

# CT417 SOFTWARE ENGINEERING III

## BUFFER OVERFLOW CASE STUDY – THE HEARTBLEED BUG

Dr. Michael Schukat

# A Bug with its own Website (heartbleed.com) and Icon

2

## The Heartbleed Bug

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.



### What leaks in practice?

We have tested some of our own services from attacker's perspective. We attacked ourselves from outside, without leaving a trace. Without using any privileged information or credentials we were able to steal from ourselves the secret keys used for our X.509 certificates, user names and passwords, instant messages, emails and business critical documents and communication.

### How to stop the leak?

As long as the vulnerable version of OpenSSL is in use it can be abused. [Fixed OpenSSL](#) has been released and now it has to be deployed. Operating system vendors and distribution, appliance vendors, independent software vendors have to adopt the fix and notify their users. Service providers and users have to install the fix as it becomes available for the operating systems, networked appliances and software they use.

X X X

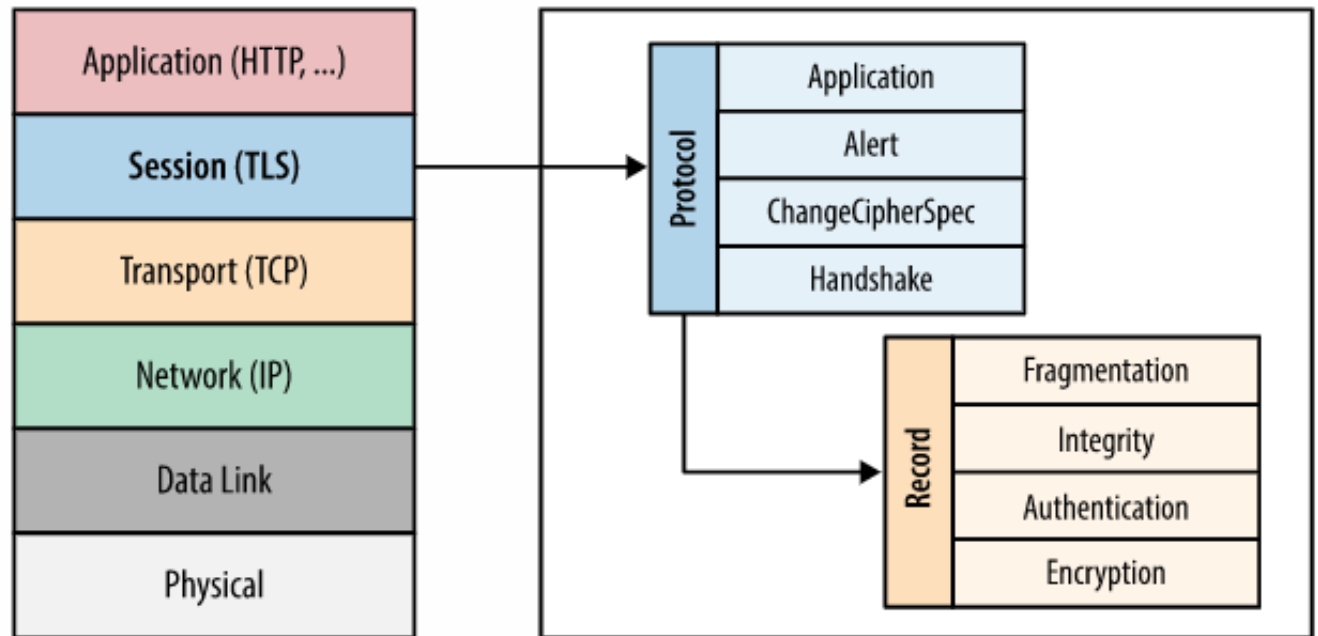
# TLS Overview

- Based on the SSL protocol, which was originally developed in the 1990s to secure ecommerce transaction on the web, i.e.
  - ▣ encryption to protect customers' personal data
  - ▣ authentication and integrity check of transactions
- To achieve this, the SSL protocol was implemented at the application layer, directly on top of TCP, enabling (application layer) protocols above it (e.g. HTTP) to operate unchanged

# TLS Overview

4

TLS location in protocol stack



# Encryption, Authentication and Integrity

- The TLS protocol provides three essential services to all application layer protocols running above it
- Encryption
  - ▣ A mechanism to obfuscate what is sent from one host to another (typically between a client and a server)
- Authentication
  - ▣ A mechanism to verify the validity of provided identification material (i.e. (mutual) authentication using digital certificates)
- Integrity
  - ▣ A mechanism to detect message tampering and forgery (messages cannot be manipulated in transit, and messages cannot be forged by a threat actor)

