The Heartbleed Vulnerability

An unseen theft of protected information

Francesca Ponzetta

Lucas Norman Jonsson

Group 26

Table of content

1. Introduction

Many bugs and vulnerabilities have been discovered and exploited ever since the first computer system was created but few have caused as much widespread damage and cost as much as the Heartbleed vulnerability. Many servers, clients, and IoT devices with the vulnerable versions of OpenSSL were affected resulting in millions of devices in total, with big sites such as Yahoo!, Github, Stack Overflow, etc. and an estimated total cost of US$ 500 million.[1]

Analyzing vulnerabilities can help in understanding how to avoid future bugs and give insights into how small flaws can lead to severe security consequences. The goal of this report will be to take a deeper look into the mechanics of the bug and the surrounding circumstances, look into how an attack could be done, and investigate the different possible countermeasures. Section 2 of this paper describes the functionality of the different components and explains the vulnerability in detail. Section 3 showcases the different steps in performing an attack that exploits the vulnerability, while section 4 explores the different countermeasures and mitigation techniques. In section 5 the work is analysed and discussed and finally, in section 6 the report is summarized.

2. Background

Heartbleed is a vulnerability that affects the OpenSSL cryptographic library. The exploitation of this vulnerability allows an attacker to steal information from the TLS/SSL server involved in the current TLS connection. In particular, the attacker may steal not only the secret key used by the TLS protocol in order to secure the communication but also users credentials and other sensitive information stored on the server. As the TLS protocol is used to secure communications and users data, this vulnerability should be patched in order to prevent its exploitation.[2]

OpenSSL:

OpenSSL is an open-source library used to implement cryptographic functions, such as secure connection to websites via TLS (Transport Layer Security) or SSL (Secure Socket Layer). Since it was introduced, several vulnerabilities were found. Among these vulnerabilities, the most common and impactful is Heartbleed.[3]

Heartbeat Extension:

In version 1.0.1 of OpenSSL, the Heartbeat extension was introduced and with it the heartbleed vulnerability. The purpose of its introduction was to allow TLS connections between clients and servers to stay open for longer periods of time without the need for re-exchanging session parameters and establishing new sessions (which requires time and resources).
When the heartbeat extension is enabled, the client and the server exchange some “dummy packets” to check if the other system is still active in order to keep the connection alive. This is done through that one of the two end-points of the TLS connection sends the other end-point a Heartbeat-Request packet containing a payload of arbitrary data, a field that specifies the length of the payload and random padding. The other peer answers this packet with a Heartbeat-Response packet containing a copy of the payload (retrieved from memory) from the request and its own generated random padding.

The Heartbleed Vulnerability:

The main problem of the Heartbeat extension is that it allows reading data contained in the memory of one of the two peers by specifying a payload length that is larger than the actual length of the payload. In the heartbeat packet, 2 bytes are reserved for specifying the payload length. This means that packets of sizes up to 64KB (2^{16} = 64 KB) can be requested from the memory of one of the two peers. Figure 1 shows how the functionality is used normally vs maliciously.

