# Limits of Proofs: Implementation Attacks / Heartbleed

Advanced Topics in ~~Cybersecurity~~Cryptography (7CCSMATC)

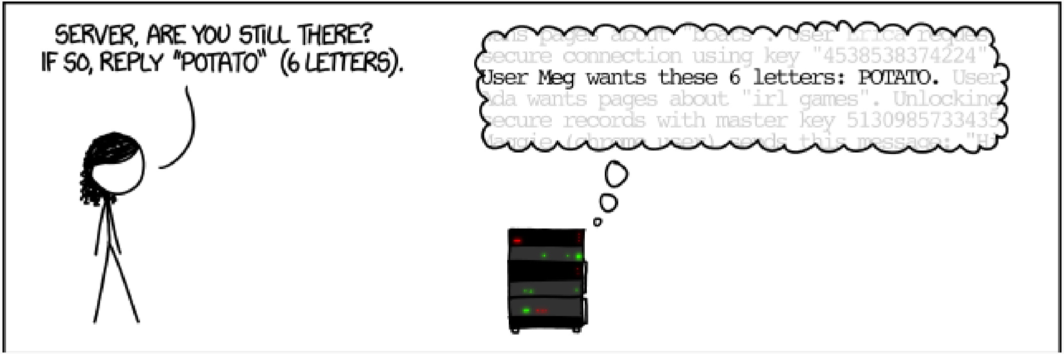
---

Martin R. Albrecht

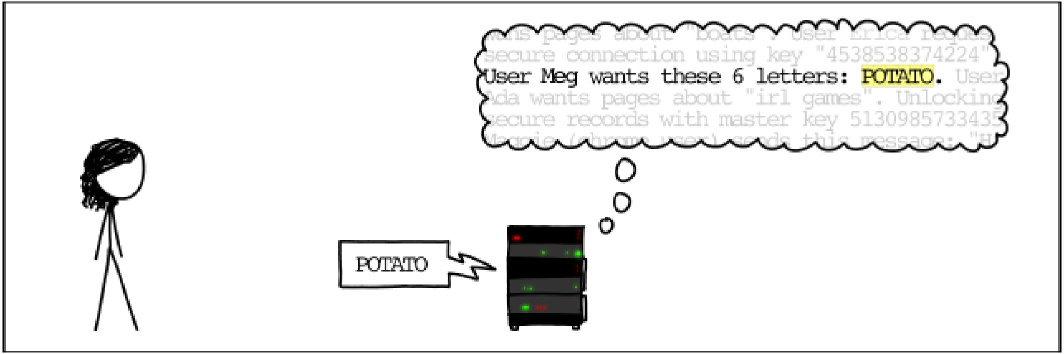
# Programme

The last lecture dealt with an attack exposing the shortcoming of a prior model. This shortcoming was addressed in a subsequently published model. Implementation vulnerabilities are vulnerabilities usually not covered by any model.

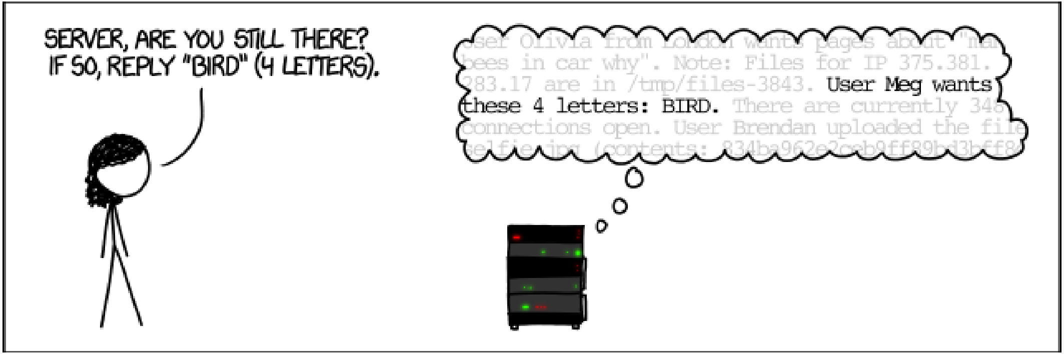
# XKCD #1354


SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).
secure connection using key "4538538374224"
User Meg wants these 6 letters: POTATO.

