OpenSSL Vulnerabilities: the Heartbleed Bug and Cupid

Andrei-Sorin JERCA
The Special Telecommunications Service, Bucharest
ROMANIA
andrei.jerca@gmail.com
http://www.asj.ro

Abstract: In this paper we expose the importance of information security in today's context of fast technological evolution. Our main objective is to study one of the most recent security issue, discovered in the well-known OpenSSL cryptographic software library, named "the Heartbleed Bug". This is a serious vulnerability of the Heartbeat Extension for the transport layer security protocols (TLS/DTLS) implementation in OpenSSL. This weakness allows remote attackers to obtain sensitive information (secret keys used for X.509 certificates, user credentials, instant messages, emails and other critical documents and communication) for applications such as web, email, instant messaging and some virtual private networks, which should be protected, under normal conditions, by the SSL/TLS encryption. Further, "Cupid", shows that the bug can be used, with the same effect, against any device which implies Extensible Authentication Protocol (EAP) authentication mechanisms and a vulnerable version of OpenSSL. In addition we present guidelines, fixes and methods for preventing and managing possible attacks against vulnerable systems.

Key-Words: Buffer Overflow, OpenSSL, Heartbeat Extension, EAP, The Heartbleed Bug, Cupid, Ethical Hacking Exercise

1. Introduction

Nowadays, when technology is evolving very fast and we hear more often terms like "hacker" or "cyber-attack", the information security has become an important element in our life, from home users, organizations or large companies up to state government. All are facing with many problems caused by security flaws.

If originally to "hack" meant to have extraordinary computer skills to extend the limits of a system or network, today there are automated tools and codes available on the Internet that make it possible to anyone, with a will, desire and a goal, to hack and succeed [1]. Ethical hacking simulates the behavior used by attackers, involving the use of hacking tools, tricks and techniques to identify vulnerabilities and highlights remedial actions in the security system. Most people do not understand how important is to take the right security measures at the right time.

Information security is based on three major components: confidentiality, integrity and availability. If one of these components is vulnerable, an attacker can exploit and hack the system.

Our main objective is to study and propose suitable guidelines and methods for preventing and managing an attack that is targeting the confidentiality component of the information security. For achieving it, we use an ethical hacking exercise to demonstrate how easy is to steal the secret keys used for X.509 certificates, user names and passwords, instant messages, emails and other critical documents and communication, even if you use encryption.

In the current paper we focused our study on the flaw that affects the confidentiality component of the OpenSSL cryptographic software library. The vulnerability, named "the Heartbleed Bug" (officially referenced as CVE-2014-0160 [3]), was discovered independently by a security company called Codenomicon and a Google Security researcher named Neel Mehta, in the OpenSSL's implementation of the TLS/DTLS (transport layer security protocols) Heartbeat Extension (RFC6520).

Based on the same attack as Heartbleed, Luis Grangeia, from SysValue, presents a new threat called "Cupid" which implies the authentication framework called EAP (Extensible Authentication Protocol) used in Wireless networks or wired networks

ISSN 2067 – 4074

that use 802.1x Network Authentication and peer to peer connections [12].
Moreover a system can be hijacked without using any privileged information or credentials and can't be detected because it leaves no traces of anything abnormal happening to the logs.
A brief introduction to Heartbeat Extension is provided in Section 2. The main aspects of EAP framework are revealed in Section 3. The proposed ethical hacking exercise involving "the Heartbleed Bug" and "Cupid" is presented in Section 4. Some fixes, guidelines and methods for protecting a system are presented in Section 5. The conclusions are drawn in Section 6.

2. Heartbeat Extension (RFC6520)

The Heartbeat Extension for the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols was documented and proposed as a standard in early 2012 by IETF, under direction of R. Seggelmann, M. Tuexen and M. Williams.
The Heartbeat Extension provides a new protocol for TLS/DTLS allowing the usage of keep-alive functionality without performing a renegotiation and a basis for path MTU (PMTU) discovery for DTLS. Basically the session is kept open even when any official data is not exchanged over the encrypted connection.

2.1 Overview

The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications, but there is not necessarily a feature available to keep the connection alive without continuous data transfer. The DTLS is basically designed to secure traffic running on top of unreliable transport protocols.
Usually, such protocols have no session management. The only mechanism available at the DTLS layer to figure out if a peer is still alive is a costly renegotiation, particularly when the application uses unidirectional traffic [3]. Furthermore, DTLS needs to perform path MTU (PMTU) discovery but has no specific message type to realize it without affecting the transfer of user [5].
The Heartbeat Extension is designed to overcome these limitations.