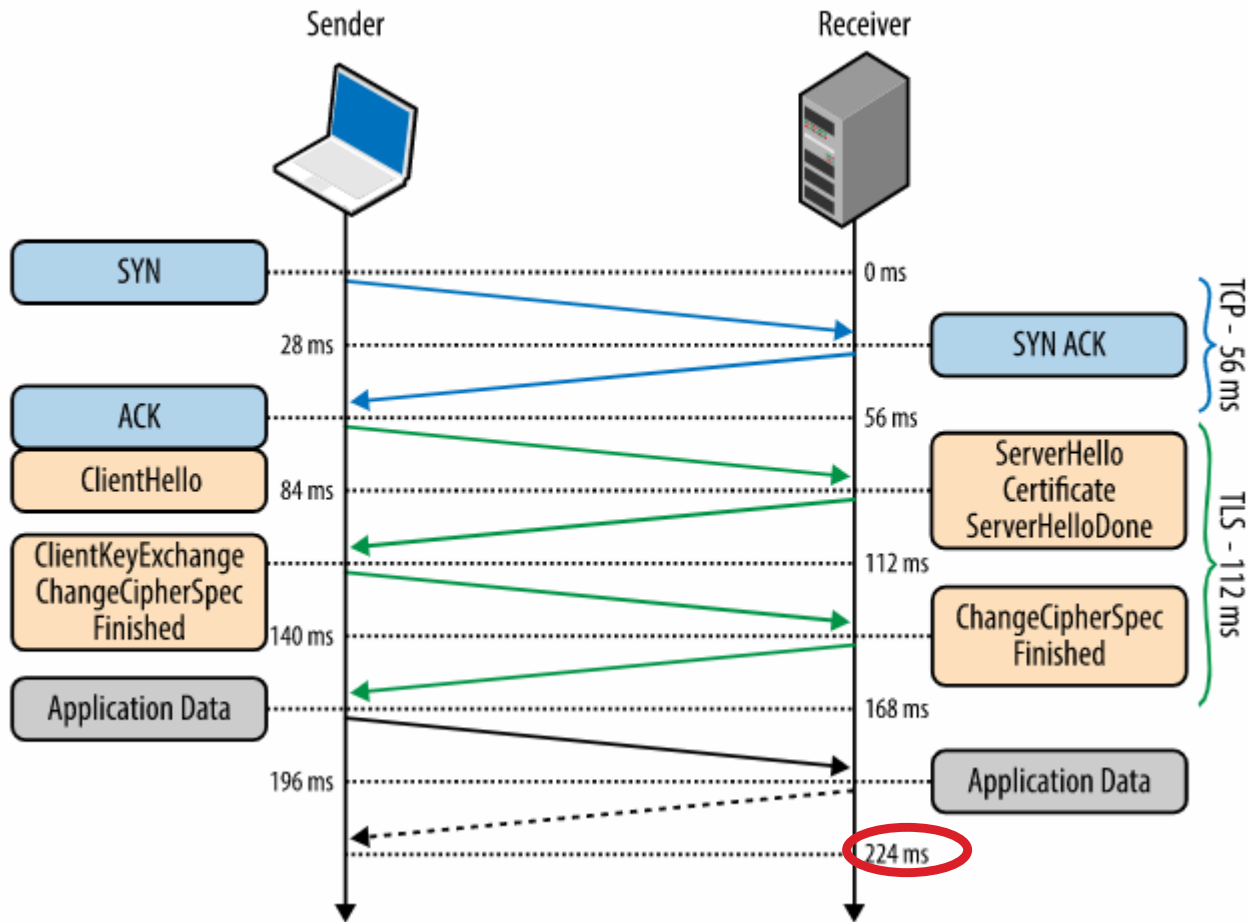
# HTTPS

6

- “HTTP over TLS”
- HTTPS protects the integrity of the website
  - ▣ Encryption prevents intruders from tampering with transmitted data
- HTTPS protects the privacy and security of the user
  - ▣ Encryption prevents intruders from eavesdropping and abusing the exchanged data
- HTTPS enables new features on the web
  - ▣ Necessary to safely use new web platform features, such as accessing users geolocation, VoIP and videoconferencing

# TLS Handshake

7



# Overview Heartbleed

8

- ❑ Discovered in 2014
- ❑ Exploits a bug in the OpenSSL implementation of the TLS “heartbeat hello” extension
- ❑ Can affect both client and server side



