Operation RedTrace – KYPO Edition

# Introduction

Operation RedTrace is an immersive cybersecurity training scenario that simulates a small enterprise under attack by a sophisticated adversary. Participants assume both Red Team and Blue Team roles. The Red Team uses MITRE Caldera for automated adversary emulation, while the Blue Team uses Wazuh (a SIEM/XDR platform) for threat detection and response. The storyline involves an APT group codenamed “RedTrace” targeting the organisation. Trainees will deploy attacks via Caldera and detect/ respond via Wazuh.

This guide provides a complete blueprint of the virtual environment and step-by-step deployment instructions for recreating the scenario on the KYPO Cyber Range Platform (CRP). All components are defined in KYPO’s YAML-based topology format for automated provisioning. By the end of the setup, you will have a realistic lab environment with multiple networked machines, ready to execute a scripted cyber attack and test the defensive monitoring in place.

Scope: This is a deployment guide. It covers building the environment from scratch (including installing and configuring Caldera, Wazuh SIEM, ELK Stack, Filebeat, Active Directory domain, and Wazuh agents on Linux/Windows hosts). It does *not* cover the detailed execution of the adversary simulation or analysis of results, though basic verification steps are included. All code snippets (Ansible playbooks, configuration files, etc.) are provided in full and have been tested on the specified OS versions. The instructions assume you have a working KYPO CRP and sufficient permissions to deploy custom scenarios.

# Virtual Environment Topology

Diagrama

El contenido generado por IA puede ser incorrecto.

*Figure 1: Operation RedTrace scenario topology.* The training network mimics a small corporate LAN with multiple hosts and a central logging/SIEM server. All machines reside in a single network (e.g. 172.25.0.0/24) so that the Red Team server and SIEM can communicate with all endpoints. Key nodes in this scenario include:

* Audit-Control-01 – Kali Linux 2025.1 (Red Team attack platform). This VM hosts the MITRE Caldera C2 server and comes with common offensive tools (Kali distribution). It will be used to launch simulated attacks via Caldera.

The sizing for this machine (without KYPO) is as follows:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

*Figure 2: The sizing for this machine.*

The operating system is as follows:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

*Figure 3: Kali Linux 2025.1c.*

* WebServer-01 – Ubuntu Server 22.04/24.04.2 (Linux web server, vulnerable). Hosts a legacy web application (simulated outdated CMS) to serve as an initial intrusion point. A Caldera Linux agent (“Sandcat”) will be run here to emulate compromise, and the Wazuh agent will be installed for monitoring.

The sizing for this machine (without KYPO) is as follows:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

*Figure 4: The sizing for this machine.*

The operating system is as follows:

Texto

El contenido generado por IA puede ser incorrecto.

*Figure 5: Ubuntu Server 24.04.2.*

* FileServer-01 – Windows Server 2019 (Windows file server, domain member). Acts as an internal file share server (SMB shares) and secondary target for lateral movement. This host will be joined to the Active Directory domain. We install the Wazuh agent and Microsoft Sysmon here to collect detailed telemetry (process creation, file modifications) for detection. A Caldera Windows agent will be executed here to simulate an adversary-controlled foothold.
* DomainController – Windows Server 2019 (Active Directory domain controller). Provides Active Directory services for the domain (e.g. domain redtrace.local). It manages domain accounts and policies, enabling simulation of domain-specific techniques (credential theft, lateral movement using domain creds, etc.). We promote this server to be the AD Domain Controller. The Wazuh agent (and Sysmon) will be installed here as well to monitor security events on the DC. *Note:* Including a Domain Controller adds realism (allowing techniques like credential dumping or detection of DC-specific logs), though the primary attack path in this scenario can be executed without heavy interaction with the DC for simplicity.
* SIEM-Node – Ubuntu Server 22.04/24.04.2 (Central SIEM server). This is the monitoring node running Wazuh Manager and the ELK stack (Elasticsearch, Logstash, Kibana). It aggregates logs/alerts from all other systems. The Wazuh web console (as a Kibana plugin) will be used by the Blue Team to analyze alerts. This server has a static IP known to all agents (for example, 10.10.0.50) so agents can connect to it. Elastic Stack components (Elasticsearch, Kibana, Filebeat, Logstash) are installed here to store and visualize the collected data.

The sizing for this machine (without KYPO) is as follows:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

*Figure 6: The sizing for this machine.*

The operating system is as follows:

Texto

El contenido generado por IA puede ser incorrecto.

*Figure 7: Ubuntu Server 24.04.2.*

All hosts are connected to the same LAN ( in the KYPO topology) so they can freely

enterprise\_net

communicate. We assign static IPs as noted above for consistency (see figure). Basic network services like DNS can be provided by the DomainController (since AD DS installs DNS for the domain), or simple entries can be used in a lab context if needed. In our case, the Domain Controller will act

/etc/hosts

as the DNS server for *redtrace.local*.

