Implementing a DoS Attack through HTTP Flood using Metasploit Framework

Submitted in partial fulfillment of the requirements of
Introduction to Computer Networks

ABSTRACT

This project focuses on implementing a DoS attack using HTTP flood and Metasploit Framework on an Ubuntu virtual machine. The target is the Apache web server hosting a vulnerable web application, DVWA. The goal is to overwhelm the server with a high volume of requests and assess the effectiveness of its security measures. Through practical experimentation, we gain insights into DoS vulnerabilities, mitigation techniques, and network security. The project contributes to understanding the risks and defenses against DoS attacks.

Table of Contents

Table Of Figures

SECTION I: INTRODUCTION

In today's interconnected world, network security is of paramount importance to ensure the availability and integrity of online services. As part of our Networks course project, we aim to explore the concept of Denial-of-Service (DoS) attacks, specifically focusing on a type known as HTTP flood. This project will involve simulating a controlled DoS attack on a web server, Apache, by overwhelming it with a high volume of HTTP requests. It is crucial to note that the purpose of this project is solely educational and should not be replicated or used for any malicious intent. Understanding the vulnerabilities and potential risks associated with DoS attacks is essential for network administrators, security professionals, and developers in order to design robust systems and implement appropriate security measures.

SECTION II: METASPLOITABLE & DVWA

Metasploit is a versatile penetration testing platform widely used by security professionals and ethical hackers to identify, exploit, and validate vulnerabilities in various systems and applications. It offers a comprehensive suite of tools, exploit modules, payloads, and auxiliary modules that aid in conducting penetration tests and performing thorough security audits.

One of the key features of Metasploit is its extensive database of known vulnerabilities, which enables users to search for specific weaknesses in target systems. The framework provides a unified interface and scripting language that simplifies the process of executing exploits and automating tasks. It supports multiple operating systems and offers flexibility in customizing and extending its functionalities.

Metasploit's modular architecture allows users to combine different components to create tailored attack scenarios. It provides an environment for safely emulating attacks, validating vulnerabilities, and testing the effectiveness of security measures.

Figure 1: Metasploitable

The Damn Vulnerable Web Application, or DVWA, is a purposely vulnerable web application designed for educational and training purposes. It is specifically created to provide a safe and controlled environment for individuals to learn about web application vulnerabilities and practice penetration testing techniques.

DVWA consists of a front-end, back-end, web server, and operating system, all intentionally configured with known vulnerabilities. These vulnerabilities include common issues like SQL injection, cross-site scripting (XSS), command injection, file inclusion, and more. By exploring DVWA, users can gain hands-on experience identifying and exploiting these vulnerabilities, thereby enhancing their understanding of web application security. The application serves as a practical learning tool to simulate real-world scenarios and understand the potential risks associated with insecure coding practices and misconfigurations.

Figure 2: Damn Vulnerable Web App

When combined with the Metasploit Framework, DVWA provides an ideal environment for testing and validating exploits. By utilizing Metasploit's capabilities, security professionals can demonstrate the impact of vulnerabilities present in DVWA, evaluate the effectiveness of security measures, and develop strategies to mitigate these vulnerabilities in real-world applications.

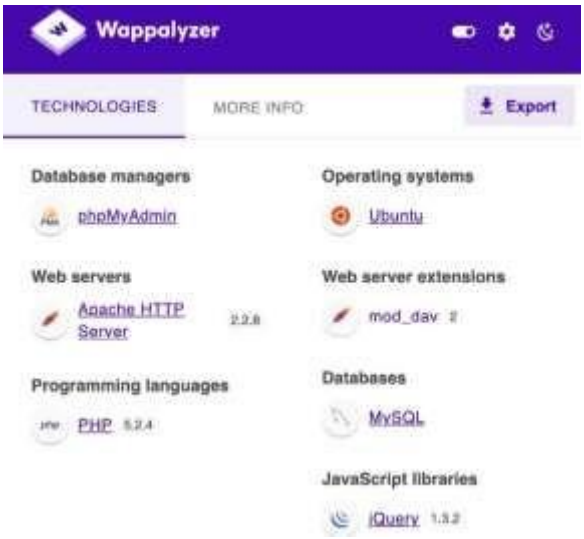

Figure 3: Front end, Back end, Webserver, OS

SECTION III: TYPES OF DoS

There are several types of Denial-of-Service (DoS) attacks, each with its own characteristics and methods. Here are some common types of DoS attacks:

1. **Ping Flood:** Also known as ICMP flood, this attack involves overwhelming the target system with a flood of Internet Control Message Protocol (ICMP) echo request packets (pings). The target becomes overloaded with responding to these requests, causing a denial of service.