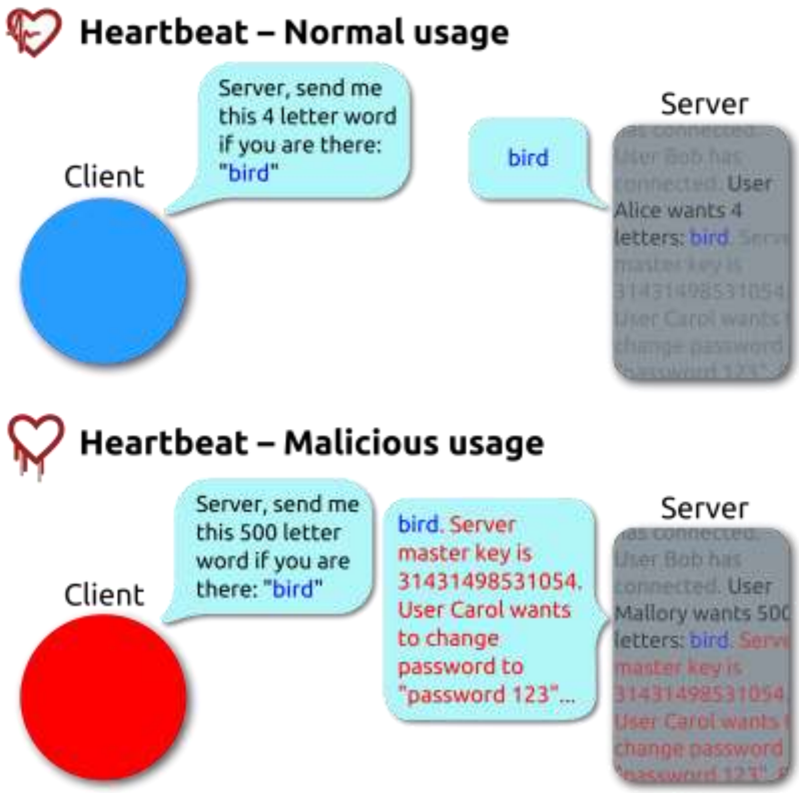

Figure 1: Simplified explanation of the Heartbeat

The vulnerability is caused by the fact that the peers assume that the declared length is equal to the actual length of the payload, without checking if it is true. So if they receive a Heartbeat request, they just execute *memcpy(bp, pl, payload),* where *bp* is the buffer where the new payload will be copied, *pl* is the pointer to the buffer to be copied (payload received in the request) and *payload* is the declared length of the payload to be copied. Actually, this function copies in *bp* a buffer that is of the same length of *payload*, so if *pl* is shorter, *pb* will be filled with bytes taken from the memory of the peer that is answering the heartbeat request.[4]

3. Exploiting the vulnerability

To be able to abuse the vulnerability, a server system and a client system needs to be set up, the target of the attack needs to be scanned for vulnerabilities and an attack against the target needs to be launched. The following will be described in the rest of this chapter.

Configuring the systems

We choose to host the target server on a virtual machine containing the operating system Ubuntu SEED 12.04 which is a prebuilt Ubuntu System with the necessary software installed.[5] On this machine, an Apache HTTP server is run with the implemented OpenSSL library version 1.0.1, which is one of the versions containing the heartbleed bug.

For the attacking client system, we choose to use another virtual machine with the operating system Kali Linux which is an open-source Debian Linux distribution focused on penetration testing and security research.[6] The necessary software that we use comes preinstalled with it, we will mainly use the Nmap (Network Mapper) software[7] which is an open-source utility used for network scans and security analysis and the open-source Metasploit software[8] used for penetration testing.

In order to give a more realistic demonstration, the attack has been performed against an open-source social network application that has been released for educational purposes. The name of the application is ELGG and it is hosted at the following URL: *https://heartbleedlabelgg.com.* On the attacker machine we have performed the following actions:

1. We have modified the */etc/hosts* file in order to map the IP address of the victim server with the URL of the web application
2. We have opened the application and we have logged in with the admin credentials (Username: *admin*, Password: *seedelgg*)
3. We have added Bob as a friend (More -> Members -> Bob -> Add Friend) and we have sent Bob a private message (Subject: ‘Test’, Body: ‘Hell0 there’) (Figure 2).

Figure 2: www.heartbleedlabelgg.com, message sent to Bob

Scanning the target:

To be able to launch an attack we first need to gather some information about the targeted server, mainly the IP address and the port number from where the server is running. There are many ways to check for the IP address, we have chosen to do it by simply issuing the command *ifconfig* from a terminal on the server system. Next, a port scan has been performed using the Nmap software which also searches for the heartbleed vulnerability during the scan. The command used is "*nmap -sV --script=ssl-heartbleed www.heartbleedlabelgg.com"* which scans all available ports on the target IP address, detects the services used on the open ports, and checks if the services are vulnerable to the heartbleed bug. The result, which can be seen in Figure 3, shows that port 443 hosting OpenSSL is vulnerable.