Machines created on grits3 in Proxmox VE:

Texto, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

*Figure 8: Machines created on grits3 in Proxmox VE.*

Topology Definition: The KYPO below describes the network and VMs. This can be

topology.yaml

used in KYPO CRP to define the scenario environment. It specifies the base images, roles, network settings, and any initial software to be present on each VM.

Filename: (KYPO Scenario Topology Definition)

topology:

name: Operation\_RedTrace version: 1.0 description: |

Virtual topology for the "Operation RedTrace" scenario.

Emulates a corporate environment with Windows and Linux machines, an Active Directory domain, and a central SIEM/logging node.

nodes:

* name: Audit-Control-01

os: kali-linux-2025.1 # Kali Linux attack platform role: Red Team Operator

ip: 172.25.0.10

resources: cpu: 2

ram: 4096 software:

* + mitre-caldera
  + wireshark

topology.yaml

* openssh-server
* python3-pip notes: |

Attack VM (Kali) for Red Team. Hosts the Caldera server and offensive tools. Will run APT emulation via Caldera "Sandcat" agents on targets.

* + name: WebServer-01

os: ubuntu-22.04 # Ubuntu 22.04 LTS server role: Vulnerable Web Server

ip: 172.25.0.20

resources: cpu: 2

ram: 2048 software:

* apache2
* php
* mysql-server
* openssh-server
* legacy-cms (simulated) notes: |

Simulated vulnerable web server (e.g. running an outdated Joomla CMS). This will be an initial foothold for the attacker (Linux target).

* + name: FileServer-01

os: windows-server-2019 # Windows Server 2019 role: File Share (Domain Member)

ip: 172.25.0.30

resources: cpu: 2

ram: 4096 software:

* SMB (file sharing)
* RDP
* PowerShell Remoting notes: |

Windows file server with shared folders (SMB) for lateral movement simulation. Will be joined to the AD domain.

* + name: DomainController

os: windows-server-2019 # Windows Server 2019 role: Active Directory Domain Controller

ip: 172.25.0.40

resources: cpu: 2

ram: 4096 software:

* Active Directory Domain Services
* DNS
* DHCP

notes: |

Active Directory Domain Controller for the enterprise (e.g., domain "redtrace.local"). Manages domain identities and policies.

* + name: SIEM-Node

os: ubuntu-22.04

role: Monitoring & SIEM ip: 172.25.0.50

resources: cpu: 4

ram: 8192 software:

* wazuh-manager
* filebeat
* logstash
* elasticsearch
* kibana notes: |

# Ubuntu 22.04 LTS server

Central SIEM node running Wazuh Manager and the Elastic Stack (Elasticsearch, Logstash, Kibana) to collect and analyze logs/alerts from all hosts.

networks:

- name: enterprise\_net cidr: 172.25.0.0/24

accessible\_by\_user: true # The user can access nodes on this network

(e.g., via a jumpbox or direct if allowed)

Note: The entries above indicate key services or packages for each host.

software

Actual installation of these services is accomplished through Ansible provisioning (as

described in the next section). The KYPO platform uses Ansible playbooks associated

ansible\_groups

with groups ( ) to automatically configure each VM after creation. We

will provide the full Ansible playbooks for each role in the following steps. Ensure the KYPO control node (or wherever you run Ansible) has network access and credentials for SSH (Linux VMs) and WinRM (Windows VMs) to configure the machines. In our scenario, we assume the Windows base images have WinRM enabled and a known Administrator password, or that you set up an Ansible inventory with the credentials. Some tasks (like domain promotion) will reboot the target VM – the automation accounts for this with appropriate handlers.

# Step-by-Step Deployment

Now we walk through provisioning each component of the scenario. The deployment can be automated via Ansible as shown, or done manually following the same steps. It is crucial to follow the order of steps, because certain configurations depend on others (for example, the domain controller must be set up before a client can join the domain, and Wazuh manager must be running before agents attempt to register).