2. **SYN Flood:** In a SYN flood attack, the attacker exploits the TCP three-way handshake process. They send a large number of SYN packets to the target system, but do not complete the handshake by sending the final ACK packet. This leaves the target with half-open connections, consuming its resources and making it unable to respond to legitimate requests.

3. **HTTP Flood:** In an HTTP flood attack, the attacker directs a massive amount of HTTP requests to the target web server. This flood of requests consumes the server's resources such as bandwidth, CPU, and memory, making it unable to handle legitimate user traffic.

4. **Distributed Denial-of-Service (DDoS):** A DDoS attack involves multiple sources, often a botnet, coordinating simultaneous attacks on a target. By distributing the attack traffic across multiple sources, the attacker amplifies the impact, making it harder to mitigate and defend against.

In our project, we focused on implementing the HTTP flood technique as a type of Denial-of-Service (DoS) attack. The HTTP flood attack involved overwhelming the target web server with a high volume of HTTP requests. By flooding the server with these requests, we aimed to exhaust its resources, such as bandwidth, processing power, and memory, ultimately leading to a denial of service for legitimate users.

SECTION IV: DoS MITIGATION TECHNIQUES

To mitigate the impacts of a DoS incident, consider the following options:

1. **Traffic scrubbing:** Utilize DDoS protection services that act as intermediaries to filter out malicious traffic before it reaches your system.

2. **Source or location blocking:** Add IP addresses, CIDR ranges, or geolocations to a deny list on network devices or CDNs to block traffic from specific sources.

3. **Pattern and behavior blocking:** Block traffic that matches specific patterns or deviates from normal user behavior, such as oversized or malformed packets.

4. **Disabling dynamic functions:** Temporarily disable resource-intensive dynamic content during an attack to reduce the load on backend servers.

5. **Displaying CAPTCHA challenges:** Implement CAPTCHA challenges for suspicious requests to distinguish between bots and legitimate users.

Ensure that the mitigation measures are set to activate at appropriate thresholds. Preparedness and understanding the capacity of your infrastructure are crucial for timely response during a DoS event.

SECTION V: METHODOLOGY

Our implementation was carried out on an Ubuntu virtual machine, where we set up a vulnerable web application called DVWA (Damn Vulnerable Web Application). DVWA consists of a front-end, back-end, web server, and operating system, all of which have security measures in place to detect and mitigate attacks. The objective of our project was to test the resilience of this system to a DoS attack and assess the effectiveness of its security mechanisms.

To execute the DoS attack, we leveraged the powerful Metasploit Framework. Metasploit is an open-source penetration testing framework that provides a wide range of tools and exploits to assess the security of various systems and applications. By utilizing Metasploit's capabilities, we demonstrated how an attacker could flood the Apache web server with a massive influx of HTTP requests, overwhelming its capacity and causing a denial of service for legitimate users.

Our implementation of the DoS Attack was carried out according to the following steps:

1. **Setup Environment (virtual lab to simulate DoS):** installed and setup virtual machine Ubuntu and installed the necessary software components, including Apache web server and Metasploit Framework. Configured the virtual machine to have the desired network connectivity.

2. **Install and Configure DVWA:** Setup the Damn Vulnerable Web Application (DVWA) on the Apache web server. Configured DVWA with different security levels to simulate varying levels of vulnerability.

3. **Identify Target IP and Port:** Determined the IP address and port of the target service that will be flooded with HTTP requests. This information is crucial for directing the flood attack.

4. **Developed Python Script:** developed a Python script specifically designed for conducting an HTTP flood attack. This script was capable of sending a high volume of HTTP requests to the target IP and port.

5. **Configure Attack Parameters:** Modified the Python script to customize the attack parameters. This included specifying the target IP and port, the number of concurrent connections, and the frequency of the HTTP requests.

