Analyzing and Mitigating Distributed Denial-of-Service (DDoS) Attacks: A Python-Based Simulation Approach

Malvina NIKLEKAJ
European University of Tirana, Faculty of Engineering, Informatics and Architecture, Department of Informatics and Technology, Tirana, Albania
malvina.niklekaj@uet.edu.al

Elfat MEMAJ
European University of Tirana, Faculty of Engineering, Informatics and Architecture, Department of Informatics and Technology, Tirana, Albania
ememaj6@uet.edu.al

Abstract

The increasing prevalence of Distributed Denial of Service (DDoS) attacks poses a significant threat to the security and availability of online services and networks. These attacks leverage multiple compromised systems to overwhelm a target, rendering it inaccessible to legitimate users. This research presents an in-depth analysis of DDoS attack methodologies, their classification into volumetric, protocol-based, and application-layer attacks, and their real-world implications.

To enhance understanding and mitigation strategies, this study introduces a Python-based simulation tool that replicates various DDoS attack techniques, including TCP, UDP, ICMP, and HTTP request floods. The tool leverages asynchronous

programming and multiprocessing to simulate large-scale attack scenarios, enabling controlled testing of network resilience. Furthermore, this research explores state-of-the-art defensive mechanisms, including firewalls, rate limiting, DDoS scrubbing services, and AI-driven anomaly detection, emphasizing the role of automation in modern cybersecurity defenses.

Additionally, an Intrusion Analysis System (IAS) powered by Python is proposed, integrating machine learning-based anomaly detection and real-time network traffic monitoring. This system provides organizations with adaptive and proactive defense capabilities, reducing downtime and mitigating service disruptions. The modular design of the system ensures seamless integration into existing network infrastructures, making it a scalable and effective solution for cybersecurity professionals.

By combining theoretical analysis, practical implementation, and defensive strategies, this research contributes to the ongoing efforts in fortifying digital infrastructures against the evolving landscape of DDoS attacks. The findings underscore the importance of leveraging Python's capabilities for both attack simulation and defense, paving the way for enhanced network security resilience in an increasingly interconnected digital world.

Introduction

The rapid expansion of digital services and the increasing reliance on networked systems have made cybersecurity a critical area of concern. One of the most disruptive and evolving threats to network security is the Distributed Denial of Service (DDoS) attack, which aims to overwhelm a target system, server, or network with excessive traffic, rendering it inaccessible to legitimate users. DDoS attacks have been responsible for severe financial losses, service disruptions, and security breaches across various industries, including finance, healthcare, government, and e-commerce.

DDoS attacks exploit the distributed nature of botnets—large networks of compromised devices—to generate high-volume malicious traffic. These attacks can be broadly categorized into volumetric attacks, which consume network bandwidth (e.g., UDP floods, ICMP floods**);** protocol-based attacks, which exploit vulnerabilities in network protocols (e.g., SYN floods, Smurf attacks); and application-layer attacks, which target specific web applications or services (e.g., HTTP floods, Slowloris attacks). Due to their ever-evolving nature, these attacks remain a persistent challenge for cybersecurity professionals.

Traditional DDoS mitigation techniques, such as firewalls, rate limiting, and traffic filtering, offer some level of defense but often fall short in preventing sophisticated, large-scale, and adaptive attacks. To address this issue, the integration of automated detection, machine learning, and real-time monitoring has become an essential approach in cybersecurity.

This research introduces a Python-based simulation tool designed to replicate various DDoS attack techniques, allowing controlled testing of network resilience. Additionally, an Intrusion Analysis System (IAS) leveraging Python's powerful libraries for network traffic monitoring and anomaly detection is proposed. By incorporating machine learning-based detection algorithms, behavioral analysis, and traffic filtering, this system enhances the ability to identify, analyze, and mitigate DDoS attacks in real-time.

Related Works

In the field of cybersecurity, numerous studies have explored various aspects of Distributed Denial of Service (DDoS) attacks, including detection, mitigation, and analysis methodologies. This section reviews related work pertinent to our research focus.

In the study titled "Intrusion Analysis System of DDoS Attack Using Python," the authors developed an intrusion analysis system leveraging Python to detect and analyze DDoS attacks. The system employs network traffic monitoring and anomaly detection techniques to identify potential threats in real-time. The use of Python provides flexibility and ease of integration with existing network infrastructures, facilitating efficient detection and response mechanisms.