Figure 3: The result of the Nmap scan

Performing the attack:

The attack was performed in two different ways: first using the hacking tools provided by the Metasploit framework and then using a python script.

Attack performed with Metasploit:

Metasploit is a framework that contains different modules allowing to perform vulnerability scanning and vulnerability exploitation.[8]
After running Metasploit with the command *msfconsole*, we have searched for the heartbleed module (*command: 'search openssl_heartbleed'*) and we have set up Metasploit to use this module (*command: 'use auxiliary/scanner/ssl/openssl_heartbleed'*).

In order to perform the attack against a server, we had to supply the URL of our target server. We have set up the RHOST (Remote Host), with the URL of our target server, while the RPORT (Remote Port) was already set to 443 (no need to be changed as we know that TLS is already running on this port). (Figure 4)

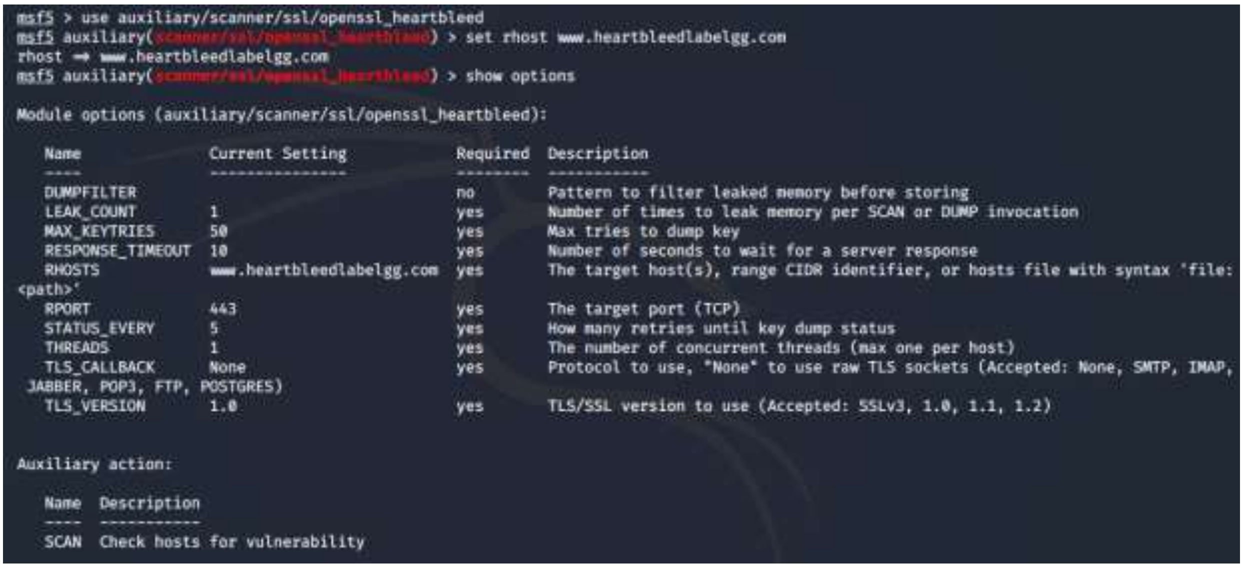

Figure 4: Set remote host

In order to know which actions can be performed by the heartbleed module, we have run the command *show info (Figure 4)*:

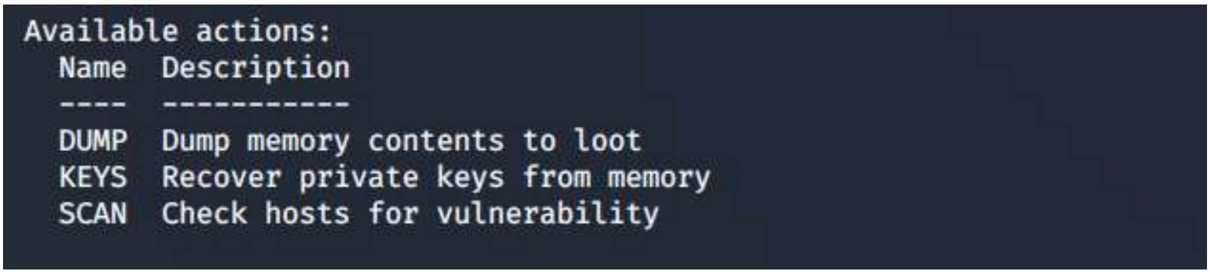

Figure 5: Show info

The action SCAN is used to check the presence of the vulnerability (in the same way as we did with Nmap), DUMP (Figure 6) is used to exploit the vulnerability and retrieve the content of the memory of the server (which will be saved into a bin file, *Figure 7,8*) and KEYS is used to retrieve the private RSA key of the server (which will be also stored in a text file) (*Figure 9*).

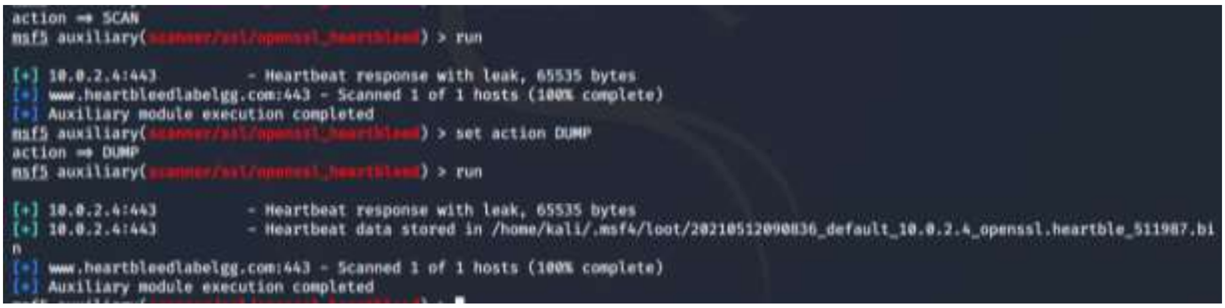

Figure 6: Set the actions SCAN and DUMP and run them.

Figure 7: Successful theft of admins credentials.

Figure 8: Successful reading of example message.