6. **Executed the HTTP Flood Attack:** Run the modified Python script to initiate the HTTP flood attack. The script will continuously send a large number of HTTP requests to the target service, overwhelming its resources and hindering its ability to handle legitimate user requests.

It was essential to conduct this project in a controlled and responsible manner, ensuring that the DoS attack was performed solely within the designated environment and without causing harm to any external systems or networks.

Throughout the project, we examined the impact of the HTTP flood attack on the web server's performance, its ability to handle legitimate user traffic, and the effectiveness of the security measures in place. Additionally, we explored potential countermeasures and mitigation techniques that can be implemented to defend against such attacks.

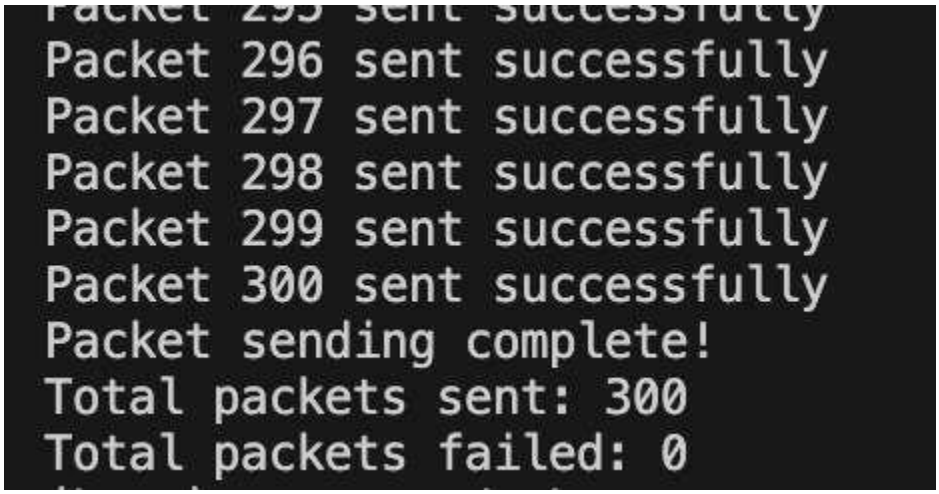

Figure 4: DoS Attack

SECTION VI: RESULTS

1. **Server Overload:** The implemented DoS attack through HTTP flood might successfully overwhelm the targeted Apache web server with a high volume of HTTP requests. This can lead to the server becoming overloaded, resulting in slow response times or even a complete denial of service for legitimate users trying to access the web application.

2. **Performance Degradation:** The excessive volume of incoming requests during the attack may cause a significant degradation in the performance of the Apache web server. This could manifest as increased latency, reduced throughput, and increased error rates, making it difficult for legitimate users to interact with the web application effectively.

3. **Resource Exhaustion:** The continuous flood of HTTP requests can exhaust system resources on the target server, such as CPU, memory, and network bandwidth. This resource exhaustion can further exacerbate the server's performance issues and hinder its ability to handle legitimate user traffic.

SECTION VII: CONCLUSION

By undertaking this project, we aimed to gain practical insights into the workings of DoS attacks, comprehended the challenges faced by network administrators, and explored potential mitigation strategies.

In conclusion, our project focused on implementing a DoS attack using the HTTP flood technique on an Apache web server. Through the utilization of the Metasploit framework on an Ubuntu virtual machine, we flooded the server with a high volume of HTTP requests, overwhelming its capacity to handle legitimate user traffic. The project allowed us to explore DoS vulnerabilities, evaluate the effectiveness of security measures, and gain insights into mitigation strategies. We successfully executed the attack, demonstrating the impact on server performance and availability. By utilizing the Metasploit framework, we assessed its capabilities and identified potential vulnerabilities. Overall, the project provided valuable knowledge in network security and emphasized the importance of robust defenses against DoS attacks.

REFERENCES

[1] Codebots: "Ubuntu 18.04 Virtual Machine Setup." Available online: [URL]

[2] Metasploit: "Metasploit Framework." Available online: [URL]

[3] VulnHub: "Damn Vulnerable Web Application (DVWA)." Available online: [URL]