## Red Team Attack Platform – Install Caldera on Kali (Audit-Control-01)

On the Kali Linux machine, we install the MITRE Caldera server and its dependencies. Kali 2025.1 already includes Python 3 and many tools, but we ensure Python3, pip, and Git are present. We then clone the Caldera repository from GitHub, install the Python requirements, and set up Caldera to run as a service. Finally, we open or confirm access to Caldera’s web interface port (default 8888/tcp). The

--insecure

service will run with certificates).

mode to allow HTTP (for a lab environment, this avoids needing

Filename: (Playbook for Kali/Caldera setup)

install\_caldera\_server.yml

- hosts: Audit-Control-01

become: yes

vars:

caldera\_dir: /opt/caldera

caldera\_repo: "https://github.com/mitre/caldera.git"

caldera\_venv: "{{ caldera\_dir }}/venv"

caldera\_port: 8888

vite\_url: "http://{{ ansible\_default\_ipv4.address }}:{{ caldera\_port }}"

tasks:

- name: Install required packages

apt:

name:

- python3

- python3-pip

- python3-venv

- git

- nodejs

- npm

- python3-lxml

state: present

update\_cache: yes

- name: Clone Caldera repository

git:

repo: "{{ caldera\_repo }}"

dest: "{{ caldera\_dir }}"

version: master

recursive: yes

force: yes

- name: Remove lxml from requirements.txt to avoid build failures

lineinfile:

path: "{{ caldera\_dir }}/requirements.txt"

regexp: '^lxml.\*'

state: absent

- name: Create Python virtual environment

command: python3 -m venv venv

args:

chdir: "{{ caldera\_dir }}"

creates: "{{ caldera\_venv }}/bin/activate"

- name: Install Caldera requirements in virtualenv

pip:

requirements: requirements.txt

virtualenv: "{{ caldera\_venv }}"

args:

chdir: "{{ caldera\_dir }}"

- name: Set VITE\_CALDERA\_URL for external access

copy:

dest: "{{ caldera\_dir }}/plugins/magma/.env"

content: |

VITE\_CALDERA\_URL={{ vite\_url }}

- name: Build Caldera magma UI

shell: |

npm install && npm run build

args:

chdir: "{{ caldera\_dir }}/plugins/magma"

- name: Activate magma plugin as HTTP interface

copy:

dest: "{{ caldera\_dir }}/conf/local.yml"

content: |

app.contact.http:

- plugin: magma

- name: Create systemd service for Caldera

copy:

dest: /etc/systemd/system/caldera.service

content: |

[Unit]

Description=MITRE Caldera Server

After=network.target

[Service]

Type=simple

User=root

WorkingDirectory={{ caldera\_dir }}

ExecStart={{ caldera\_venv }}/bin/python {{ caldera\_dir }}/server.py --insecure

Restart=on-failure

[Install]

WantedBy=multi-user.target

- name: Reload systemd

systemd:

daemon\_reload: yes

- name: Enable and start Caldera

service:

name: caldera

enabled: yes

state: restarted

- name: Allow Caldera port via UFW (if available)

ufw:

rule: allow

port: "{{ caldera\_port }}"

proto: tcp

when: ansible\_facts['os\_family'] == "Debian"

- name: Wait until Caldera responds

wait\_for:

host: "{{ ansible\_default\_ipv4.address }}"

port: "{{ caldera\_port }}"

delay: 5

timeout: 120

- name: Confirm HTTP response from Caldera

uri:

url: "http://{{ ansible\_default\_ipv4.address }}:{{ caldera\_port }}"

status\_code: 200

register: http\_check

retries: 10

delay: 5

until: http\_check.status == 200

**Una captura de pantalla de una computadora

El contenido generado por IA puede ser incorrecto.**After running the above, the Caldera server should be up and listening on port 8888 on the Kali machine. You can verify on the Kali VM: sudo systemctl status caldera (should show it active), or try accessing the Caldera web interface via a browser at http://10.10.0.10:8888 (adjust to the Kali IP). Caldera’s default credentials are usually admin:admin (you can change these in Caldera after first login). At this point, no agents are connected yet – we will deploy "Sandcat" agents to targets later to simulate the adversary footholds.

*Figure 9: install\_caldera\_server.yml execution.*

## SIEM Server – Install Wazuh Manager and Elastic Stack (SIEM-Node)

On the SIEM-Node (Ubuntu 22.04/24.04.2), we set up the Wazuh manager and the ELK stack components. This includes: - Wazuh Manager – core server that analyzes data from agents and raises alerts. - Filebeat – beats agent to forward Wazuh alerts to Logstash. - Logstash – to receive and process alerts from Filebeat, and index them into Elasticsearch. - Elasticsearch – datastore for logs and alerts. - Kibana – front-end web UI for searching data; we will install the Wazuh Kibana plugin to get a dedicated UI for Wazuh.

We will use the official Wazuh apt repository for Wazuh Manager (and its dependencies) and download the Elastic Stack components (version 7.17.10 in this scenario, which is compatible with Wazuh 4.x) as .deb packages to install. We also enable the Filebeat Wazuh module (which knows how to read Wazuh alert JSON) and configure Filebeat to send data to Logstash instead of directly to Elasticsearch. Finally, we install the Wazuh app plugin into Kibana.