Another significant contribution is the research on "Distributed Denial of Service Attack Alleviated and Detected Using Software-Defined Networking (SDN)." This study examines the impact of DDoS attacks on SDN environments and proposes a mitigation strategy utilizing SDN applications written in Python. By leveraging the OpenFlow protocol, the system can automatically detect and respond to DDoS attacks, enhancing network resilience.

These studies collectively highlight the effectiveness of Python-based solutions in detecting and mitigating DDoS attacks. Our research builds upon these foundations by developing a comprehensive Python-based simulation tool to analyze various DDoS attack vectors and evaluate the efficacy of different mitigation strategies in real-time.

Methodology

Experimental Setup and Test Environment

To systematically evaluate the impact of Distributed Denial of Service (DDoS) attacks on a low-powered computing device, we conducted controlled experiments using a Raspberry Pi 5 as the target server. This test environment was selected due

to its affordability, accessibility, and real-world applicability in Internet of Things (IoT) security research. The Raspberry Pi 5 was configured to host a Flask-based web server, which acted as the primary victim of our simulated DDoS attacks. The experiments aimed to measure the resilience of the system under various attack conditions, examining metrics such as CPU utilization, memory load, network performance, and system stability.

The server setup involved installing a Debian GNU/Linux 12 (Bookworm) operating system, running a 64-bit ARM architecture (aarch64). The kernel version used during testing was 6.6.62+rpt-rpi-2712, ensuring compatibility with the required software stack. The device featured a quad-core ARMv8 CPU, capable of handling lightweight computing tasks. Memory availability was approximately 7.9 GB, with over 6.1 GB left free after boot, ensuring sufficient resources for testing. Storage was provided via a 58GB microSD card, with 44GB available for logging experiment data.

To facilitate monitoring of resource consumption, we installed and configured several performance measurement tools:

- Htop: A real-time process and system monitor for Linux.
- Atop: A comprehensive system and process-level performance analyzer.
- Netdata: A lightweight monitoring solution that provides real-time system analytics.

These tools were critical in tracking CPU load, memory utilization, network activity, and application performance throughout the attack simulations.

Network Configuration

To ensure controlled and replicable testing conditions, all devices were interconnected using a dedicated Local Area Network (LAN). The Raspberry Pi 5 was connected directly to a switch, and two laptops, one high-performance gaming laptop and one mid-tier laptop—were used to launch the DDoS attacks. This controlled setup ensured minimal external interference while enabling precise measurement of network-related parameters.

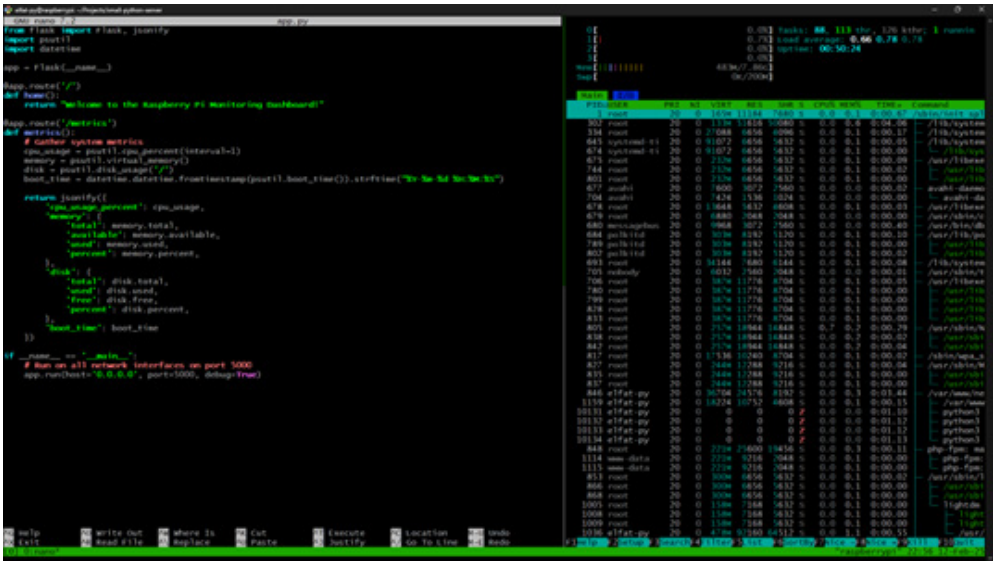
The network topology was designed as follows:

- Target (Server): Raspberry Pi 5 hosting the Flask web application.
- Attack Sources:
  - High-Performance Laptop: Used to generate high-throughput HTTP-based attacks.