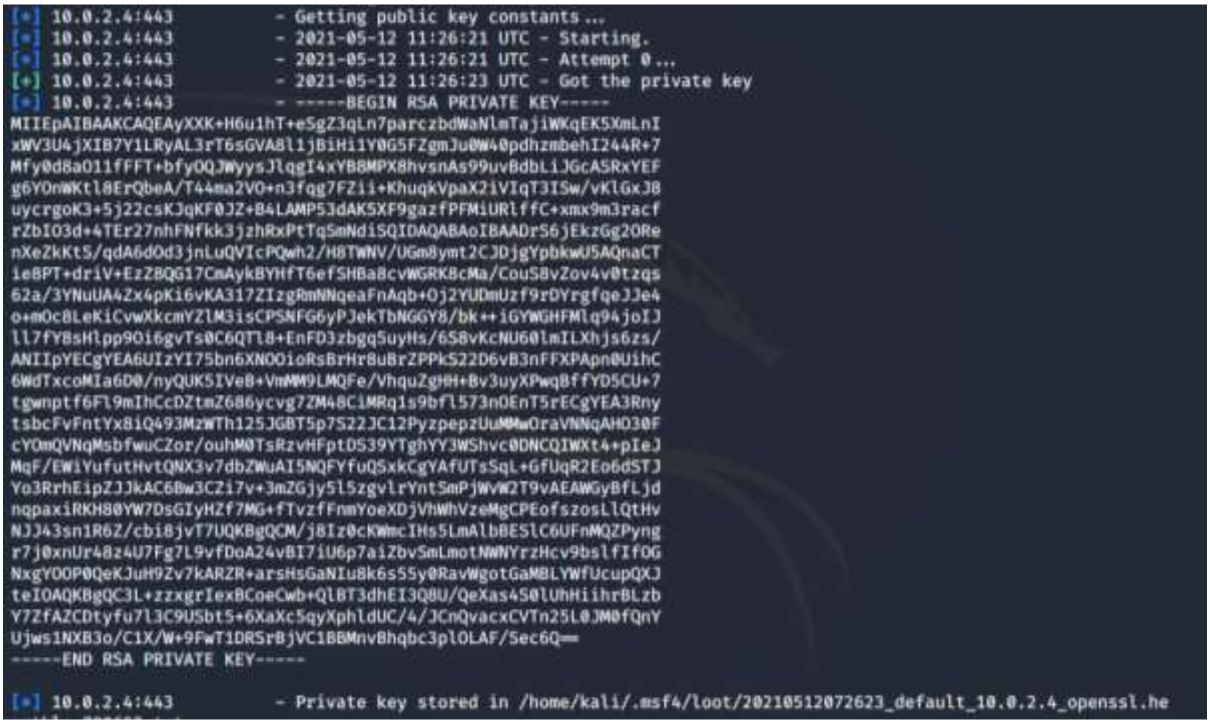

Figure 9: Set action KEYS and run, the private key of the server can be seen as output

Attack performed with a python script

The same attack can also be performed without using the tools provided by Metasploit but by writing and executing an attack python script. The source of our attack script is the following: https://github.com/roflcer/heartbleed-vuln.

The script **attack.py** contains the code that allows checking if the vulnerability is present and then to exploit the vulnerability in order to retrieve the content of the memory of the target server.

After saving the code on our client virtual machine, we changed his permissions with chmod in order to make it executable and then we executed the script with the following command: **./attack.py www.heartbleedlabelgg.com**. (Figure 10,11)