Filename: (Playbook for Wazuh server and ELK setup)

install\_wazuh\_manager.yml

- hosts: SIEM-Node

become: yes

vars:

wazuh\_repo: https://packages.wazuh.com/4.x/apt/

wazuh\_gpg\_key: https://packages.wazuh.com/key/GPG-KEY-WAZUH

elasticsearch\_deb: https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.17.9-amd64.deb

kibana\_deb: https://artifacts.elastic.co/downloads/kibana/kibana-7.17.9-amd64.deb

logstash\_deb: https://artifacts.elastic.co/downloads/logstash/logstash-7.17.9-amd64.deb

filebeat\_deb: https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.17.9-amd64.deb

wazuh\_plugin\_url: https://packages.wazuh.com/4.x/ui/kibana/wazuh\_kibana-4.5.3\_7.17.9-1.zip

wazuh\_plugin\_zip: /tmp/wazuh\_kibana\_plugin.zip

tasks:

# Wazuh

- name: Add Wazuh GPG key

apt\_key:

url: "{{ wazuh\_gpg\_key }}"

state: present

- name: Add Wazuh APT repository

apt\_repository:

repo: "deb {{ wazuh\_repo }} stable main"

state: present

- name: Update APT cache

apt:

update\_cache: yes

- name: Install Wazuh Manager

apt:

name: wazuh-manager

state: present

# Elasticsearch

- name: Download Elasticsearch

get\_url:

url: "{{ elasticsearch\_deb }}"

dest: /tmp/elasticsearch.deb

mode: '0644'

- name: Install Elasticsearch

apt:

deb: /tmp/elasticsearch.deb

- name: Enable and start Elasticsearch

service:

name: elasticsearch

enabled: yes

state: started

# Kibana

- name: Download Kibana

get\_url:

url: "{{ kibana\_deb }}"

dest: /tmp/kibana.deb

mode: '0644'

- name: Install Kibana

apt:

deb: /tmp/kibana.deb

- name: Enable and start Kibana

service:

name: kibana

enabled: yes

state: started

- name: Download Wazuh Kibana plugin

get\_url:

url: "{{ wazuh\_plugin\_url }}"

dest: "{{ wazuh\_plugin\_zip }}"

mode: '0644'

- name: Install Wazuh Kibana plugin

shell: "/usr/share/kibana/bin/kibana-plugin install file://{{ wazuh\_plugin\_zip }}"

args:

creates: /usr/share/kibana/plugins/wazuh

notify: Restart Kibana

# Logstash

- name: Download Logstash

get\_url:

url: "{{ logstash\_deb }}"

dest: /tmp/logstash.deb

mode: '0644'

- name: Install Logstash

apt:

deb: /tmp/logstash.deb

- name: Configure Logstash pipeline

copy:

dest: /etc/logstash/conf.d/wazuh.conf

content: |

input {

beats {

port => 5044

codec => "json\_lines"

}

}

output {

elasticsearch {

hosts => ["localhost:9200"]

index => "wazuh-alerts-4.x-%{+YYYY.MM.dd}"

}

}

notify: Restart Logstash

- name: Enable and start Logstash

service:

name: logstash

enabled: yes

state: started

# Filebeat

- name: Download Filebeat

get\_url:

url: "{{ filebeat\_deb }}"

dest: /tmp/filebeat.deb

mode: '0644'

- name: Install Filebeat

apt:

deb: /tmp/filebeat.deb

- name: Disable Elasticsearch output in Filebeat config

replace:

path: /etc/filebeat/filebeat.yml

regexp: '^output.elasticsearch:'

replace: '#output.elasticsearch:'

- name: Enable Logstash output

replace:

path: /etc/filebeat/filebeat.yml

regexp: '^#output.logstash:'

replace: 'output.logstash:'

- name: Set Logstash host

lineinfile:

path: /etc/filebeat/filebeat.yml

insertafter: 'output.logstash:'

line: ' hosts: ["localhost:5044"]'

- name: Enable and start Filebeat

service:

name: filebeat

enabled: yes

state: started

handlers:

- name: Restart Kibana

service:

name: kibana

state: restarted

- name: Restart Logstash

service:

name: logstash

state: restarted

This playbook installs all components on the SIEM-Node. Once complete, verify the following on the SIEM-Node machine: - Elasticsearch is running and reachable on port 9200. For example, run

curl

http://localhost:9200

– it should return cluster information (name, status, version, etc.). - Kibana

is running on port 5601. You should be able to access http://172.25.0.50:5601 from a browser.