2.2 The Protocol

The Heartbeat protocol is a new protocol running on top of the Record Layer. The protocol itself consists of two message types:

- HeartbeatRequest;
- HeartbeatResponse.

These messages consist of their type, payload_length, an arbitrary payload and padding as seen in Figure 1.

```
struct
{
  HeartbeatMessageType type;
  uint16 payload_length;
  opaque payload[HeartbeatMessage.payload_length];
  opaque padding[padding_length];
} HeartbeatMessage;
```

Figure 1. Hartbeat message

The user can use the HeartbeatRequest message almost at any time during the lifetime of a connection. The peer should answer immediately with a corresponding HeartbeatResponse message carrying an exact copy of the arbitrary payload contained by HeartbeatRequest. According to [3] the maximum size of the message is restricted to 2^14 (16Kbytes) or max_fragment_length. If the payload_length of a received heartbeat message is too large or if a HeartbeatResponse does not contain the expected payload the message must be discarded silently.

3. Extensible Authentication Protocol

Extensible Authentication Protocol, or EAP, is an authentication framework frequently used in wireless networks and point-to-point connections [14]. EAP only defines message formats (it is not a wire protocol) and each protocol that uses it defines a way to encapsulate EAP

messages within that protocol's messages.
EAP is widely used and it provides some negotiation of authentication functions called EAP methods. Five of them are the official authentication mechanisms for the WPA and WPA2 standards.
In our paper we discuss only about those methods which use TLS over EAP to secure some part of the authentication process [13]: EAP-TLS, EAP-TTLS and EAP-PEAP.
EAP-Transport Layer Security (EAP-TLS) is an IETF open standard that uses the Transport Layer Security (TLS) protocol [14] and is considered one of the most secure EAP standards available. The majority of implementations of EAP-TLS requires client-side X.509 certificates which increase the security level and assures a two factor authentication mechanism.
EAP-Tunneled Transport Layer Security (EAP-TTLS) is an EAP protocol that extends TLS [14]. The server is securely authenticated to the client and then the server can use the established secure connection to authenticate the client. In this protocol the username used for authentication is never transmitted unencrypted. The secure tunnel provides protection from eavesdropping and man-in-the-middle attack.
EAP-Protected Extensible Authentication Protocol (EAP-PEAP) encapsulates EAP within a secure TLS tunnel, providing facilities for protection of the EAP conversation. The purpose was to correct deficiencies in EAP.

4. OpenSSL Heartbeat Extension Exploit

"The Heartbleed Bug" is a critical vulnerability discovered in OpenSSL's implementation of the TLS Heartbeat Extension. When it is exploited it leads to the leak of memory contents between server and client and vice versa.

4.1 How it works?

The Heartbleed vulnerability originates in the HeartbeatResponse messages, which suppose to contain a copy of the arbitrary payload from the request. An attacker sends a HeartbeatRequest message with a small sized arbitrary payload and a high payload_length which is set to the maximum possible size 0xFFFF (65535 bytes).
The broken OpenSSL source code exposed in Figure 2, point out that the HeartbeatResponse is constructed as follows: the code writes the response type to the start of the buffer, increments the buffer pointer, write the 16-bit payload_length to memory, increment the buffer pointer by two bytes and then it copies the received arbitrary payload into the outgoing payload for the reply [8].

```
/* Allocate memory for the response, size is 1 bytes
 * 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);
memcpy(bp, pl, payload);
```

Figure 2. Payload

In order to exemplify, we can send a HeartbeatRequest message having the payload_length field set to 0xFFFF (65535 bytes) which is controlled by the attacker and an arbitrary payload with one byte. The code has to send back a copy of the incoming heartbeat message, so it allocates a buffer big enough to hold the 64KBytes payload plus one byte to store the message type, two bytes to store the payload length, and some padding bytes. So even if the buffer allocated size is bigger than the received arbitrary payload, the memcpy() function will read beyond the end of the data received. That means OpenSSL runs off the end of your data and collect whatever else is next to it in memory at the other end of the connection, for a potential data leakage of approximately 64KB each time you send a malformed HeartbeatRequest [9].