X X



# OpenSSL

- OpenSSL is an open-source library (→ GitHub) that contains routines / algorithms / protocol implementations / ciphers used for secure network communication
  - ▣ Including SSL (deprecated) and TLS implementations
- It is written in C and widely used in Linux distributions
  - ▣ Linux is a widely used server-side OS

# Heartbleed Impact

10

- Reported via CVE-2014-0160 (later)
- The following operating system distributions were potentially affected:
  - ▣ Debian Wheezy (stable)
  - ▣ Ubuntu 12.04.4 LTS
  - ▣ CentOS 6.5
  - ▣ Fedora 18
  - ▣ OpenBSD 5.3
  - ▣ FreeBSD 10.0
  - ▣ NetBSD 5.0.2
  - ▣ OpenSUSE 12.2

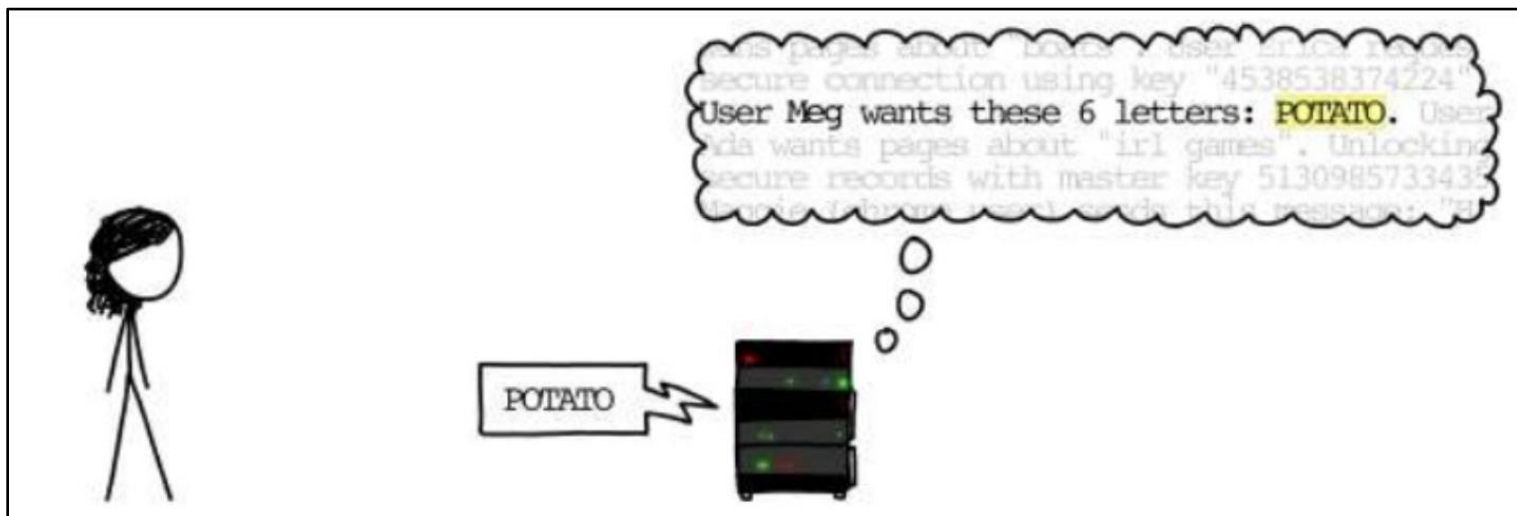
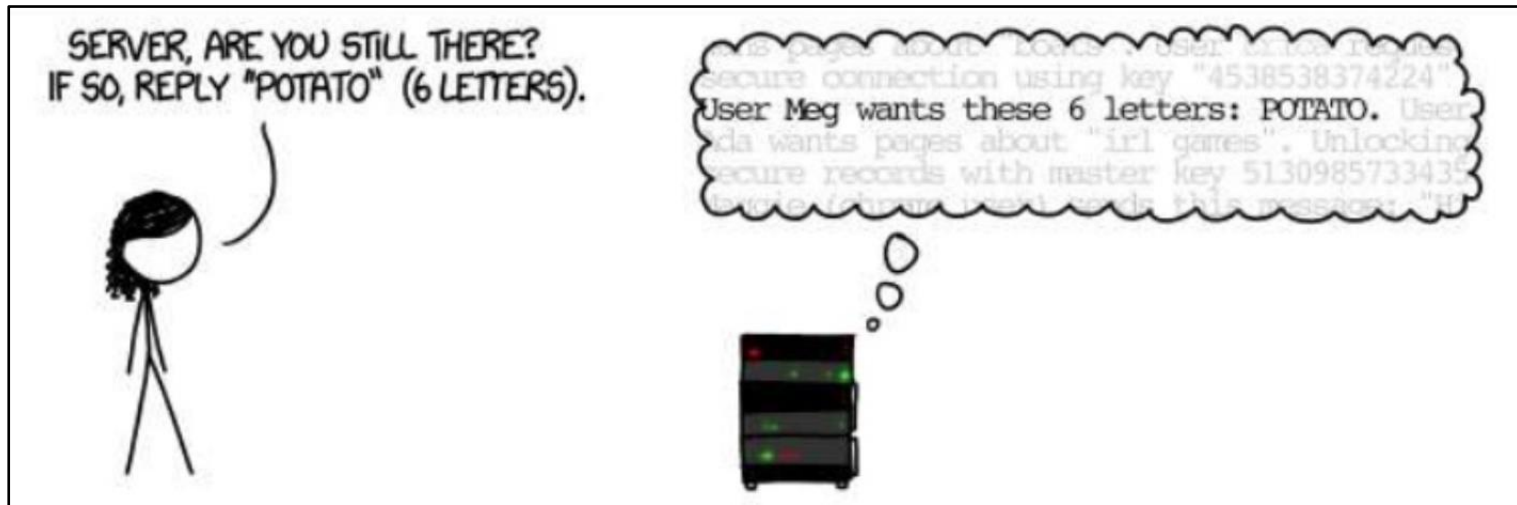
# TLS Heartbeat Extension