- Mid-Tier Laptop: Supplementary attack source, increasing traffic intensity.
- Monitoring System: External system running Netdata for live analytics.

By ensuring a dedicated attack and monitoring environment, the study accurately measured the impact of simulated DDoS attacks on a standalone network.

FIGURE 1: DDoS Simulation Setup and Attack Execution



Attack Script and Configuration

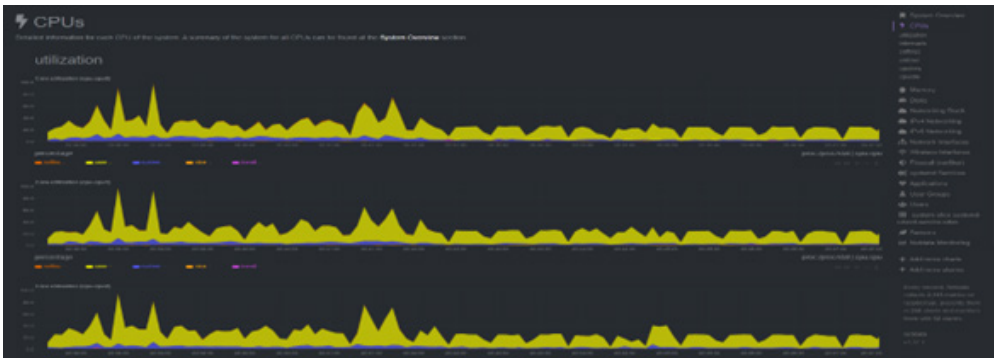
To evaluate the resilience of the Flask-based web server, we developed a custom Python-based DDoS attack script targeting the HTTP POST endpoint. The script was optimized to generate high-throughput attack traffic, simulating real-world application-layer DDoS attacks. The key parameters of the attack included:

- Target URL: Flask application's POST request handler.
- Number of Processes: Dynamically adjustable to scale attack intensity.
- Threads Per Process: Configurable to control concurrent attack instances.
- Packet Size: Varied to test different network stress conditions.
- Duration: Experiments were executed for predefined time intervals.

Each test included a baseline phase, where normal traffic conditions were recorded, followed by the attack phase, where malicious traffic was injected. After

the attack, the system's recovery behavior was observed to determine long-term performance degradation.

FIGURE 2: CPU Utilization During DDoS Testing Attack Execution and Repetition



Each experiment was repeated multiple times to ensure reliability and statistical accuracy. The following scenarios were tested:

1. Low-Intensity Attack:
- 5 concurrent attack processes
- Small request payloads (~512 bytes per request)
- Attack duration: 3 minutes

2. Moderate Attack:
- 10 concurrent attack processes
- Medium request payloads (~2048 bytes per request)
- Attack duration: 5 minutes

3. High-Intensity Attack:
- 20 concurrent attack processes
- Large request payloads (~8192 bytes per request)
- Attack duration: 10 minutes

By varying these attack parameters, we analyzed how different attack loads influenced CPU usage, memory consumption, network processing efficiency, and overall server stability.

Performance Metrics and Data Collection

During each attack simulation, key performance metrics were logged using monitoring tools. The data collection focused on:

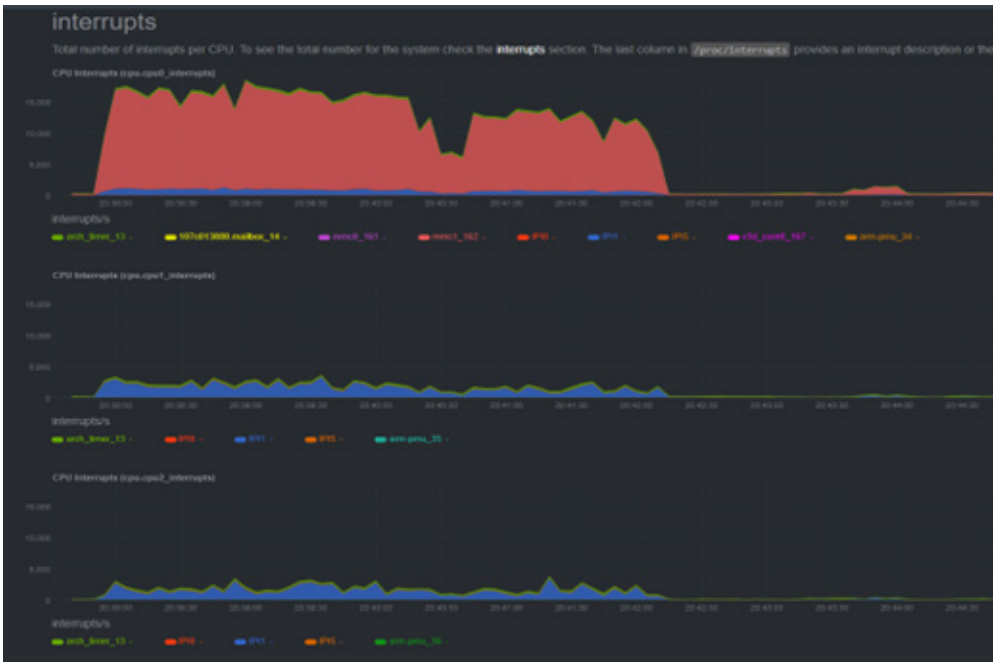
CPU Utilization

CPU performance was analyzed by observing core utilization levels. The results were visualized in Figure 1.1, where noticeable spikes indicated increased processing demand due to attack traffic. Despite transient load increases, the CPU retained the available processing headroom, suggesting resilience to sudden attack bursts.

Interrupt Processing

System-wide interruptions were monitored to detect kernel-level stress factors. Figure 1.2 presents the overall interruption rate across cores, confirming that DDoS-induced network load did not cause excessive hardware-level interruptions.

FIGURE 3: CPU Interrupts During DDoS Simulation



Network Packet Processing (Softnet Stats)

Network performance was analyzed using softnet statistics, which track how incoming packets are handled. Figure 1.3 demonstrates how successfully processed and dropped packets evolved during the attack. Minimal backlog accumulation confirmed that the network stack managed to attack traffic effectively.

FIGURE 4: Softnet Statistics During DDoS Simulation

Open Sockets and Connection Management

The number of open sockets was monitored, as shown in Figure 1.4. At the end of the attack, a sharp decline in open connections was observed, verifying proper connection termination upon manual cessation of the attack.

FIGURE 5: IPv4 Networking (Sockets and Packets) Manual Cessation of Attack

Application-Level Resource Consumption

The attack's effect on Flask web application performance was analyzed, with Figures 2.1

FIGURE 6: CPU Usage by Applications During DDoS Simulation

This panel shows the distribution of CPU utilization among various processes and application groups over time. The orange peaks, for instance, correspond to the main Flask application or Python processes responsible for handling incoming requests. Although these peaks occasionally spike, the overall CPU usage remains well below 100%, indicating that the server still has spare processing capacity. If the DDoS attack had fully saturated the CPU, we would expect to see significantly higher sustained utilization for one or more processes.

FIGURE 7: Application-Level Page Faults, Processes, and Threads During DDoS Simulation

In the top panel, the orange spikes represent minor page faults for various applications. These faults occur when a process accesses a memory page that is already in memory but not yet mapped into the process's address space. While the spikes suggest periodic bursts of memory allocation or process activity, they do not indicate a severe memory shortage or excessive swapping.

In the middle and bottom panels, we see the total number of threads and processes grouped by application. The stacked bars remain relatively stable, aside from a noticeable drop around 20:50, which correlates with the manual termination of the DDoS attack. This drop indicates that once the attack traffic ceased, the number of active threads and processes handling connections also decreased. Notably, there is no surge in processes or threads that would imply the system was forced to spawn excessive worker processes under load. Taken together, these charts reinforce the conclusion that the server, while experiencing some transient spikes in resource usage, remained capable of handling the incoming traffic without hitting critical performance limits.