4.2 Ethical Hacking Exercise

Now that we have studied the vulnerability, we can perform the ethical hacking exercise. Because this attack is ethical,

ISSN 2067 – 4074

we attacked ourselves from outside. Our target was a simple website with a basic login form protected by the SSL/TLS encryption (Figure 3). The web application was hosted on an Apache Server v2.4.7 with mod_ssl installed and OpenSSL version 1.0.1.e.

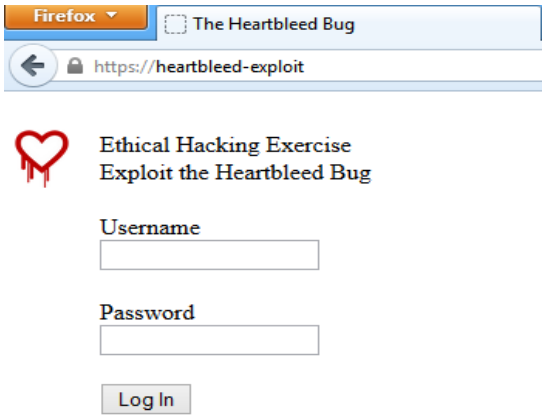

Figure 3. Test web application

On the attacker side we used a machine with Kali Linux 1.0.5 x64 distribution installed and an exploit script provided by [10] adapted for our needs. The exploit must be run against a target which is linked to a vulnerable OpenSSL (versions 1.0.1 through 1.0.1f are vulnerable).

In brief, this exploit uses OpenSSL to create an encrypted connection and trigger the heartbleed leak. The leaked information is returned within encrypted SSL packets and is then decrypted and wrote to a file. The exploit can set heartbeat payload_length arbitrarily and leaks up to 65532 bytes of remote memory for each request. The exploit can be run in a loop until the connected peer ends connection [10].

Further, the exploit can be used for plain text communication protocols which use STARTTLS for upgrade a plain text connection to an encrypted one such as SMTP, IMAP or POP3. During the exercise we had to reduce the payload_length requested size because the client often forcefully closed the connection during large leak requests.

First, the target must have one SSL/TLS port open to be able to initialize the encrypted connection. We scan our target for open ports using Nmap tool (Figure 4).

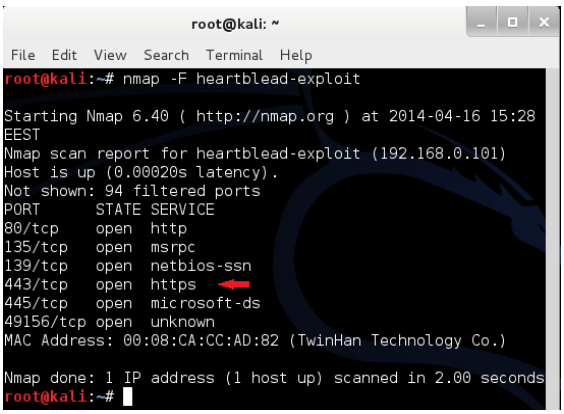

Figure 4. Nmap output

Next, we run the heartbleed exploit script sending as input the target IP address, the SSL/TLS port, the padding_lenght and the output file to write data to.

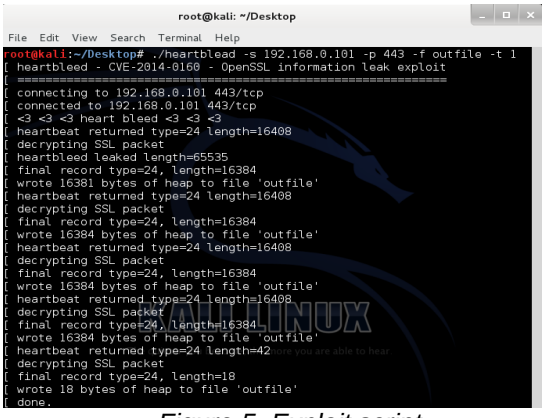

Figure 5. Exploit script

The result can be seen in Figure 6, where it is exposed the content of the output file generated by the script. We clearly see that we were able to read private data and other critical information from target memory, even if the website uses encryption and should be protected.