11

- Originally TLS had no provisions to keep a client / server connection alive without continuous data transfer
  - ▣ Idle connections would timeout instead and a computationally expensive reconnect would have to take place (224 ms in example)
- The heartbeat extension provides a new protocol for “keep-alive” messages
  - ▣ One endpoint could send out a *HeartbeatRequest* message, which would be immediately responded with a *HeartbeatResponse* message

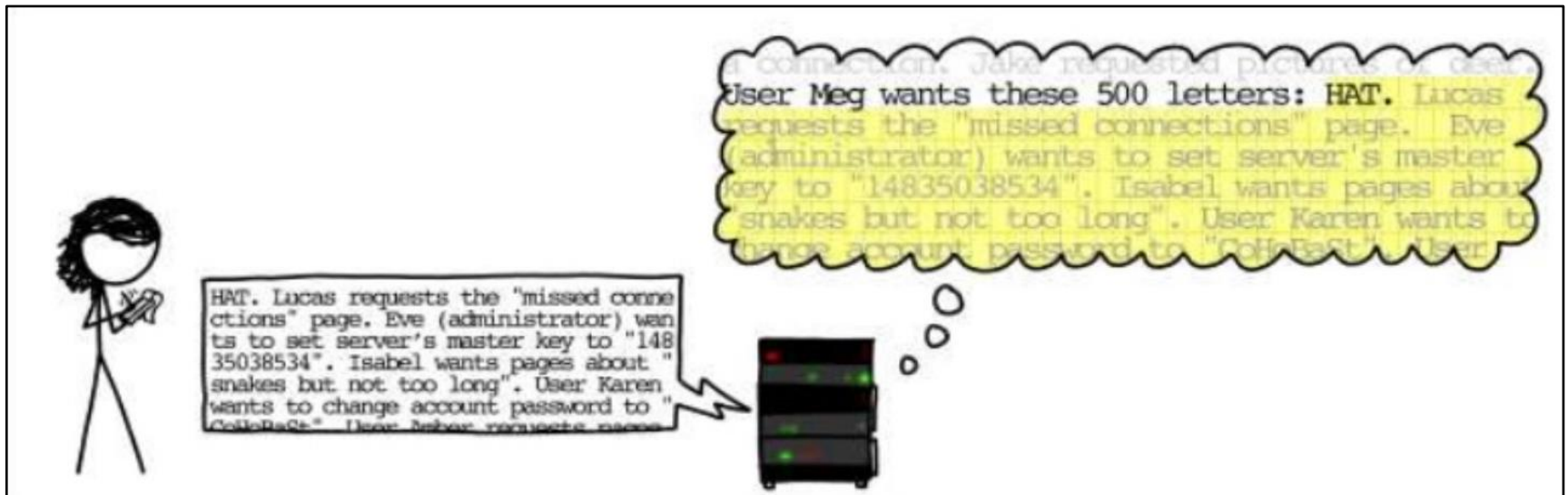
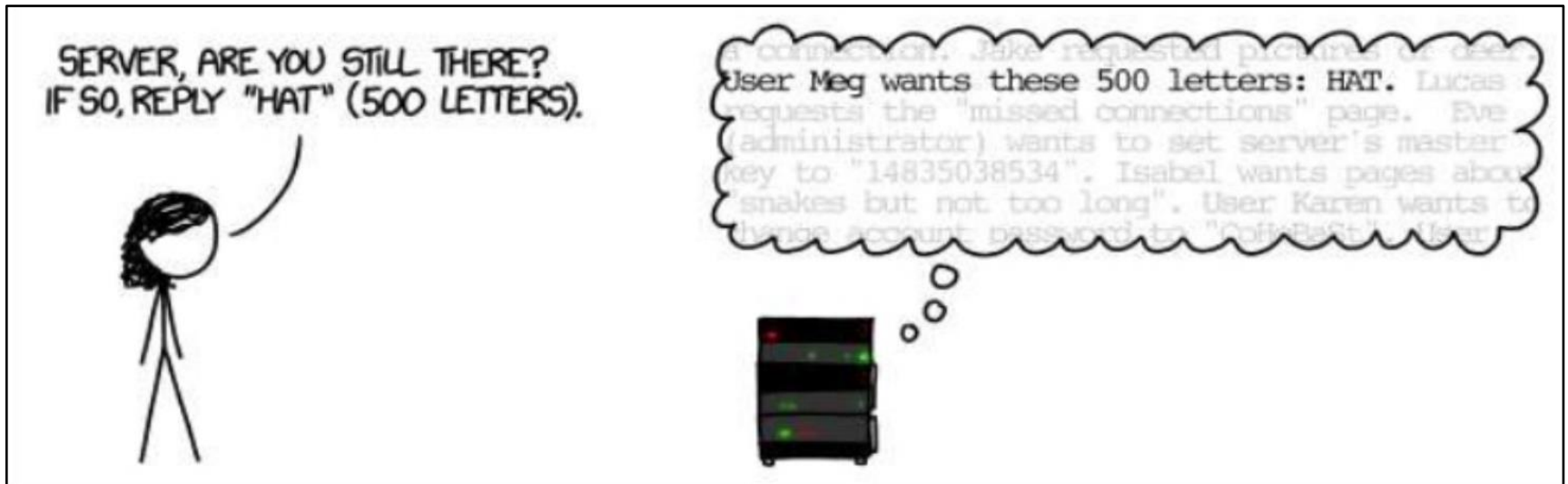
# Heartbeat with incoming Message (correctly) buffered

12



# The Heartbleed Attack

13



# Heartbeat Request / Response Message

14

- The Heartbeat protocol messages consist of their type and an arbitrary payload and padding.

heartbeat\_request or heartbeat\_response

- struct {  