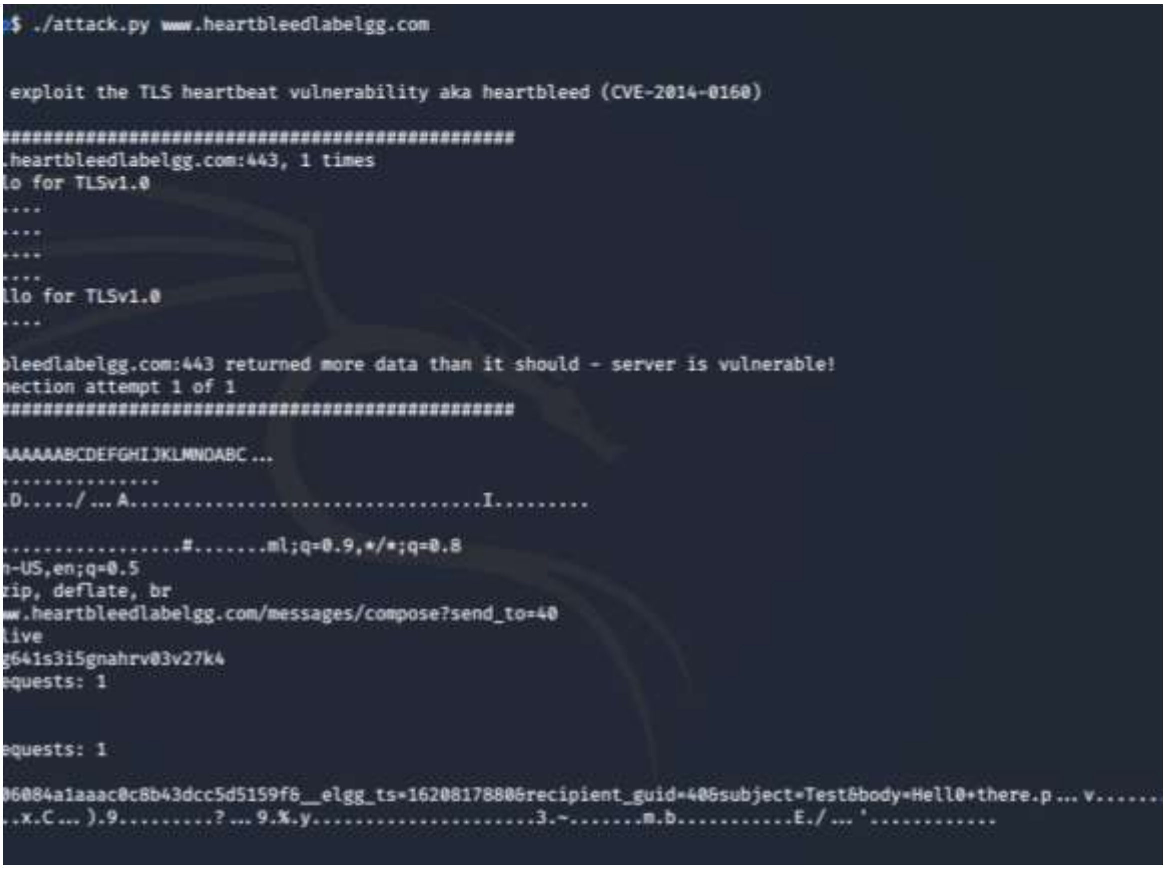

Figure 10: Message subject and body leaked from the server after running the attack script

Figure 11: Admin password leaked from the server after running the attack script

4. Countermeasures

From when the bug was publicly disclosed, it didn't take long for a fix for it to be released and a patch solving the issue was pushed out in version 1.0.1g of OpenSSL. The patch and the other recommended mitigation techniques will be described below. Other non-standard countermeasures will also be explored.

General mitigation techniques:

After the patch solving the issue was released on 7th April 2014, the main recommendation to fix the Heartbleed vulnerability was to upgrade to a version equal to or newer than 1.0.1g of OpenSSL. The patch is basically adding missing bound checks that prevent anyone from receiving more data back than they sent in the heartbeat messages.

The most important change can be seen in figure 12, here the first part of the code makes sure that the heartbeat request isn't 0 Kb in size which can cause some problems, and the second part performs a check to make sure that the size of the payload matches the included specified length. If these checks don't match, the request will be silently dropped to not give out any information to malicious actors.[9]

```
* Read type and payload length first */

if (1 + 2 + 16 > s->s3->relent)

return 0;

/* silently discard */

hbtype = *p++;

n2s(p, payload);

if (1 + 2 + payload + 16 > s->s3->rrec.length)

return 0;

/* silently discard per RFC 6520 sec. 4 */

pl = p;
```

Figure 12: The most important changes in the patch fixing the bug.

To check if a server is affected by the Heartbleed bug, a web-based tool was also developed by Pentest-Tools.com: giving an URL as an input, it checks if the server has been patched properly and if not includes a memory dump of the server.[10]

If a server under your control is discovered to be vulnerable and has potentially been vulnerable for a while, it is not enough to just patch it. As abusing the heartbleed bug can give access to the private keys of the target, it is also important to revoke and reissue the keypairs and certificates that are being used on the server, and to invalidate the compromised session keys and session cookies.

If you are a user on an affected server it is important that you change your credentials on that server and also on all other services that you use the same password on as your credentials could have been stolen and be used without your knowledge.

Explored countermeasures:

In order to prevent the attack, on the server virtual machine we have downloaded and installed version **1.1.1g** of **OpenSSL** which is not affected by the vulnerability. Trying to scan the server for the vulnerability with Nmap we have obtained the following result (Figure 13):

```
i@kali:/etc$ nmap -sV --script=ssl-heartbleed www.heartbleedlabelgg.com
arting Nmap 7.80 ( https://nmap.org ) at 2021-05-12 10:08 EDT
ap scan report for www.heartbleedlabelgg.com (10.0.2.5)
st is up (0.00088s latency).
t shown: 992 closed ports
RT     STATE SERVICE    VERSION
/tcp   open  ftp        vsftpd 2.3.5
/tcp   open  ssh        OpenSSH 5.9p1 Debian 5ubuntu1.10 (Ubuntu Linux; protocol 2.0)
/tcp   open  telnet     Linux telnetd
/tcp   open  domain     ISC BIND 9.8.1-P1
/tcp   open  http       Apache httpd 2.2.22 ((Ubuntu))
http-server-header: Apache/2.2.22 (Ubuntu)
3/tcp  open  ssl/ssl    Apache httpd (SSL-only mode)
http-server-header: Apache/2.2.22 (Ubuntu)
http-trane-info: Problem with XML parsing of /evox/about
28/tcp open  http-proxy Squid http proxy 3.1.19
http-server-header: squid/3.1.19
80/tcp open  http       Apache httpd 2.2.22 ((Ubuntu))
http-server-header: Apache/2.2.22 (Ubuntu)
rvice Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

rvice detection performed. Please report any incorrect results at https://nmap.org/submit/ .
ap done: 1 IP address (1 host up) scanned in 29.71 seconds
```

Figure 13: The Nmap scan no longer finds a vulnerability.