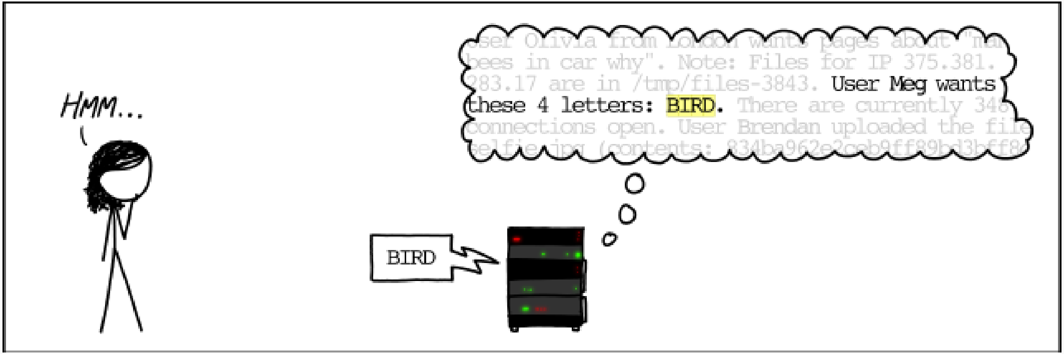
# XKCD #1354


secure connection using key "4538538374224"
User Meg wants these 6 letters: POTATO. User
Ada wants pages about "irl games". Unlocking
secure records with master key 5130985733435
POTATO

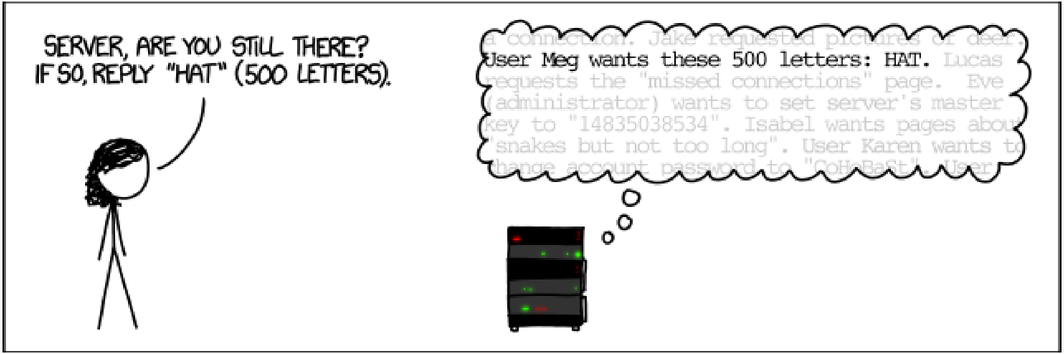
# XKCD #1354


SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
bees in car why". Note: Files for IP 375.381.
283.17 are in /tmp/files-3843. User Meg wants
these 4 letters: BIRD. There are currently
connections open. User Brendan uploaded the

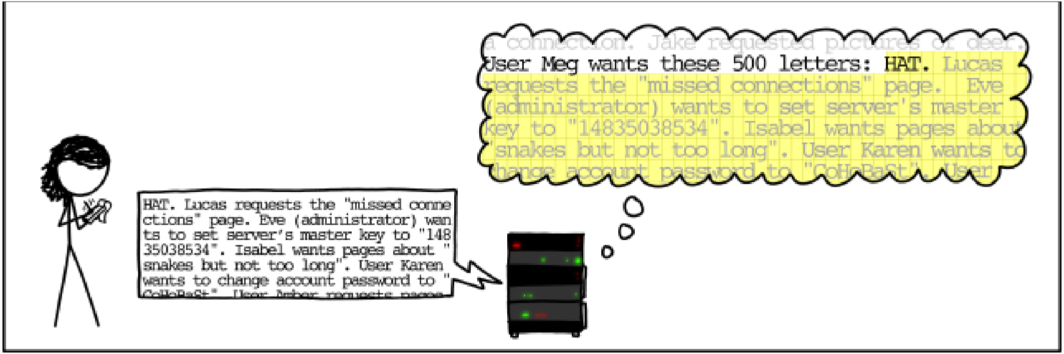
# XKCD #1354


HMM...
Olivia from London wants pages about
bees in car why". Note: Files for IP 375.381.
283.17 are in /tmp/files-3843. User Meg wants
these 4 letters: BIRD. There are currently
connections open. User Brendan uploaded the
BIRD

# XKCD #1354


SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
a connection. Jake requested pictures of deer.
User Meg wants these 500 letters: HAT. Lucas
requests the "missed connections" page. Eve
(administrator) wants to set server's master
key to "14835038534". Isabel wants pages about
"snakes but not too long". User Karen wants to
change account password to "CoHoBaSt". User

# XKCD #1354


HAT. Lucas requests the "missed conne
ctions" page. Eve (administrator) wan
ts to set server's master key to "148
35038534". Isabel wants pages about "
snakes but not too long". User Karen
wants to change account password to "
a connection. Jake requested pictures of deer.
User Meg wants these 500 letters: HAT. Lucas
requests the "missed connections" page. Eve
(administrator) wants to set server's master
key to "14835038534". Isabel wants pages about
"snakes but not too long". User Karen wants to
change account password to "CoHoBaSt". User

# RFC 6520: Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension I

*This document describes the Heartbeat Extension for the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols.*
*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.*

— *RFC 6520*

# RFC 6520: Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension II

- The heartbeat extension implements a simple ping:
  "Are you still there?"
- This is a common feature for communication protocols:
  - ICMP
  - HTTP keep-alive
  - SSH KeepAlive
  - ...

