Heartbleed Vulnerability Exploit and CTF Challenges

Team Name
Tel Aviv University

September 2024

Abstract

This report provides an overview of the Heartbleed vulnerability and describes how it was exploited in a controlled CTF (Capture The Flag) environment, including a complete End-to-End (E2E) attack demonstration. The E2E attack simulates a real-world scenario where a vulnerable server stores sensitive information, which is then leaked through a malicious Heartbeat request. Two challenges were developed for the CTF: a Beginner Challenge, where participants exploit the Heartbleed bug by crafting a malicious Heartbeat request, and an Intermediate Challenge, where participants secure the server to prevent memory leakage. This report covers the attack implementation, the CTF design, instructions, and the final reflections on the vulnerability's impact.

1 Introduction

The **Heartbleed vulnerability** (CVE-2014-0160) is one of the most infamous security flaws in modern history, impacting millions of web servers globally. It exposed sensitive information, such as passwords, encryption keys, and personal data, by exploiting a flaw in the popular OpenSSL cryptographic library.

To understand the Heartbleed vulnerability, several key concepts are required:

- **OpenSSL**: A widely used open-source implementation of the SSL/TLS protocols that enable secure communication over the internet.
- **SSL/TLS Handshake**: A process that establishes a secure connection between a client and server, negotiating encryption parameters and exchanging keys.
- **Heartbeat Extension**: Introduced in RFC 6520, this extension allows clients and servers to check if the connection is alive. The Heartbleed vulnerability originates from this extension.

As Bruce Schneier stated, *"On the scale of 1 to 10, this is an 11."* The Heartbleed bug immediately affected over 500,000 websites and critical systems, leading to a global security crisis.

This report outlines the steps to recreate the vulnerability in a CTF environment and explains how participants can learn to exploit and secure a vulnerable server.

2 Related Work

Several sources provide a detailed analysis of the Heartbleed vulnerability and its impact:

- **Durumeric, Z., et al. (2014)**: *The Matter of Heartbleed* provides a comprehensive analysis of the global impact of the Heartbleed vulnerability. The authors explore how quickly websites and servers responded to patch the vulnerability and assess the widespread exposure of sensitive data due to this bug. This study highlights the large-scale internet security risks associated with Heartbleed.
- **Durumeric, Z., Kasten, J. A., Adrian, D., et al. (2014)**: *Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed* focuses on the response to the vulnerability, particularly the reissuance and revocation of SSL certificates. This research discusses the challenges faced by organizations in ensuring their certificates were updated post-vulnerability, revealing the broader impact on the security ecosystem.
- **Ristic, I. (2013)**: *SSL/TLS Deployment Best Practices* outlines best practices for securely deploying SSL/TLS, emphasizing the importance of following security protocols to avoid vulnerabilities like Heartbleed. This guide serves as a valuable resource for understanding why strong SSL/TLS implementations are critical for preventing attacks.
- **RFC 6520 (2012)**: *Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension* defines the Heartbeat extension, which was designed to maintain secure communication channels without renegotiating the SSL/TLS handshake. The improper implementation of this extension in OpenSSL led to the Heartbleed vulnerability, making this RFC a key document in understanding the bug's origin.
- **D. Adrian (2014)**: *The Heartbleed Bug* is a source specifically designed to inform the public about the Heartbleed bug, detailing how it works, which versions of OpenSSL are affected, and the potential consequences of unpatched systems.

These studies provide foundational insight into the severity and technical mechanisms behind Heartbleed.

3 Attack Description

This e2e attack demonstrates the Heartbleed vulnerability by setting up a vulnerable OpenSSL server and two interacting clients.

3.1 Server Setup

The server is configured with **OpenSSL 1.0.1f** to simulate the vulnerability. The server accepts SSL/TLS connections, stores sensitive data, and processes Heartbeat requests, making it susceptible to attack.

3.2 Client Operations

Client 1: Establishes a connection, sends sensitive data, and closes the connection.
Client 2: Exploits the vulnerability by sending a malicious Heartbeat request using *heartbleed_test.py*, causing the server to leak memory (specific the first client sensitive data).