The Kibana UI should load (this may take a couple of minutes on first start). Log in (if prompted; default Kibana setup might not require auth by default for Elastic 7.x). Navigate to the Kibana sidebar – you should see a “Wazuh” app icon (under “SIEM” or in the main menu). This confirms the Wazuh Kibana plugin loaded successfully. - Logstash is running and listening on port 5044 (you can check

netstat -

tupan | grep 5044

or Logstash logs to confirm). - Filebeat is running and has the Wazuh module

enabled. Initially, it might not send anything until Wazuh manager generates an alert. We configured

/var/ossec/logs/alerts/alerts.json

Filebeat to read Wazuh alert logs (by default from ship them to Logstash.

The configuration above creates a Logstash pipeline ( Filebeat and indexes them into Elasticsearch indices

wazuh.conf ) that listens for JSON alerts

named wazuh-alerts-4.x-YYYY.MM.DD

) and

from

. We

rely on the Wazuh index template to be loaded for proper mappings (the Wazuh Kibana plugin usually takes care of this, but if not, one can load the template JSON via Elasticsearch API). With Kibana and the Wazuh app installed, these indices will be recognized and used by the Wazuh app interface for showing alerts and managing agents.

At this stage, the Wazuh Manager is installed but not yet receiving data from any agents (since we have

wazuh-manager

not installed any agents on endpoints). The Wazuh manager’s service ( ) should be

systemctl status wazuh-manager

running (check with ). It includes an internal Authd service

**Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.**listening on port 1515/TCP for agent registration and uses port 1514/UDP for receiving agent event data. We will leverage these in the next steps when deploying the agents.

*Figure 10: install\_wazuh\_manager.yml execution.*

## Windows Domain Controller – Active Directory Setup (DomainController)

Next, we configure the Domain Controller VM. We will promote the Windows Server 2019 instance to be an Active Directory Domain Services (AD DS) controller for a new forest/domain named

**redtrace.local**

. We also ensure the machine has the Wazuh agent and Sysmon (though in our sequence, we will install those via a later step alongside the file server – it's mentioned here for completeness).

The promotion to domain controller involves installing the AD DS server role and then running the PowerShell command to create a new domain. This will automatically

Install-ADDSForest

configure DNS for the domain and reboot the server upon completion. We’ll use Ansible to perform these steps unattended, specifying a Safe Mode Administrator password (for AD restore, etc.) – in this lab we use a sample password `"P@ssw0rd!" (you should choose a secure one in production).

Filename: promote.ps1 (script for AD Domain Controller setup)

Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools

Install-ADDSForest `

-DomainName "redtrace.local" `

-DomainNetbiosName "REDTRACE" `

-SafeModeAdministratorPassword (ConvertTo-SecureString "P@ssw0rd!" -AsPlainText -Force) `

-InstallDns `

-Force

Filename: domain.ps1 (script for AD Domain Controller setup)

Import-Module ActiveDirectory

Set-ADAccountPassword `

-Identity "Administrator" `

-NewPassword (ConvertTo-SecureString "P@ssw0rd!" -AsPlainText -Force) `

-Reset

This will trigger the AD installation. Note: The server will reboot automatically as part of the domain

Install-ADDSForest

promotion process (the command will schedule a reboot when finished). After

this step, give the DomainController a few minutes to come back online as the new domain controller for redtrace.local. The local Administrator account becomes the domain Administrator. For simplicity, we will use the built-in Administrator for domain admin tasks in this scenario. (In a real environment, you might create dedicated domain user accounts, but that’s optional for our simulation.)

Once the DomainController is up, it should be functioning as a DNS server for the domain as well. Ensure that other domain-joined hosts (like the FileServer we’ll configure next) point to this DC for DNS (the KYPO environment may handle this automatically if the network is closed; if not, we might need to

/etc/hosts

set the DNS IP on the FileServer to 172.25.0.40 or add an entry on the Linux host for

"redtrace.local"). For our purposes, it’s enough that the domain join and name resolution within the network work via the DC.

## Windows File Server – Join Domain and System Prep (FileServer-01)

The FileServer-01 is a Windows Server 2019 machine that we will join to the newly created redtrace.local domain. This will allow us to simulate lateral movement using domain credentials and group policy effects if needed. We also intend to install the Wazuh agent and Sysmon on this host (as well as on the DomainController) to monitor them.

Domain Join: We can automate domain joining using the Ansible module.

win\_domain\_membership

This requires the domain name and credentials of a domain admin. We will use the domain’s

Administrator account (username “Administrator”) and the password we set for the DomainController’s admin (which, following the above, is the password you set for the local Administrator when creating the VM, since after promotion that becomes the domain admin password; if you started with a known password for the Windows image’s Administrator, it remains the admin password unless changed). In our example, we assume Administrator and a password defined in an Ansible variable

domain\_admin\_pass

. Provide the actual Administrator password of your DomainController here.

The domain join will cause the FileServer to reboot upon success.

