Conference on Computer Intellectual Systems and Networks
CISN-2016
Kryvyi Rih

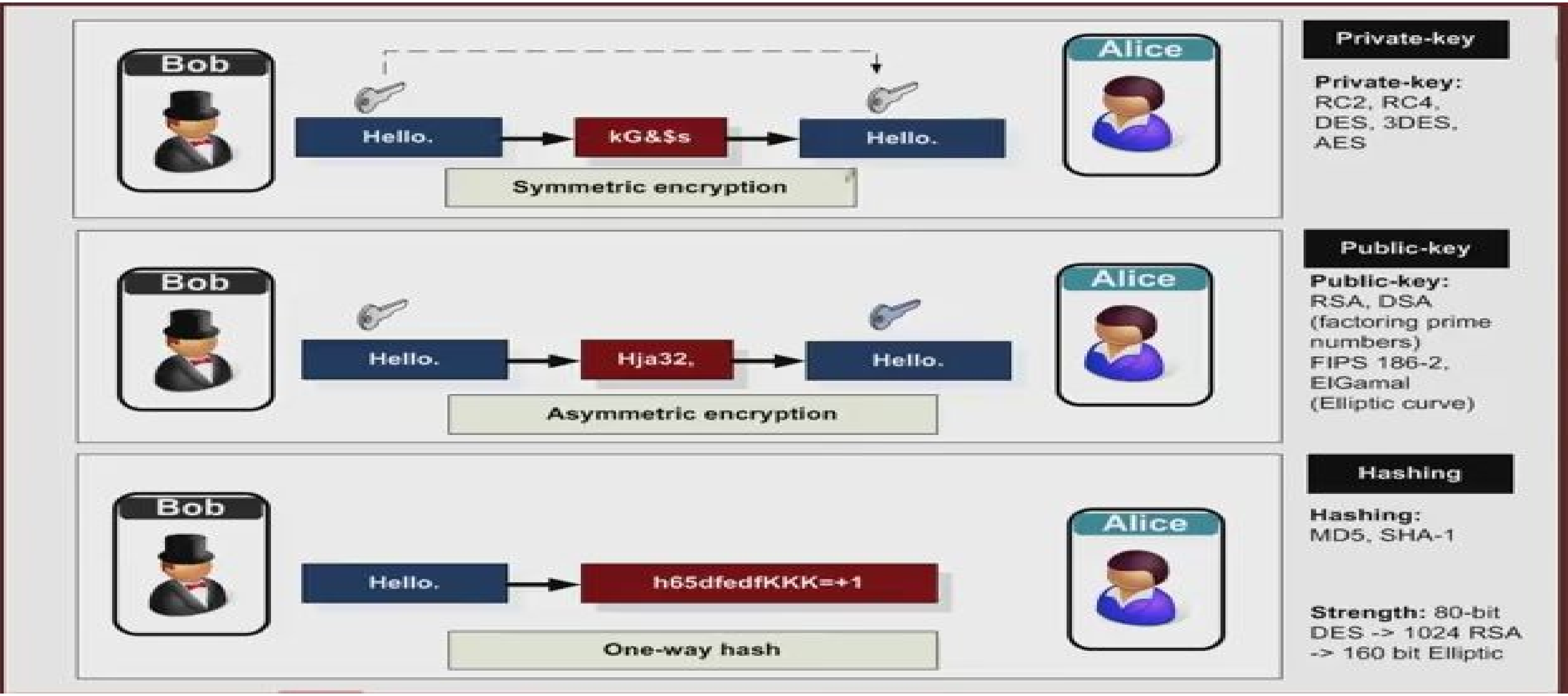
Heartbleed Vulnerability

Presented by
Mbah Johnas Fortem
Candidate Dr. of Information Technology
Author of Heartbleed Bug OpenSSL Vulnerability
fortem751@gmail.com

What is Heartbleed and why OpenSSL?