As we can see, compared to the previous result of Nmap in Figure 3, here in Figure 13 the server is not considered vulnerable.

We have then tried to perform the attack with Metasploit and we have obtained the result shown in figure 14. Performing the scan we can see that there's not leaked information, in fact with the damp and the keys options we don't get anything from the memory of the server.

```
kali@kali:~/Desktop$ ./attack.py www.heartbleedlabelgg.com

defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)

##################################################################
Connecting to: www.heartbleedlabelgg.com:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result....
Analyze the result....
Analyze the result....
Analyze the result....
Received Server Hello for TLSv1.0
Analyze the result....
Received alert:
Please wait... connection attempt 1 of 1
##################################################################

.F
```

Figure 14: Failed attack returning nothing after a secure version is installed.

Despite the simplest way to prevent this attack is to upgrade OpenSSL to a non-vulnerable version, there are many countermeasures that could be implemented:

1. Discard any suspicious packets, for example, the ones with unreasonably long payloads.

2. If the actual length of the payload doesn't match the declared length, the payload could be filled by copying the received payload for declared length/actual length times instead of retrieving sensitive content from the memory.

3. When a TLS connection is established, it's usually the server that is interested in keeping the connection alive (so that it doesn't need to re-exchange many times the TLS parameters with the same client). That's why it could be reasonable to think that a TLS heartbleed request sent by the client could be considered suspicious. On the contrary, it's wrong to think that a heartbeat request sent by a server cannot be malicious.
   If the security software BIG-IP is used, it is possible to specify some iRules[1] (for both client-side and server-side) that allow to protect from the attack:
   - **Client-side Heartbleed iRule**: the connection is immediately closed when a heartbeat request from the client is detected.
   - **Server-side Heartbleed iRule**: it looks at the Heartbleed requests/responses sent by the server: if it is longer than 128 bytes, it is considered malicious and the connection is rejected.

4. If it is not possible to upgrade to a patched version of OpenSSL, the best solution would be to **disable the Heartbeat extension** by recompiling OpenSSL with the option -DOPENSSL_NO_HEARTBEATS.

[1] BIG-IP is a family of software and hardware products designed for application availability, access control and security solutions.[11] An iRule is a script that allows to make use of some of the extended capabilities of BIG-IP and it allows to interact more directly with the traffic passing through the device.[12]

5. Discussion

The Heartbleed bug was a small coding mistake with great consequences, in our opinion one of the reasons why it led to these severe issues was that the bug happened to be in a trusted software used by many services for the purpose of handling or improving the security. As OpenSSL handles all the vital information, if something goes wrong, like in this case, the system will be completely vulnerable to anyone with malicious intent.

Despite that OpenSSL is an open-source library and everyone can look at the code, the Heartbleed vulnerability wasn't discovered until two years after the introduction of the heartbeat extension. This happened mainly because users usually tend to trust the software they are using and because the bug was actually a small mistake that could be hard to notice. The security problems created by this bug could have been avoided if both the programmers of the library and the users had looked at the code more carefully.

Since the bug was a minor mistake, a clear and simple solution to it was implemented very shortly after the bug was discovered. The problem itself wasn't very complex, and the best fix to it was pretty straightforward. The explored countermeasures in this report are all acceptable solutions from what we believe but none of them can be seen as equally good or better than the official patch.