HeartbeatMessageType type;  
uint16 payload\_length;  
opaque payload[HeartbeatMessage.payload\_length];  
opaque padding[padding\_length];  
} HeartbeatMessage;

16+ bytes of random  
content, ignored by receiver

- The sender composes a request message containing a payload with a specified length (i.e. *payload\_length*)
- The receiver returns a response message containing a copy of the sender's payload (with length *payload\_length*)
- “opaque” seems to be a typedef (i.e. unsigned char)

# Pseudo-Code Example (correct)

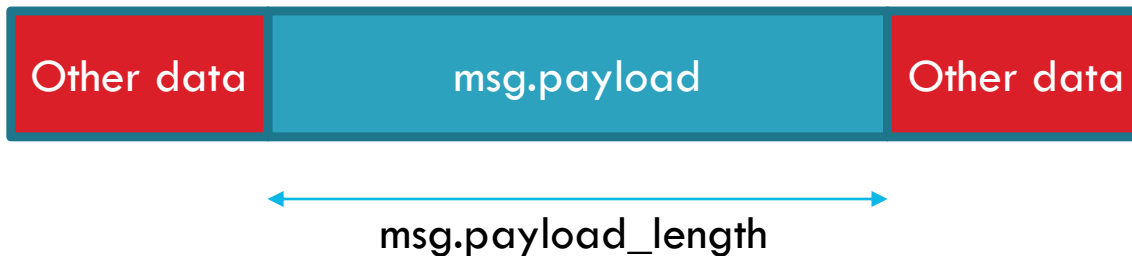
15

**Sender (constructs correct request message):**

```
struct HeartbeatMessage msg;  
msg.HeartbeatMessageType = heartbeat_request;  
msg.payload_length = 2;  
alloc(msg.payload, 2); // Note that the payload array is dynamically allocated  
msg.payload = "AB";  
...
```