3.3 Demonstration Steps

1. Set up the server by running:

```
docker build -t openssl-heartbleed-server .
docker run --rm -it -p 443:443 openssl-heartbleed-server
```

2. Run the clients by executing:

```
docker build -t vulnerable-ssl-client .
docker run --rm vulnerable-ssl-client 172.17.0.2
```

This will demonstrate the leakage of sensitive data through the Heartbleed exploit.

4 CTF Challenges

4.1 Beginner Challenge

Participants craft a malicious Heartbeat request to exploit the vulnerability. By modifying the *HEARTBEAT_REQUEST_HEX* field in the provided Python script, they can cause the server to leak memory.

4.2 Intermediate Challenge

Participants fix the Heartbleed vulnerability in the server code. They must modify the *handle_heartbeat_request()* function to properly validate the payload length.

5 Design

The CTF project is designed using Docker to containerize both the server and client environments, ensuring isolation and reproducibility. The vulnerable server is implemented in C, running an outdated version of OpenSSL (1.0.1f), which contains the Heartbleed bug. The client, written in Python, sends specially crafted Heartbeat requests to exploit the vulnerability. Docker simplifies the deployment and setup for participants, allowing them to interact with the server and test exploits or fixes in a controlled environment.

The tools and libraries used include:

- **Docker**: To create isolated containers for the server and client, ensuring the environment is consistent across different systems.
- **OpenSSL**: The server uses OpenSSL 1.0.1f, which is vulnerable to Heartbleed, simulating real-world conditions.

- **C Programming**: The server is written in C to handle SSL/TLS connections and process Heartbeat requests.
- **Python**: The client leverages the Python `socket` library to communicate with the server and send exploit requests.

This design provides a practical, hands-on environment for participants to exploit and patch the Heartbleed vulnerability.

6 CTF Instructions

6.1 Beginner Challenge

In this challenge, participants must exploit the Heartbleed vulnerability by modifying the `HEARTBEAT_REQUEST_HEX` field in the Python script `beginner.py`. The goal is to craft a malicious Heartbeat request that causes the server to leak sensitive data from memory.

To run the challenge, modify the Heartbeat request and execute:

```
python3 beginner.py
```

If the exploit is successful, the script will print a success message along with the leaked data and the CTF flag.

6.2 Intermediate Challenge

In this challenge, participants are required to fix the Heartbleed bug in the server code. Specifically, they need to modify the `server.c` file to correctly handle the Heartbeat request in the `handle_heartbeat_request` function, ensuring that the payload length is properly validated.

Once the server code is modified and the Docker container is rebuilt, run the client script to test the fix:

```
python3 intermediate.py
```

If the server is successfully patched, the script will print a success message indicating that the vulnerability is fixed and the challenge is completed. The flag will be displayed as a reward for solving the challenge.

7 Conclusion

The Heartbleed vulnerability demonstrates how easily a seemingly minor bug in SSL/TLS implementations can lead to severe consequences for internet security. By participating in this CTF, participants learn how to both exploit and secure systems from memory leakage vulnerabilities like Heartbleed.

8 References

- D. Adrian, “The Heartbleed Bug,” Heartbleed.com, April 2014. Available: `https://heartbleed.com/`

- OpenSSL Project, “OpenSSL: Cryptography and SSL/TLS Toolkit,” OpenSSL.org, 2024. Available: `https://www.openssl.org/`
- Z. Durumeric, E. Wustrow, and J. A. Halderman, ”The Matter of Heartbleed,” in Proceedings of the 2014 Internet Measurement Conference, 2014.
- Z. Durumeric, J. A. Kasten, D. Adrian, et al., ”Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed,” in Proceedings of the 2014 ACM Internet Measurement Conference (IMC), 2014. Available: `https://conferences2.sigcomm.org/imc/2014/papers/p489.pdf`
- I. Ristic, ”SSL/TLS Deployment Best Practices,” Feisty Duck, 2013. Available: `https://www.feistyduck.com/library/openssl-cookbook/online/ch-openssl.html`
- RFC 6520, ”Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension,” Internet Engineering Task Force (IETF), 2012. Available: `https://datatracker.ietf.org/doc/html/rfc6520`