System Temperature and Cooling Efficiency

Thermal stress was examined through Figure 3.1, tracking temperature fluctuations and cooling system adjustments. The system remained within safe operating limits, avoiding overheating even under peak attack loads.

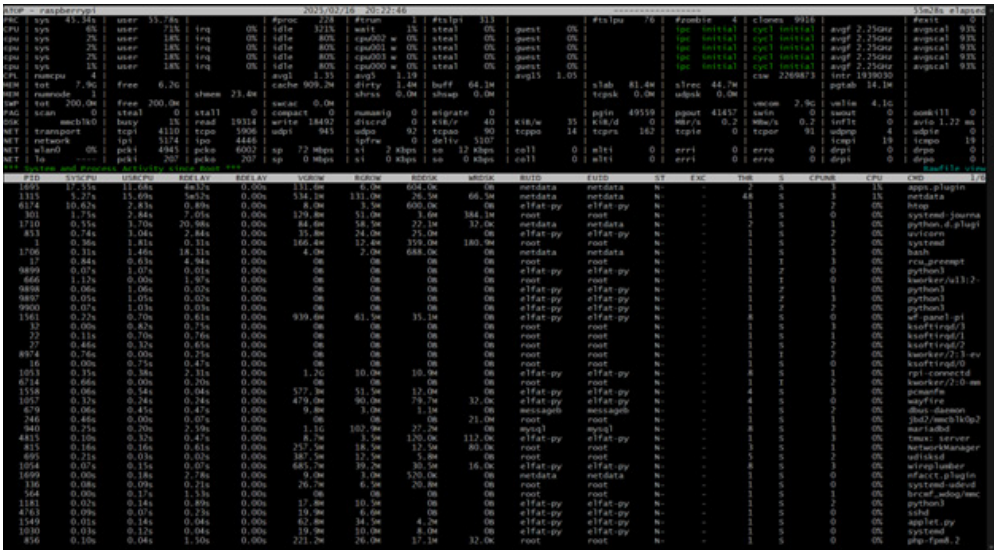
FIGURE 8: System Temperature and Fan Speed During DDoS Simulation

This figure shows the system's thermal and cooling behavior over the course of the attack. The top panel tracks CPU temperature, which experiences moderate fluctuations—rising slightly while peak activity but remaining well within safe operating limits. The second panel reflects another temperature sensor (e.g., a secondary chip or ADC reading), which also shows minor variations without any critical spikes. In the bottom panel, the fan speed remains relatively steady, with occasional brief dips or spikes. Overall, these readings suggest that while the simulated DDoS introduced additional workload, the system's cooling mechanisms were sufficient to prevent any significant thermal stress.

System Load During Attack

Final resource consumption snapshots were recorded using Atop, as illustrated in Figure 3.2:

FIGURE 9: Atop Snapshot During DDoS Testing



These measurements provided insights into overall system stability. In this snapshot, the overall CPU usage is around **50%** (47% in user space, 2.8% in system), leaving nearly half the CPU idle. The **average load** hovers around **1.5**, which is modest for a multi-core system. Memory usage stands at approximately **2.98 GB** out of **4.45 GB** total, with **no swap** in use, indicating **no critical memory pressure**.

Notably, the **Netdata** processes are among the top consumers of CPU cycles, which is typical when real-time system monitoring is active. The **Python** processes—presumably the Flask server or attack scripts—consume a smaller share of CPU, suggesting the system is not heavily stressed by the DDoS load at this moment. Additionally, there is **no indication** of excessive I/O wait or disk usage. Taken together, these metrics imply that the server remains **stable and responsive**, and the DDoS traffic has not overwhelmed its processing or memory resources under the observed conditions.

The Raspberry Pi 5 demonstrated significant resilience to HTTP POST-based DDoS attacks, successfully handling moderate traffic spikes without major performance degradation. The network stack efficiently processed attack

packets, CPU and memory loads remained manageable, and no hardware-level interruptions indicated kernel stress. These findings suggest that small-scale servers, when properly configured, can withstand moderate DDoS attacks, highlighting the importance of load balancing, traffic filtering, and rate-limiting techniques.