During this project, we have chosen to utilize pre-built virtual machines for both the attacking client and the target server with all the necessary software preinstalled. Another option would have been to set up and configure the systems ourselves, but as the focus lay on the heartbleed bug itself we felt that it was better to focus directly on the implementation of the attack instead of spending time in the set up of the systems.
For performing the attack we have also chosen to use existing software and existing code, another option would have been to write the attack code from scratch. Reading the attack code that we have found allowed us to have a better insight into the topic, but probably if we had implemented our own code we would have had a more in-depth knowledge. Considering our scope we decided to use the already existing software and script, limiting us in exploring the topic more broadly. This could be a starting point for a future re-implementation of the attack where it could be interesting to implement our own malicious script.

Concerning the explored countermeasures, we didn't have the time to implement and test all of them, for this reason, some of them were only investigated in theory. Given more time this is probably what we would have focused on as we feel like this would have added more value to our work and more insight into how bugs and vulnerabilities can be effectively solved.

6. Conclusion

In conclusion, we can say that the heartbleed vulnerability was to be considered severe for 2 main reasons: the number of affected users and devices was large since OpenSSL is a really popular cryptographic library and that it stayed undetected for a long time giving the chance to attack the users without leaving traces.

The main goal of this report was to describe the vulnerability and to show how a small bug in the code (such as the lack of an if statement) can cause severe damage to many users. In order to prove the danger caused by heartbleed, we have implemented attacks in an ethical way, showing how much sensitive information can easily be leaked from the memory of the victim server.

Information security is a concept born in the last years when, thanks to the development of new technological devices such as computers, mobile phones and IoT, we moved from “on paper” data to digital data. The quantity of sensitive data held by the devices that we frequently use is huge and it is important to implement proper mechanisms in order to secure them. The implementation of information security is something that has to be taken into consideration from the beginning of the development of whatever software or hardware system.

References:

[1] Kerner S. *Heartbleed SSL Flaw's True Cost Will Take Time to Tally*. 2014 April 19. Available from: https://www.eweek.com/security/heartbleed-ssl-flaw-s-true-cost-will-take-time-to-tally/

[2] *The Heartbleed Bug*. [updated 2020 June 03]. Available from: https://heartbleed.com/

[3] *OpenSSL Cryptography and SSL/TLS Toolkit. c1999-2018.* Available from: https://www.openssl.org/

[4] Thakkar A. *Heartbleed: A Formal Methods Perspective*. 2020. Available from: https://aalok-thakkar.github.io/webpage_files/publications/CIS673_report.pdf

[5] Du W. *SEED Labs.* Available from: https://seedsecuritylabs.org/lab_env.html

[6] g0tmi1k. *What is Kali Linux?*. [updated 2021 April 22] Available from: https://www.kali.org/docs/introduction/what-is-kali-linux/

[7] *Nmap: The Network Mapper - Free Security Scanner.* Available from: https://nmap.org/

[8] *Metasploit | Penetration Testing Software*. Available from: https://www.metasploit.com/

[9] Fruhlinger J. *What is the Heartbleed bug, how does it work and how was it fixed?*. 2017 September 13. Available from: https://www.csoonline.com/article/3223203/what-is-the-heartbleed-bug-how-does-it-work-and-how-was-it-fixed.html

[10] *OpenSSL Heartbleed vulnerability scanner*. Available from: https://pentest-tools.com/network-vulnerability-scanning/openssl-heartbleed-scanner?run

[11] Abbott C. *What is Big-IP?*. 2017 January 19. Available from: https://devcentral.f5.com/s/articles/what-is-big-ip-24596

[12] Holmes D. *What is an iRule?.* c2021. Available from: https://www.devcentral.f5.com/s/articles/heartbleed-network-scanning-irule-countermeasures?page=1

Appendix

A:

The contributions to this project have been equal. The practical work has been performed mostly together and when it has been done separately both members have taken turns investigating different attacking methods and implementing countermeasures. For the report, Francesca has done the majority of the work on the *Background* chapter, and the *Conclusion* chapter while Lucas has done the majority of the *Introduction* chapter and the *Discussion* chapter. For the *Exploiting the vulnerability* chapter and the *Countermeasures* chapter both members have contributed equally, all chapters have later been modified and worked on by both members.