- Heartbleed is a vulnerability in the OpenSSL software
- OpenSSL is a standard used by many organizations
- OpenSSL is the encryption software that accesses websites through a “secure” connection, HTTPS://

Cryptography Review



Private-key

Private-key:
RC2, RC4,
DES, 3DES,
AES

Public-key

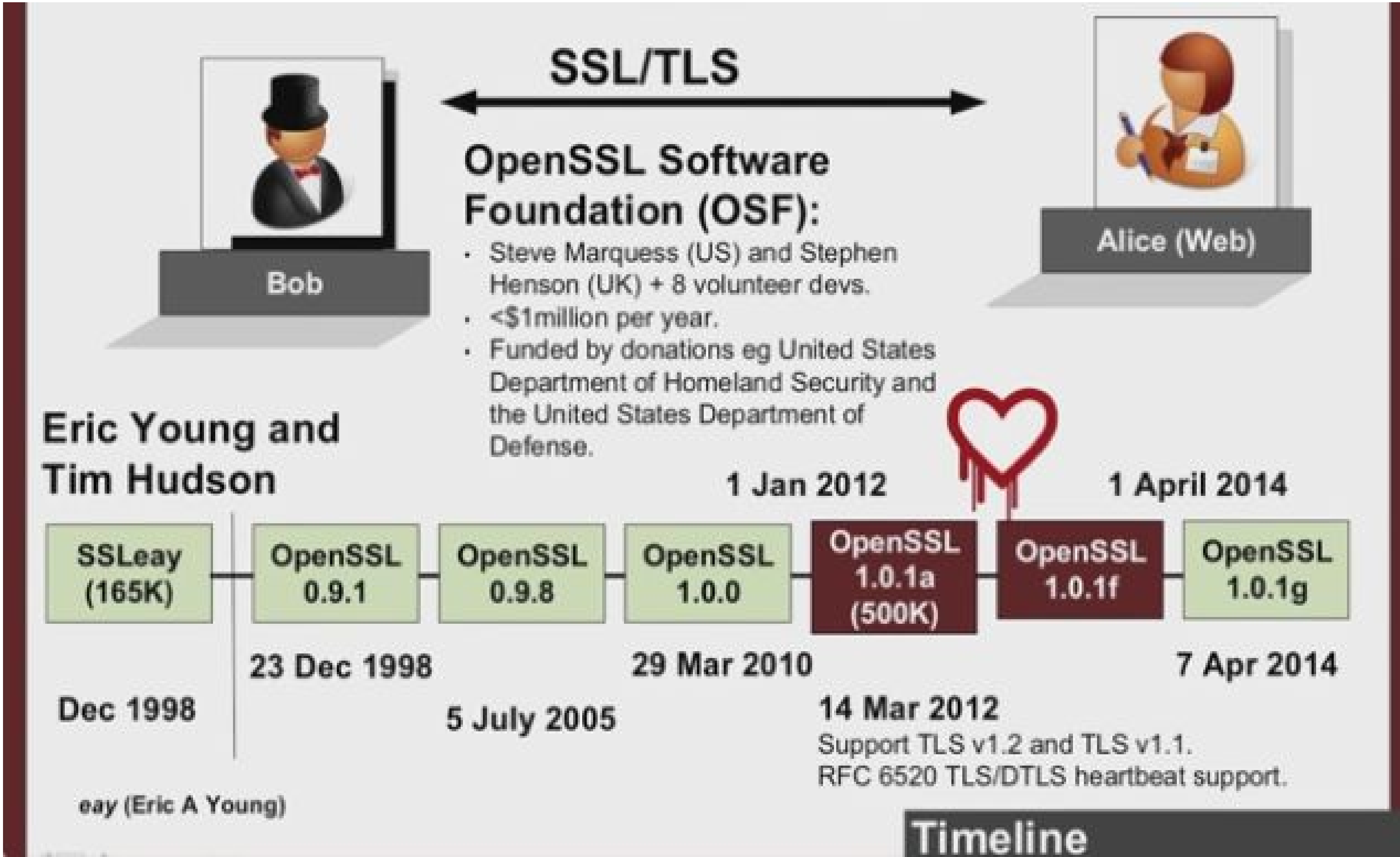
Public-key:
RSA, DSA
(factoring prime
numbers)
FIPS 186-2,
ElGamal
(Elliptic curve)

Hashing

Hashing:
MD5, SHA-1

Strength: 80-bit
DES -> 1024 RSA
-> 160 bit Elliptic

OpenSSL History



SSL/TLS

Bob

Alice (Web)

OpenSSL Software Foundation (OSF):

- Steve Marquess (US) and Stephen Henson (UK) + 8 volunteer devs.
- <$1million per year.
- Funded by donations eg United States Department of Homeland Security and the United States Department of Defense.

