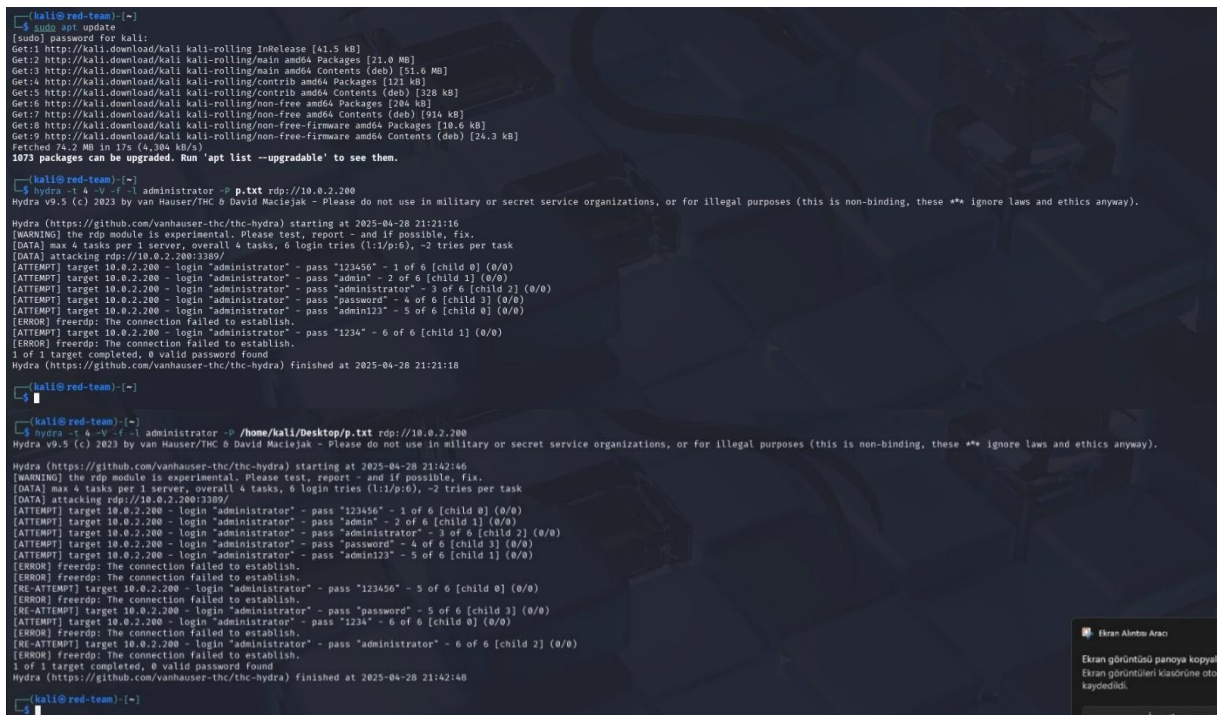
```
hydra -t 4 -V -f -l administrator -P /home/kali/Desktop/p.txt rdp://10.0.2.200
```

- **-t 4**: Number of parallel connections (threads).
- **-V**: Verbose output; display each login attempt.
- **-f**: Exit after the first found login/password pair.
- **-l administrator**: Username to test.
- **-P**: Path to the password file.
- **rdp://10.0.2.200**: Target IP address using RDP protocol.

Result:

Hydra attempted multiple password combinations listed in p.txt.

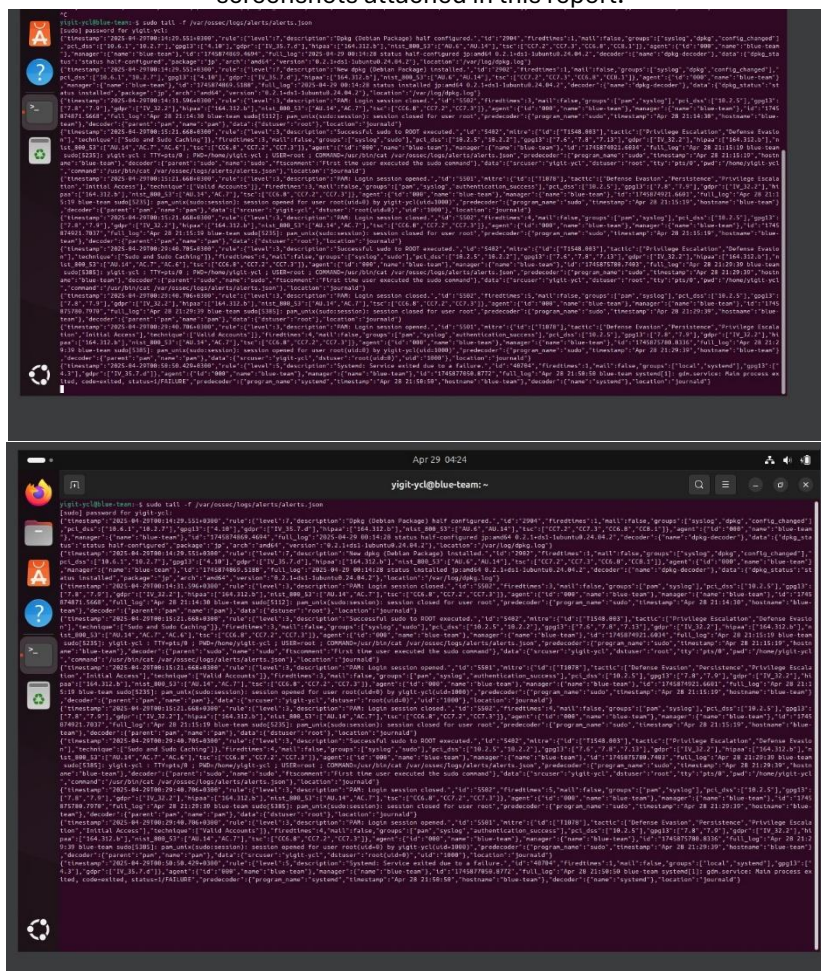
The login attempts were unsuccessful (no valid credentials found), but the authentication failures were logged on the Windows system.