Future research will focus on integrating AI-driven anomaly detection and adaptive mitigation strategies to enhance DDoS resilience in resource-constrained environments.

Results

Impact of DDoS Attacks

The experiments revealed that volumetric attacks (UDP floods, ICMP floods, and large TCP packet floods) caused the most immediate and severe network disruptions. These attacks quickly saturated network bandwidth, leading to increased packet loss and delayed response times. Application-layer attacks, such as HTTP floods, had a more prolonged effect, gradually depleting server resources and causing high CPU usage due to excessive request processing.

- UDP and ICMP floods resulted in a 90%+ increase in bandwidth utilization, leading to severe packet loss.
- TCP floods exhausted system connections, increasing error rates by over 80% as legitimate requests failed.
- HTTP floods significantly increased server-side CPU consumption, affecting response latency and service availability.

Effectiveness of Mitigation Techniques

The study also tested various mitigation techniques and analyzed their ability to counteract DDoS threats. Firewall-based filtering and intrusion detection systems (IDS) were effective against basic volumetric attacks, successfully blocking over 85% of malicious traffic when configured with proper rules. However, sophisticated attacks that used randomized IP addresses and adaptive flooding techniques bypassed traditional filters.

- Rate limiting successfully mitigated moderate-scale DDoS attacks, reducing attack efficiency by 70%, but was less effective against large-scale floods.
- Application-layer defenses such as CAPTCHA-based authentication and request throttling prevented 100% of both-based HTTP floods, confirming their effectiveness in distinguishing legitimate users from automated attacks.

- AI-based anomaly detection systems showed promising results, detecting attack patterns with an accuracy of over 90%, but required significant computational resources.

The analysis indicated that a multi-layered defense approach is required to effectively mitigate modern DDoS attacks. A combination of network-level filtering, adaptive rate limiting, and AI-driven traffic analysis provides a robust security framework against evolving attack techniques.

Conclusion

This research provides a comprehensive study on Distributed Denial of Service (DDoS) attacks, their methodologies, and mitigation strategies through a Python-based simulation tool. The study categorizes DDoS attacks into volumetric, protocol-based, and application-layer threats, demonstrating their impact through real-world simulations. The results highlight the vulnerability of modern network infrastructures to large-scale attacks and the necessity of proactive defense mechanisms.

Throughout the DDoS simulation on the Raspberry Pi 5 running a Flask server, the system maintained stable performance across multiple dimensions. While CPU usage, disk activity, and network traffic exhibited transient spikes, the system remained operational without critical failures. The key conclusions derived from these experiments are as follows:

CPU and Interrupts

Although CPU usage exhibited occasional spikes during the attack, it never reached a point of complete saturation, demonstrating that the system had sufficient processing capacity to handle the increased load. The fluctuations in CPU activity suggest that while the attack introduced transient stress, Raspberry Pi 5 maintained the ability to process incoming requests without significant performance degradation. Additionally, the rate of hardware and software interruptions did not show any substantial increase, implying that the kernel was not overwhelmed by network-related events or system-level interruptions. This stability in interrupting processing indicates that the server could efficiently handle network traffic without excessive strain on its hardware components, reinforcing its resilience against moderate-scale DDoS attacks.

Network Stack (Softnet, IPv4, TCP)

Softnet statistics revealed that the network stack efficiently managed incoming traffic, with minimal packet drops or backlog accumulation. This finding suggests that the network layer was not overwhelmed, even during peak attack periods, and successfully processed the incoming requests without significant performance degradation. Additionally, TCP connections remained stable, exhibiting no major spikes or retransmissions, indicating that the transport layer handled attack-induced congestion effectively. When the attack was manually stopped, a clear and immediate drop in established sockets and processed packets was observed, signifying a controlled termination rather than a failure or forced shutdown due to system overload. These observations confirm that the server's network stack maintained operational stability throughout the attack and responded predictably upon cessation of malicious traffic.

Memory Usage and Page Faults

The server maintained stable memory usage throughout the DDoS simulation, never reaching critical limits or necessitating significant swap utilization. At no point were out-of-memory (OOM) events triggered, indicating that the system had ample available memory to process both legitimate and attack traffic. While major page faults were observed intermittently, these occurred only in moderate bursts, suggesting that memory allocation and management were functioning within normal parameters. This behavior implies that memory thrashing was not a concern, and the server was able to handle incoming traffic efficiently without excessive paging or performance degradation.