**Receiver (receives above incoming msg) embedded in TCP/IP/TLS packet and constructs response s\_msg:**

```
struct HeartbeatMessage s_msg;  
s_msg.HeartbeatMessageType = heartbeat_response;  
s_msg.payload_length = msg.payload_length;  
alloc (s_msg.payload, msg.payload_length);  
memcpy(s_msg.payload, msg.payload, msg.payload_length);  
...
```



# Heartbleed Exploit

16

- The server receives a request message and stores in in (stack and heap) memory
  - ▣ Memory also contains other sessions-related information including tokens, keys, session IDs etc. from other sessions
- If `(uint16) payload_length` is actually larger than `(opaque) payload[..]`, the server will copy heap memory content beyond the payload array into the response message payload array (e.g. `ret_payload`), which is then sent back to the sender:

*`memcpy(ret_payload, payload, payload_length);`*

`memcpy(s_msg.payload, msg.payload, msg.payload_length);`



# Pseudo-Code Example (Heartbleed Exploit)

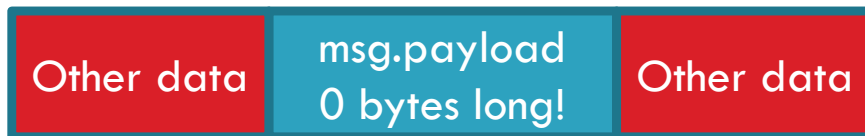
17

**Sender (constructs correct request message):**

```
struct HeartbeatMessage msg;  
msg.HeartbeatMessageType = heartbeat_request;  
msg.payload_length = 0xFFFF;  
msg.payload = "";  
...
```

**Receiver (receives above incoming msg) embedded in TCP/IP/TLS packet and constructs response s\_msg:**

```
struct HeartbeatMessage s_msg;  
s_msg.HeartbeatMessageType = heartbeat_response;  
s_msg.payload_length = msg.payload_length;  
alloc(s_msg.payload, msg.payload_length);  
memcpy(s_msg.payload, msg.payload, msg.payload_length);  
...
```



← msg.payload\_length →

# Heartbleed Exploit Extract (Python Code)

18

□ <https://gist.github.com/eelsivart/10174134>

Heartbleed (CVE-2014-0160) Test & Exploit Python Script

```
heartbleed.py Raw
1  #!/usr/bin/python
2
3  # Modified by Travis Lee
4  # Last Updated: 4/21/14
5  # Version 1.16
6  #
7  # -changed output to display text only instead of hexdump and made it easier to read
8  # -added option to specify number of times to connect to server (to get more data)
9  # -added option to send STARTTLS command for use with SMTP/POP/IMAP/FTP/etc...
10 # -added option to specify an input file of multiple hosts, line delimited, with or without a port specified (host:port)
11 # -added option to have verbose output
12 # -added capability to automatically check if STARTTLS/STLS/AUTH TLS is supported when smtp/pop/imap/ftp ports are entered and automatically
13 # -added option for hex output
14 # -added option to output raw data to a file
15 # -added option to output ascii data to a file
16 # -added option to not display returned data on screen (good if doing many iterations and outputting to a file)
17 # -added tls version auto-detection
18 # -added an extract rsa private key mode (orig code from epixoip. will exit script when found and enables -d (do not display returned data
19 # -requires following modules: gmpy, pyasn1
20
21 # Quick and dirty demonstration of CVE-2014-0160 by Jared Stafford (jspenguin@jspenguin.org)
22 # The author disclaims copyright to this source code.
23
24 import sys
25 import struct
26 import socket
27 import time
28 import select
29 import re
```