# RFC 6520: Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension III

> *The larger takeaway actually isn't "This wouldn't have happened if we didn't add Ping", the takeaway is "We can't even add Ping, how the heck are we going to fix everything else?".*
>
> — *Dan Kaminsky*

# The bug …

- … is not in the standard: RFC 6520.
- … is in an implementation of RFC 6520 in OpenSSL.
- … affectes OpenSSL versions
  - 1.0.1 (up until and including 1.0.1f) and
  - 1.0.2 (beta).
- …does not affect other SSL implementations as far as we know.

# What is OpenSSL?

- TLS (formerly known as SSL) is the most used protocol to encrypt traffic on the Internet.
- OpenSSL is an implementation of the TLS standard.
- OpenSSL is widely used.

# Show me simplified code

```
struct {
    unsigned short len;
    char payload[];
} *packet;

packet = malloc(amt);
read(s, packet, amt);
buffer = malloc(packet->len);
/* malb: packet->len == amt? */
memcpy(buffer, packet->payload, packet->len);
write(s, buffer, packet->len);
```

Credit: http://www.tedunangst.com/flak/post/heartbleed-vs-mallocconf

# Show me the actual code I

Here is the `HeartbeatMessage` packet definition:

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

# Show me the actual code II

Here is the function processing heartbeats in OpenSSL:

```
int  dtls1_process_heartbeat(SSL *s) {
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */
```

so we get a pointer to the data within an SSLv3 record:

```
typedef struct ssl3_record_st {
  int type;               /* type of record */
  unsigned int length;    /* How many bytes available */
  unsigned int off;       /* read/write offset into 'buf' */
  unsigned char *data;    /* pointer to the record data */
  unsigned char *input;   /* where the decode bytes are */
  unsigned char *comp;    /* only used with decompression  ... */
  unsigned long epoch;    /* epoch number, needed by DTLS1 */
  unsigned char seq_num[8]; /* sequence number, needed by DTLS1 */
} SSL3_RECORD;
```

# Show me the actual code III

Back to `dtls1_process_heartbeat`:

```
/* Read type and payload length first */
hbtype = *p++; // malb: p points to start of the message
n2s(p, payload); //malb: read 16-bit length into payload and p+=2
pl = p; // malb: points to payload sent by user
```

Later on, the reply is constructed, but first sufficient memory is allocated:

```
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;
```

# Show me the actual code IV

Then the message is constructed:

```
/* Enter response type, length and copy payload */
*bp++ = TLS1_HB_RESPONSE; //malb: set the type
s2n(payload, bp); //malb: write length and bp+=2
memcpy(bp, pl, payload); //malb: fire!
```

# Show me the actual code V

The offending line is

```
memcpy(bp, pl, payload);
```

where we copy `payload` bytes from `pl`.

The variable `payload` is controlled by the user as is the actual length of data in `pl`.

The user can hence request a read from `pl` requesting more data than `pl` has – up to 64kb.

**What is after `pl`?**

A pot of gold!

# Memory I

- `pl` lives on the heap.
- Memory from the heap is requested with **`malloc()`** and returned to the OS with **`free()`**.
- When asked for a certain number of bytes, the OS will find a bit of unused memory and return that (if **`mmap()`** isn't called)

```
-------- | -------- | -------- | --------
-------- | -------- | --xxxxxx | --------
-------- | -------- | -------- | -------- <- many bytes free
xxxxxxxx | -------- | -------- | --------
xxxxxxxx | xxxxxxxx | xxxxxxxx | xxxxxxxx
xxxxxxxx | xxxxxxxx | xxx--xxx | xxxxxxxx <- 2 bytes free
xxxxxxxx | xx------ | xxxxxxxx | xxxxxxxx <- 4 bytes free
```

- depending on where the OS put **`pl`** different data is after it

# Memory II

- If the data behind `p1` is in use, the attacker gets to see that
- If the data behind `p1` is not in use and was returned to the OS, it depends on the OS if it is overwritten with dummy data or not.

However,

- OpenSSL seldomly returns data to the OS.
- Instead, unused memory is re-used internally for performance reasons.
- This renders exploit mitigation techniques (such as overwriting free'd data) useless.

# The Fix

```c
/* Read type and payload length first */
if (1 + 2 + 16 > s->s3->rrec.length)
   return 0; /* silently discard */
hbtype = *p++;
n2s(p, payload);
if (1 + 2 + payload + 16 > s->s3->rrec.length)
    return 0; /* silently discard per RFC 6520 sec. 4 */
pl = p;
```

This does two things:

- The first check stops zero-length heartbeats.
- The second check checks to make sure that the actual record length is sufficiently long.

# Ping!