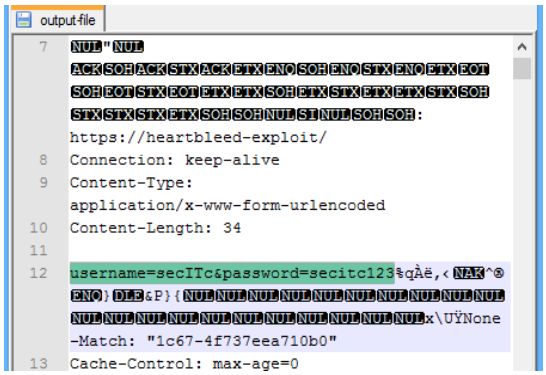

Figure 6. Atack results

During our tests we were able to obtain user names, passwords, instant messages, secret keys used for X.509 certificates, critical documents and many other sensitive data. Another important observation is that our attack had not leaved any traces in network traffic or anything abnormal happened to the logs.

4.3 "Cupid"

"Cupid", is based on a malicious heartbeat packet which is transmitted on regular TLS connections over TCP. Both clients and servers can be exploited and memory can be read off processes on both ends of the connection [12]. The difference between "The Heartbleed Bug" scenario and this one is that the TLS connection is being made over EAP.

The scenario, presented in [12] uses the patched versions of two Linux based applications to exploit the heartbleed flaw on TLS connections:

1. "hostapd" is a user space daemon for wireless access point and authentication servers [15] used on Linux, which is capable to create almost any kind of Wireless Network configuration and let clients connect to it.
2. "wpa_supplicant" is a free software implementation of IEEE 802.11i (applied as WPA2) standard used to connect to wireless networks on Linux (including Android mobile operating system).

First one it is used to exploit vulnerable clients by setting up a fake network. When a client tries to connect and the TLS connection is initialized, the "hostapd" will send malicious heartbeat requests, triggering the heartbleed vulnerability. When the client receives the request, it automatically sends back a response which contains data from target memory.

For the second application, the target is a vulnerable server (network). So when we request a new connection and send a heartbeat request right after the TLS connection is made, the attack starts.

Considering the EAP authentication mechanism, an attacker will only need a valid username to perform a connection attempt and send malformed heartbeat requests. Another important aspect is that a full TLS connection is not required, because the heartbeat request can be sent and received before keys and certificates are exchanged.

Using a network traffic analyzer tool we can find packets with the heartbeat behavior and capture the 64KB heartbeat response leaked from memory (Figure 7).

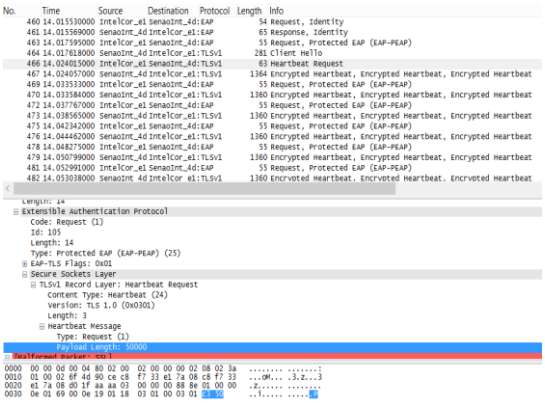

Figure 7. heartbeat response leaked from memory

The results suggest that you could be able to obtain the private keys used for TLS connection and the credentials used for connection authentication [12].

This security problem affects some version of Linux based operating systems which have vulnerable versions of OpenSSL (including Ubuntu and Android) and some wireless solutions (especially those used by corporations or large organizations) which uses EAP authentication mechanisms.

Unlike heartbleed, which is best known for giving end users the ability to collect data out of vulnerable servers, "Cupid", shows that the bug can be used, with the same effect, against any device running a vulnerable version of OpenSSL.

5. Fixes and Guidelines

First off all we must highlight that this vulnerability in not a design flaw in SSL/TLS protocol specification. This is a programming mistake in OpenSSL library.

5.1 Fixes

The bug was introduced to OpenSSL in December 2011 and has been out in the

ISSN 2067 – 4074

wild since OpenSSL release 1.0.1 on 14th of March 2012 [2]. The vulnerable versions of the OpenSSL affected are 1.0.1 through 1.0.1f (inclusive).
For fixing this bug you should upgrade to OpenSSL version 1.0.1g or newer. If this is not possible software developers can alternatively recompile OpenSSL with the option -DOPENSSL_NO_HEARTBEATS [4].
Besides, any protection given by the encryption and the signatures in the X.509 certificates can be bypassed if the secret keys were leaked. This allows the attacker to decrypt any past and future traffic to the protected services and to impersonate the service at will [2]. The owners of services must revoke the compromised keys and reissues and redistributes new keys. After this operation is complete, regular users should start changing their credentials (especially passwords) and any possible compromised encryption keys. All session keys and session cookies should be invalidated and considered compromised [2].