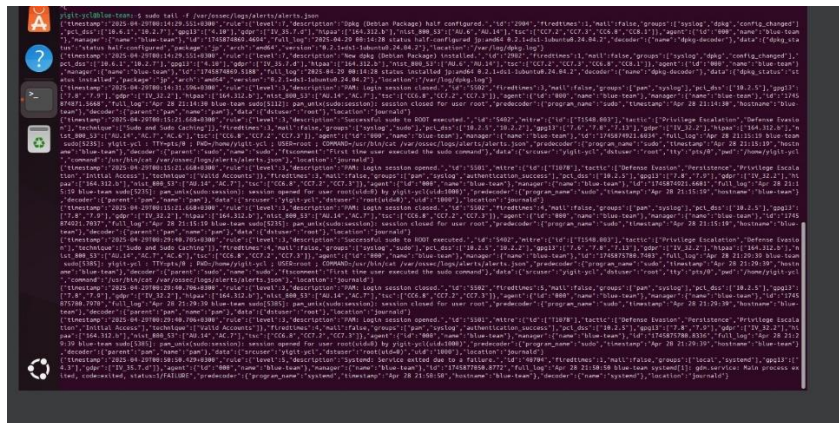


5.4 Observations and Results

- Although Hydra did not find a valid password, every login attempt triggered failed authentication logs (Event ID 4625) on the Windows Server.
- These logs were collected and analyzed by the Wazuh Manager.
- Wazuh generated real-time alerts based on brute-force detection rules.
- The detection was successfully mapped to MITRE ATT&CK Tactics and Techniques:
 - **T1110 – Brute Force**
 - **T1078 – Valid Accounts**
 - **T1548 – Abuse Elevation Control Mechanism**

The entire attack process, command execution, and system responses were documented with screenshots attached in this report.





Summary

The brute-force attack simulation confirmed that the SIEM environment (Wazuh) was capable of detecting multiple failed login attempts over RDP, validating the effectiveness of the monitoring and alerting setup.

8. General Evaluation and Summary

This project was designed to simulate scenarios that can be encountered in a real-world SOC (Security Operations Center) environment. The configurations and tests performed in a virtualized setting provided a comprehensive perspective on both system setup and security event monitoring.

Process Steps and Objectives:

1. **Virtual Environment Preparation:**
Virtual machines (Kali Linux, Ubuntu, and Windows Server 2019) were created using VirtualBox. ISO files were selected, hardware allocations were defined, and disk configurations were completed.
2. **Network and Communication Configuration:**
NAT and Host-Only adapters were configured. Static IP addresses were assigned, and communication between machines was verified using ping, ip a, and nmap.
3. **Wazuh SIEM Installation:**
Wazuh components (Manager, Indexer, Dashboard) were installed and configured on Ubuntu. Installation errors were diagnosed and resolved. Functionality was confirmed via real-time logs from the alerts.json file.
4. **Attack Scenario – Brute Force:**
A brute-force RDP attack was launched against the 10.0.2.200 Windows system using the hydra tool on Kali Linux. A custom password list was created, and multiple login attempts were made.
5. **Log and Alert Monitoring:**
The Wazuh dashboard was used to monitor real-time alerts related to the brute-force activity. Authentication failures and access attempts were logged in detail.

Conclusion and Evaluation:

This project provided both practical system configuration experience and an end-to-end simulation of a security event, offering a strong foundation in incident detection and response. Key stages such as live log tracking, alert generation, and attacker behavior analysis were successfully completed.

Key Takeaways:

- Proficiency in setting up and managing a SOC environment
- Hands-on experience with log collection and alert analysis via Wazuh
- Understanding of brute-force attacks and detection techniques
- Methodical approach to network communication, validation, and troubleshooting

This work establishes a solid baseline for Blue Team operations and can be further enhanced with advanced Wazuh modules such as custom rule writing, file integrity monitoring, and threat intelligence integration.