Eric Young and Tim Hudson

SSLeay (165K) — Dec 1998

OpenSSL 0.9.1 — 23 Dec 1998

OpenSSL 0.9.8 — 5 July 2005

OpenSSL 1.0.0 — 29 Mar 2010

OpenSSL 1.0.1a (500K) — 1 Jan 2012 — 14 Mar 2012

OpenSSL 1.0.1f

OpenSSL 1.0.1g — 1 April 2014 — 7 Apr 2014

14 Mar 2012
Support TLS v1.2 and TLS v1.1.
RFC 6520 TLS/DTLS heartbeat support.

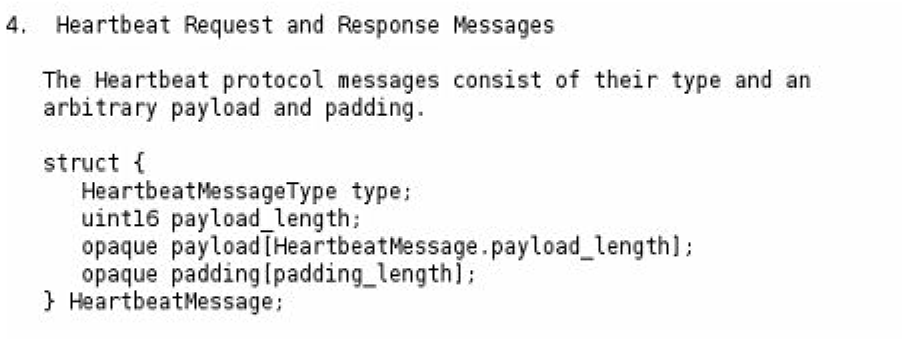
eay (Eric A Young)

Timeline

OpenSSL History cond.

- Stephen Henson only full-time developer
- Stephen Henson (UK-Mathematician) wrote 60% of entire code
- 31 December 2011 bug introduced by German developer Robin Segelmann through the addition of the Heartbeat extension protocol and okayed by Stephen Henson.
- Steve Marquess “no money going towards reviewing the code or performing audits”.
- Bug was introduced into OpenSSL version 1.0.1 code on 14 March 2012
- Funding has dropped drastically since Heartbleed