5.2 Guidelines

If we look at the bright side of this security flaw, it was a good opportunity for all service providers to upgrade their security strength of the secret keys used and to increase the security measurements.
The vulnerability presented in this paper is based on a simple programming error, and the chance that another similar flaw occurs in the future should not be ruled out. Although there can be no universal security measures to combat any possible security incident, a few basic rules for information security can be followed.
Taking into account that "the Heartbleed Bug" is actually a buffer overflow problem we present some protection measures to defend against it. A programmer should always design a program with security in mind, test and debug the code for errors, do the manual auditing of code, prevent use of dangerous functions like memcpy, malloc, strcpy etc. and prevent all sensitive information from being overwritten [1].
One of the main reasons of security breaches is the human factor mistake. The proper implementation of a strong security policy is the most important measure that must be taken to reduce the risk of security flaws.

6 Conclusions

Considering our tests, the long period in which it remains undetected, ease of exploitation and the attacks leaving no trace we consider that this vulnerability is severe. A lot of users can be affected either directly or indirectly because OpenSSL is the most popular open source cryptographic library used to assure privacy for you and your transactions. Many websites, in various fields, social, commerce, banking, hobby, even sites run by your government and many online services were affected by this vulnerability.
The main contribution of this paper is that it exposed how important is the information security in the actual context of a world ruled by and via Internet. We illustrate this idea, using an ethical hacking exercise, which shows how a basic programming error can generate a security flow that is believed to have affected more than half a million web applications.
The information security is an interesting area. Our approach should encourage the programmers, to give more importance to application security and they should always remember that buffer overflows errors produce serious data leakages.

Acknowledgement

Parts of this research have been published in the Proceedings of the 7th International Conference on Security for Information Technology and Communications, SECITC 2014.

References

[1] EC-Council, Ethical Hacking and Countermeasures CEH v8, 2013
[2] Codenomicon Ltd., The Heartbleed Bug, www.heartbleed.com, April 2014
[3] CVE-2014-0160, cve.mitre.org, 2014
[4] OpenSSL Security Advisory, www.openssl.org/news/secadv_20140407.txt,April 2014
[5] Seggelmann R., Tuexen M., Williams M., RFC6520 - Transport Layer Security (TLS) and Datagram Transport Layer Security

(DTLS) Heartbeat Extension, IETF, tools.ietf.org/html/rfc6520**,** February 2012
[6] Dierks T., Rascorla E., RFC5246 – The Transport Layer Security (TLS) Protocol Version 1.2, IETF, tools.ietf.org/html/rfc5246, August 2008
[7] Rescorla E., Modadugu N., RFC6347 – Datagram Transport Layer Security Version 1.2, IETF, tools.ietf.org/html/rfc6347, January 2012
[8] Williams C., Anatomy of OpenSSL's Heartbleed: Just for bytes trigger horror bug, www.theregister.co.uk, April 2014
[9] Ducklin P., Anatomy of a data leakage bug – the OpenSSL "heartbleed" buffer overflow, nakedsecurity.sophos.com, April 2014
[10] Heartbleed OpenSSL – Information Leak Exploit, www.exploit-db.com/exploits/32791/, April 2014
[11] Stevens D., Heartbleed Packet Capture and Heartbleed Packet Capture – Full TLS, blog.didierstevens.com, April 2014
[12] Grangeia L., Heartbleed, Cupid and Wireless, www.sysvalue.com, May 2014
[13] Grangeia L., Heartbleed && Wireless, www.slideshare.net/lgrangeia, May 2014
[14] Wikipedia, Extensible Authentication Protocol, en.wikipedia.org, 2014
[15] Wikipedia, hostapd, en.wikipedia.org, 2014
[16] Wikipedia, wpa_supplicant, en.wikipedia.org, 2014
[17] Goodin D., Meet "Cupid," the Heartbleed attack that spawns "evil" Wi-Fi networks, astechnica.com, June 2014