# What can be leaked?

19

```
38 20 2E 4E 45 R 2.0.50727; .NET CLR 3.5.30729;
30 37 32 39 38 T CLR 3.5.30729;
2E 30 2E 33 30 .NET CLR 3.0.30729; .NET CL
43 65 6E 74 65 729; Media Centre
6E 66 6F 50 61 r PC 5.0.0; Infopath
2E 30 43 3E 20 th.2; .NET4.0C;
48 6F 73 74 3A .NET4.0C)..Host:
6E 65 79 77 65 [redacted]
8E 65 63 74 69 11.com; Connecti
69 76 65 0D 0A on: keep-Alive..
20 73 69 64 65 cookie: doc-side
6D 79 77 6F 72 bar=245px; mywor
3D 66 61 6C 73 [redacted]
49 44 3D 43 32 e; SESSIONID=C2
41 34 42 44 31 7E804DA97D5A48D1
46 33 37 3E 20 B9728EDA58F37;
73 69 64 65 67 [redacted]
35 3E 20 2F 70 ar.width=285; /p

63 63 6E 3D (direct)utmccn=
63 6D 64 3D (direct)utmcd=
64 2E 74 6F [redacted]
42 4E 42 34 [redacted]
67 41 30 30 [redacted]
39 7E 03 69 [redacted]
3D 41 37 47 [redacted]
4F 4A 58 46 s-SRK5-081D-0JXF
37 62 64 32 12525dfa358e7bd2
38 33 66 38 97249f7b50b883f8
6E 3E 20 4A a44fa928e11in; j

4C 3E [redacted]
5 30 .NET CLR 2.0.50
3 2E 727; .NET CLR 3.
3 4C 5.30729; .NET CL
5 64 R 3.0.30729; Med
E 30 1a Center PC 6.0
E 4E ; Infopath.2; .N
0 45 ET4.0C; .NET4.0E
4 69 ).Accept-Encodi
1 74 ng: gzip, deflat
2 61 e..Host: [redacted]
0 0A [redacted]
5 70 Connection: Keep
A 20 -Alive..Cur[redacted]
E 7A [redacted]
```

Server details

```
0 00 00 00 00 00
E 78 73 72 66 2E at [redacted].xsr[redacted]
0 35 52 4B 35 2D token=A7G5-SRK5-
7 65 63 36 63 34 081D-0JXF17ec6a4
7 33 38 35 3E 36 da22594b48738566
9 30 33 31 31 35 c331f7969a903115
3 45 53 53 45 4F 9b|out; 1SESSIO
9 45 36 37 44 47 NTD=FC3103986706
3 44 31 34 37 38 A80561B4B43D1478
1 CD 12 93 24 C6 SDF0....1.A...S.
0 34 03 03 03 03 .....GT.....
C 2E 63 6F 6D 7C [redacted]
6 65 72 72 61 6C [redacted]
E 6E 6E 73 73 6E [redacted]
```

Keys?

```
09 00 79 09 0E 07 43 trw[suspect]ryingw
25 37 44 25 32 30 6F 20and%2Fpe%70%20o
64 61 79 2E 25 32 30 rd%20thursday.%20
6E 75 6D 62 65 72 3A .co-e103-number:
30 20 65 31 20 34 30 7701705.co-e104
25 32 30 74 58 65 25 string:if%20the%
73 25 32 30 67 69 76 20day%20is%20gily
30 74 6F 64 61 79 27 ent%20as%20today"
30 6F 72 25 62 30 74 s%20day%20or%20t
25 32 30 64 61 79 25 omorrow"s%20day%
74 61 74 25 62 30 72 20thermostat%20r
30 77 69 6C 6C 25 32 esponse%20will%2
69 6C 67 25 62 30 77 0be%20end%20ing%20w
25 32 30 6E 65 78 74 1th%20%22%20next
61 79 25 32 60 25 32 %20Thursday%20%2
6D 6F 72 72 6F 77 25 0%2F%20tomorrow%
70 74 25 32 60 6E 61 22%20except%20ona
30 74 68 65 65 72 30 me%20of%20th%20
30 20 65 31 30 35 3D day.%20.co-e105-
53 53 45 44 0A 63 30 string:PASSED.CO
62 65 72 3A 31 32 0A -e106-number:12.
```

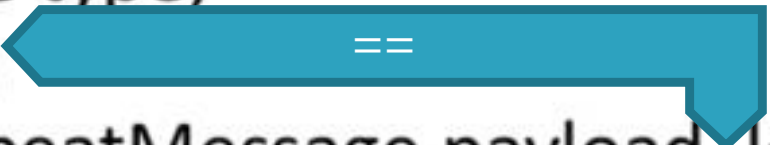
CSRF tokens

```
8 65 [redacted]
3 61 [redacted]
1 65 [redacted]
2 73 [redacted]
2 61 [redacted]
1 4C [redacted]
0 5A [redacted]
4E 26 [redacted]
69 70 [redacted]
26 6E [redacted]
38 48 [redacted]
A 41 [redacted]
2 41 [redacted]
66 4D [redacted]
48 4F [redacted]
69 5A [redacted]
1 76 [redacted]
7 55 [redacted]
69 68 [redacted]
64 67 [redacted]
6 72 [redacted]
3 46 [redacted]
0 53 [redacted]
11 70 [redacted]
13 65 [redacted]
13 74 [redacted]
15 72 [redacted]
11 20 [redacted]
```