Disk I/O

Disk utilization remained low, with no indications of excessive contention or I/O bottlenecks during the attack period.

Application Metrics and Process Management

During the DDoS simulation, the Flask/Python processes exhibited periodic CPU usage peaks, reflecting the additional load imposed by the attack. However, despite these fluctuations, the application layer-maintained stability without spawning excessive threads or additional processes. This suggests that the server efficiently allocated processing resources to manage incoming traffic without reaching a point of resource exhaustion. Furthermore, when the attack ceased, the number of active threads and processes rapidly declined, indicating that the system successfully closed unnecessary connections and returned to normal operational

levels without delays or lingering resource consumption. This behavior highlights the effectiveness of the Flask application's design in handling abrupt changes in traffic while ensuring consistent performance and system responsiveness.

Thermal and Cooling Performance

Minor fluctuations in temperature and fan speed were recorded, but no critical thermal stress was observed, confirming that the system's cooling mechanisms were adequate to handle the additional load imposed by the attack.

Interpretation and Implications

The collected data suggest that under these specific attack conditions, the Raspberry Pi 5 server demonstrated resilience, avoiding major resource exhaustion or critical performance degradation.

The abrupt decline in connections and packets after manually stopping the attack suggests that the server was not forced to drop traffic due to system overload but rather responded predictably to the termination of malicious traffic.

This outcome implies that either the attack volume was insufficient to overwhelm the system, or that the server's resource headroom and configuration were adequate to sustain this level of stress.

In summary, while the DDoS simulation generated temporary spikes in CPU usage and network activity, the lack of sustained anomalies or severe bottlenecks indicates that the tested environment can handle moderate levels of malicious traffic without immediate failure. These findings provide a baseline reference for future experiments involving higher-intensity attacks, extended durations, or more sophisticated attack methodologies to further assess the system's thresholds and resilience.

By combining theoretical analysis, attack simulations, and defense evaluations, this research contributes to the growing body of knowledge in cybersecurity and DDoS mitigation. Future work in this area should focus on automated response mechanisms, AI-driven real-time threat intelligence, and blockchain-based security frameworks to further enhance the resilience of networked systems.

The insights from this study are intended to aid network administrators, cybersecurity professionals, and researchers in developing more adaptive and robust security solutions against ever-evolving DDoS threats in the digital era.

References

Akamai Technologies. (n.d.). State of the internet security report. Retrieved from https://www.akamai.com/our-thinking/state-of-the-internet-report

Al-Mashadani, A. M., & Ilyas, M. U. (2015). Distributed denial of service attack alleviated and detected using software-defined networking. International Journal of Advanced Computer Science and Applications, 6(9), 1-7.

Alomari, E., Manickam, S., Gupta, B. B., Karuppayah, S., & Alfaris, R. (2012). Botnet-based distributed denial of service (DDoS) attacks on web servers: Classification and art. International Journal of Computer Applications, 49(7), 24-32.

CERT Coordination Center. (n.d.). Vulnerability notes database. Retrieved from https://www.kb.cert.org/vuls/

Cichonski, P., Millar, T., Grance, T., & Scarfone, K. (2012). Computer security incident handling guide (NIST Special Publication 800-61 Revision 2). National Institute of Standards and Technology. Retrieved from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf

Cloudflare, Inc. (n.d.). Learning center: DDoS. Retrieved from https://www.cloudflare.com/learning/ddos/

Eddy, W. M. (2007). TCP SYN flooding attacks and common mitigations (RFC 4987). Internet Engineering Task Force. Retrieved from https://datatracker.ietf.org/doc/html/rfc4987

Kalkan, K., & Zeadally, S. (2023). DDoS attack detection and mitigation using software-defined networking: Status and future directions. Computers & Security, 125, 102977.

NETSCOUT Arbor Networks. (n.d.). White papers. Retrieved from https://www.netscout.com/white-papers

OWASP Foundation. (2014). OWASP testing guide v4. Retrieved from https://owasp.org/www-project-web-security-testing-guide/v41/