Filename: windows\_server\_setup.ps1 (Script for Windows server domain join)

# Variables

$domainName = "redtrace.local"

$dcAddress = "172.25.0.40"

Disable-NetAdapter -Name "Ethernet" -Confirm:$false

Set-DnsClientServerAddress -InterfaceAlias "Ethernet 2" -ServerAddresses ($dcAddress)

# Display DNS config

Write-Host "`nCurrent DNS server per interface:" -ForegroundColor Cyan

Get-DnsClientServerAddress | Format-Table InterfaceAlias, ServerAddresses

# Flush DNS cache

Write-Host "`nFlushing DNS resolver cache..." -ForegroundColor Cyan

ipconfig /flushdns | Out-Null

# Check connectivity to Domain Controller

Write-Host "`nTesting connectivity to DC (port 389)..." -ForegroundColor Cyan

$ping = Test-NetConnection -ComputerName redtrace.local -Port 389

if (-not $ping.TcpTestSucceeded) {

Write-Host "Unable to connect to redtrace.local on port 389. Aborting." -ForegroundColor Red

exit 1

}

# Prompt for domain credentials and join

Write-Host "`nPlease enter domain credentials to join '$domainName'" -ForegroundColor Cyan

try {

Add-Computer -DomainName $domainName -Credential (Get-Credential) -Restart

}

catch {

Write-Host "Failed to join the domain: $\_" -ForegroundColor Red

exit 1

}

Run this play after the DomainController is fully operational. The FileServer will reboot and come back as a member of redtrace.local domain. You can log in thereafter using domain credentials (e.g., REDTRACE\Administrator with the domain admin password) or the local credentials if needed (though local creds might be disabled or changed after join, domain admin will work).

## Deploy Wazuh Agents on Endpoints (Linux & Windows)

With the SIEM (Wazuh) ready and the network configured, the final step is to deploy Wazuh agents on the target machines (WebServer-01, FileServer-01, and DomainController). The agents will collect logs and send them to the Wazuh manager (SIEM-Node). We will also deploy Sysmon on the Windows hosts to augment the log data with detailed OS-level events (which Wazuh can capture from the Windows Event Log).

Linux Agent (WebServer-01): We install the Wazuh agent via the Wazuh apt repository (added earlier on the SIEM node; we can add it to the Ubuntu web server as well). After installation, we need to configure the agent to point to the manager’s IP and then register the agent with the manager. Wazuh

agent-auth

provides an agent enrollment mechanism. Here, we use the utility to register the agent

with the manager. By default, the Wazuh manager’s Authd service (on port 1515) will accept registration

requests without a pre-shared key (since Wazuh 4.x auto-enrollment is enabled by default). We simply

agent-auth -m <manager\_ip> -A <agent\_name>

run

playbook below automates this for the Linux host.

install\_wazuh\_agent\_linux.yml

Filename:

server)

on the client to get it enrolled. Our

(Playbook for installing Wazuh agent on Ubuntu web

- name: Install and configure Wazuh agent on Ubuntu

hosts: WebServer-01

become: yes

vars:

wazuh\_manager\_ip: "172.25.0.50" # SIEM node IP

agent\_name: "Ubuntu-WebServer" # Custom agent name

tasks:

- name: Add Wazuh GPG key

apt\_key:

url: "https://packages.wazuh.com/key/GPG-KEY-WAZUH"

state: present

- name: Add Wazuh APT repository

apt\_repository:

repo: "deb https://packages.wazuh.com/4.x/apt/ stable main"

filename: wazuh

state: present

- name: Update apt cache

apt:

update\_cache: yes

cache\_valid\_time: 3600

- name: Install Wazuh agent

apt:

name: wazuh-agent

state: present

- name: Configure manager IP in agent config

replace:

path: /var/ossec/etc/ossec.conf

regexp: '<address>.\*</address>'

replace: "<address>{{ wazuh\_manager\_ip }}</address>"

- name: Set custom agent name (optional)

replace:

path: /var/ossec/etc/ossec.conf

regexp: '<agent\_name>.\*</agent\_name>'

replace: "<agent\_name>{{ agent\_name }}</agent\_name>"

- name: Register agent with Wazuh manager

shell: >

/var/ossec/bin/agent-auth -m {{ wazuh\_manager\_ip }} -p 1515 -A "{{ agent\_name }}"

register: auth\_out

failed\_when: auth\_out.rc not in [0,1]

- name: Enable and start wazuh-agent service

service:

name: wazuh-agent

state: started

enabled: yes

This will install and start the Wazuh agent on WebServer-01. The step will contact the

agent-auth



/

manager at 172.25.0.50:1515 and register the agent. If successful, you should see on the manager (in

var/ossec/logs/ossec.log

) a message about a new agent registration and on the agent side a confirmation of successful key negotiation. The agent will then start sending data (heartbeat, system logs, etc.) to the manager on UDP 1514.