# What happened next?

20

- ❑ The Heartbleet bug was fixed (of course)
- ❑ Further checks and balances were added to validate that payload length was correct

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

# Pseudo-Code Example (Heartbleed Exploit Fixed)

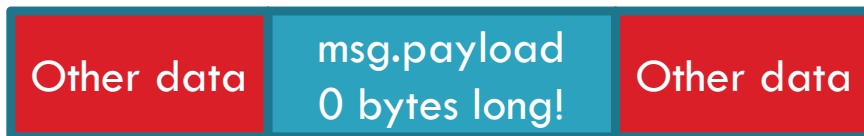
21

**Sender (constructs correct request message):**

```
struct HeartbeatMessage msg;  
msg.HeartbeatMessageType = heartbeat_request;  
msg.payload_length = 0xFFFF;  
msg.payload = "";  
...
```

**Receiver (receives above incoming msg) embedded in TCP/IP/TLS packet and constructs response s\_msg:**

```
struct HeartbeatMessage s_msg;  
int correctPayloadLen = len(msg.payload);  
s_msg.HeartbeatMessageType = heartbeat_response;  
s_msg.payload_length = correctPayloadLen;  
alloc(s_msg.payload, correctPayloadLen);  
memcpy(s_msg.payload, msg.payload, correctPayloadLen);  
...
```

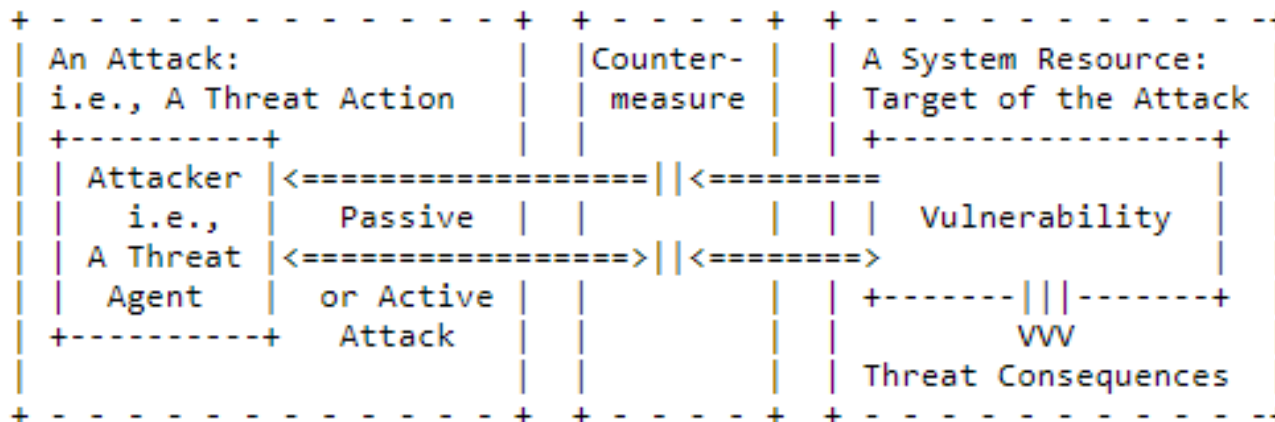


← msg.payload\_length →

# Recall (Menti Question): Attack (RFC2828, Internet Security Glossary)

22

- An assault on system security that derives from an intelligent threat, i.e. a deliberate attempt
- An "active attack" attempts to alter system resources or affect their operation
- A "passive attack" attempts to learn or make use of information from the system, but does not affect system resources



M

# Lessons learnt

24

- *OpenSSL core developer Ben Laurie claimed that a security audit of OpenSSL would have caught Heartbleed*
- *Some other quotes from the security community:*
  - *“Think about it, OpenSSL only has two fulltime people to write, maintain, test, and review 500,000 lines of business critical code”*
  - *“The mystery is not that a few overworked volunteers missed this bug; the mystery is why it hasn't happened more often”*
  - *“There should be a continuous effort to simplify the code, because otherwise just adding capabilities will slowly increase the software complexity. The code should be refactored over time to make it simple and clear, not just constantly add new features. The goal should be code that is “obviously right”, as opposed to code that is so complicated that “I can't see any problems”*