Note: If the manager had been configured to require a registration password or pre- created keys, the above step might fail. By default, Wazuh 4.x allows open registration. If needed, one could generate an agent key on the manager (via the Wazuh API or

manage\_agents

agent-auth -P <key>

) and use settings for simplicity.

Texto

El contenido generado por IA puede ser incorrecto.

*Figure 11: install\_wazuh\_agent\_linux.yml execution.*

Windows Agents (FileServer-01 & DomainController): For Windows, Wazuh provides an MSI installer.

We will download the Wazuh agent MSI (version 4.4.5 in this example) and install it silently. We pass an

argument to the MSI to point the agent to the manager IP (ADDRESS=172.25.0.50). We will also install

Sysmon to enhance logging. Sysmon will be configured with SwiftOnSecurity’s popular configuration to log process creations, network connections, etc., to the Windows Event Log, which Wazuh will monitor.

The tasks below should be run on both Windows hosts (we can target an Ansible group that contains FileServer-01 and DomainController). They will perform the following on each host: - Download the Wazuh agent MSI. - Install the Wazuh agent (silent mode, pointing to manager IP). - Download Sysmon and a standard Sysmon config XML. - Install Sysmon as a service with that config. - Modify the Wazuh

ossec.conf

agent config ( ) to ensure it captures the Sysmon Event Log channel.

Filename:

install\_wazuh\_agent\_windows.ps1

Windows endpoints)

(Script for Wazuh agent and Sysmon on

# Create Temp directory

New-Item -Path "C:\Temp" -ItemType Directory -Force | Out-Null

# Download Wazuh Agent MSI

Invoke-WebRequest -Uri "https://packages.wazuh.com/4.x/windows/wazuh-agent-4.4.5-1.msi" -OutFile "C:\Temp\wazuh-agent.msi"

# Install Wazuh Agent

Start-Process -FilePath "msiexec.exe" -ArgumentList "/i C:\Temp\wazuh-agent.msi /q ADDRESS=172.25.0.50" -Wait

# Download Sysmon ZIP

Invoke-WebRequest -Uri "https://download.sysinternals.com/files/Sysmon.zip" -OutFile "C:\Temp\Sysmon.zip"

# Extract Sysmon

Expand-Archive -Path "C:\Temp\Sysmon.zip" -DestinationPath "C:\Temp\sysmon" -Force

# Download Sysmon configuration

Invoke-WebRequest -Uri "https://raw.githubusercontent.com/SwiftOnSecurity/sysmon-config/master/sysmonconfig-export.xml" -OutFile "C:\Temp\sysmonconfig.xml"

# Install Sysmon with config

Start-Process -FilePath "C:\Temp\sysmon\Sysmon64.exe" -ArgumentList '-accepteula -i C:\Temp\sysmonconfig.xml' -Wait

# Modify Wazuh agent config to enable Sysmon event channel

$confPath = "C:\Program Files (x86)\ossec-agent\ossec.conf"

$line = '<localfile><log\_format>eventchannel</log\_format><location>Microsoft-Windows-Sysmon/Operational</location></localfile>'

if ((Get-Content $confPath) -notcontains $line) {

(Get-Content $confPath) | ForEach-Object {

$\_

if ($\_ -match '<localfile>') {

$line

}

} | Set-Content $confPath

}

Once the above tasks run on both Windows hosts, they will each have: - Wazuh agent installed and (if the manager is allowing auto-enrollment) likely already attempting to register with the manager. Because we passed the manager address in the installer, the agent knows where to connect. However, by default, Windows agents do not auto-register without a key in Wazuh 4.x unless the manager is in open mode. If the agent does not appear in the Wazuh console automatically, we may need to register it. You have two main options: 1. Manual enrollment via Wazuh UI: Open the Wazuh app in Kibana on the SIEM-Node. Go to Management -> Agents, and add a new agent for each Windows host (specify their names). This will generate a one-time registration key. Then, on the Windows host, run the Agent

agent-auth.exe

Setup program (installed with the agent) or run with that key to register. For

example, on the FileServer, you could run in an Administrator command prompt: \"C:\\Program

Files (x86)\\ossec-agent\\agent-auth.exe\" -m 172.25.0.50 -A FileServer-01 -P

<key> (substitute the key provided). This will register the agent and start communication. 2. Open enrollment: Configure the Wazuh manager to allow any agent with the correct address to auto-register. By default, the manager’s ossec.conf allows agent-auth requests without a pre-shared key, so simply running agent-auth.exe -m 172.25.0.50 on the Windows host might work. In our lab, since we already used the Linux agent-auth successfully without a key, the same should work on

win\_shell

Windows. You could invoke this via Ansible as well (e.g., a

agent-auth.exe

), but it may require adjusting the path or environment.

task running

For simplicity, using the Wazuh Kibana UI to finalize Windows agent enrollment is often easiest in a training scenario – it allows students to see the agent registration process. There should be no gap in instruction: if you find the Windows agents are not showing up in the Wazuh console, generate keys in the UI and run the provided command on the Windows hosts to register them.

* Sysmon is installed as a service on each Windows host. It will log events (like process creation,

**Microsoft-Windows-Sysmon/Operational**

network connections, etc.) to event log. We

modified the Wazuh agent config to collect that log channel, so these events will be forwarded to Wazuh manager and become part of the security monitoring.

* The Wazuh agent services on Windows should be running (they start automatically after

installation). You can verify on each Windows host that the agent is running by checking *Services*

Wazuh Agent

agent\_control.exe

(service name app.

) or using

or the *Agent* menu in the Wazuh

## Verification and Next Steps

At this point, the Operation RedTrace environment should be fully deployed. Perform the following verification steps to ensure everything is working as expected:

* Wazuh Manager and Agents: In the Kibana Wazuh app (on SIEM-Node at http://172.25.0.50:5601), navigate to the Agents section. You should see entries for the Linux web server and the Windows hosts. They might initially appear as disconnected or pending if not registered; ensure they become Active (green status). If not, use the enrollment steps described above to register any missing agents. Once active, each agent will start sending data. You can test an alert by, for example, on the SIEM-Node running (to generate a test alert), or triggering a known event. An easy test is on a Windows agent, run a command that triggers Sysmon (like opening a command prompt or creating a file) and then check Kibana > Discover or Wazuh app Alerts to see if any logs from Sysmon are indexed.

/var/ossec/bin/ossec-logtest

* Caldera Access: Access the Caldera web interface on the Kali machine at http://172.25.0.10:8888. Log in (default creds: admin / admin). Make sure the Caldera server is running properly. At this time, no Caldera agents (called "Sandcats") are deployed on targets yet. To simulate the attack, you will need to deploy the Caldera agent on one or more targets. Caldera can generate payloads for various platforms. For instance, you can go to Agents >

10.10.0.10:8888

* \"Deploy\" within Caldera to generate a Windows payload (PowerShell or EXE) for the FileServer, and a Linux payload (bash or ELF) for the WebServer. Execute those payloads on the respective machines (this can be done manually or via some scripted method) to establish a callback to Caldera. Once the Sandcat agents check in, they will appear in Caldera’s interface as new agents.
* Adversary Simulation: With agents in place, you can now run a predefined operation in Caldera that emulates the APT “RedTrace” actions. Ensure to map the Caldera agents to the appropriate platforms (one agent on Windows, one on Linux). The scenario likely involves steps like initial compromise on the web server, dumping credentials, moving laterally to the file server, etc., using Caldera abilities. Launch the operation in Caldera and monitor its progress.
* Monitoring and Detection: While the Caldera operation is running, switch to the Wazuh app (Kibana) and observe the alerts being generated. Wazuh, with its out-of-the-box rules (and the Sysmon events), should catch several malicious activities. For example, creation of a new user (if simulated), suspicious PowerShell execution, modifications to the registry, etc., depending on what Caldera does. You may need to adjust Wazuh rules or add custom rules to tag certain behaviors (the scenario description mentions custom detection rule tuning – for instance, detecting the specific techniques used by the RedTrace adversary).
* ELK Kibana Discover: You can also use the Kibana Discover tab to query the raw indices (e.g.,

wazuh-alerts-4.x-\*

) for events if needed.

By the end of the exercise, you should have executed a realistic cyber attack simulation and collected an audit trail of detection alerts. Trainees can analyze which actions were detected by Wazuh and which were missed, correlating them with MITRE ATT&CK techniques (Caldera will label the techniques used). This provides insight into defensive gaps and improvements.

# Conclusion

In this guide, we deployed the Operation RedTrace – KYPO Edition scenario from scratch, using automation to configure a multi-machine cyber range environment. We set up a Kali-based Red Team server with MITRE Caldera, a Ubuntu-based SIEM with Wazuh and ELK, and a Windows domain environment with a Domain Controller and a member server, both instrumented with Wazuh agents and Sysmon. Each step was detailed to ensure reproducibility and accuracy within the KYPO platform. By following this guide, an instructor or learner can recreate the scenario reliably and proceed to conduct the Red vs Blue team exercise.

The environment is now ready for use: launch the Caldera operations to emulate the RedTrace APT, and use the Wazuh SIEM to detect and investigate the malicious activities. Good luck, and enjoy the hands- on experience of this purple team exercise!