Heartbeat Protocol



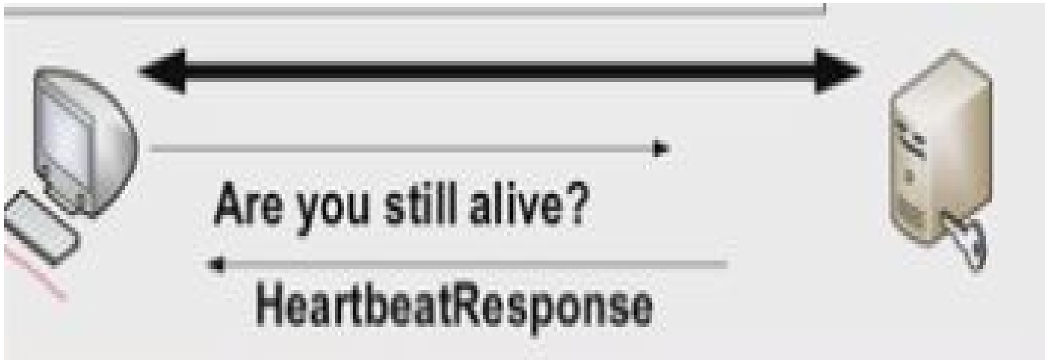
```
4.  Heartbeat Request and Response Messages

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

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

Heartbeat protocol is used to keep a TLS connection alive without the need to constantly renegotiate the SSL session.

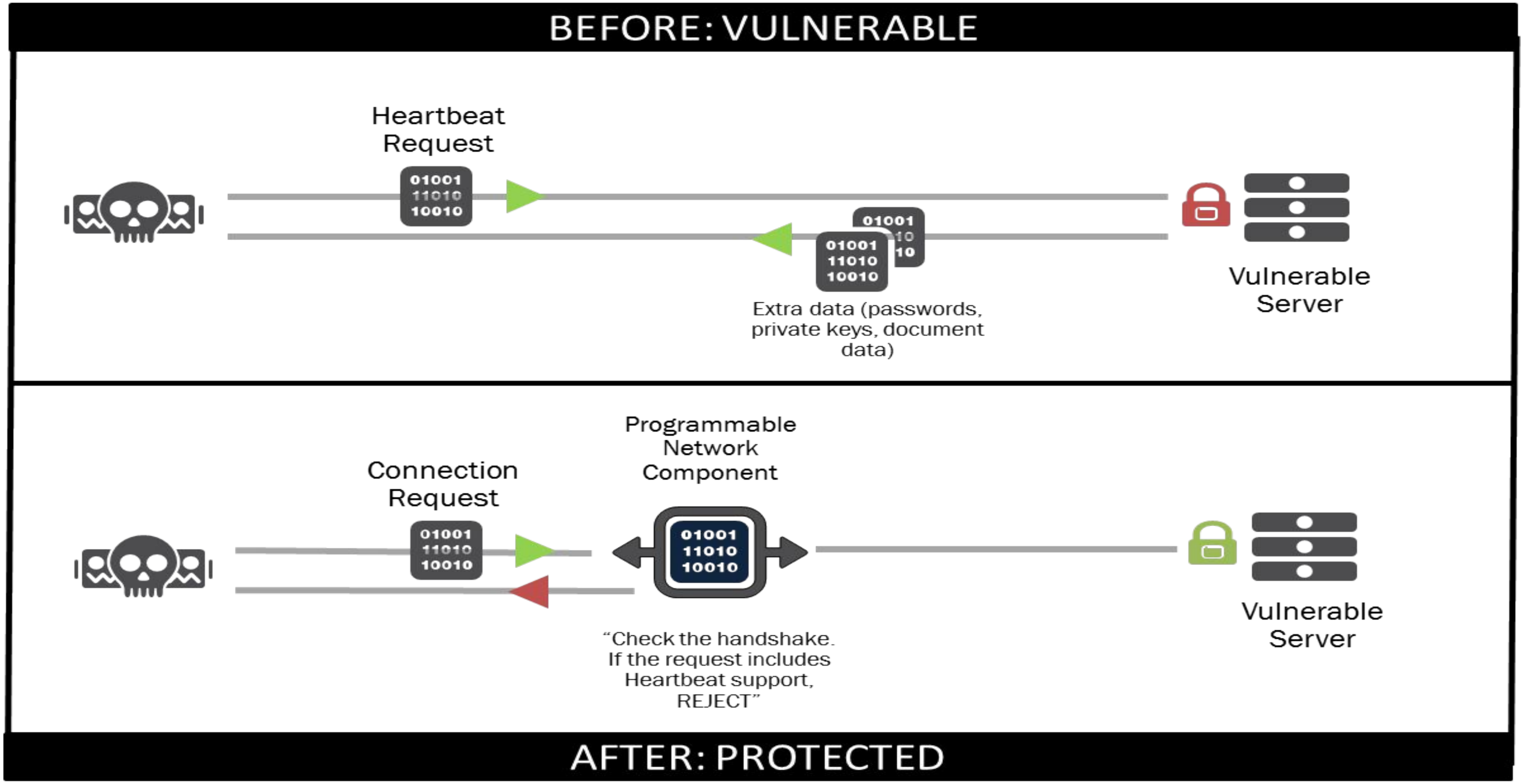
Heartbeat protocol is used to keep a TLS connection alive without the need to constantly renegotiate the SSL session.



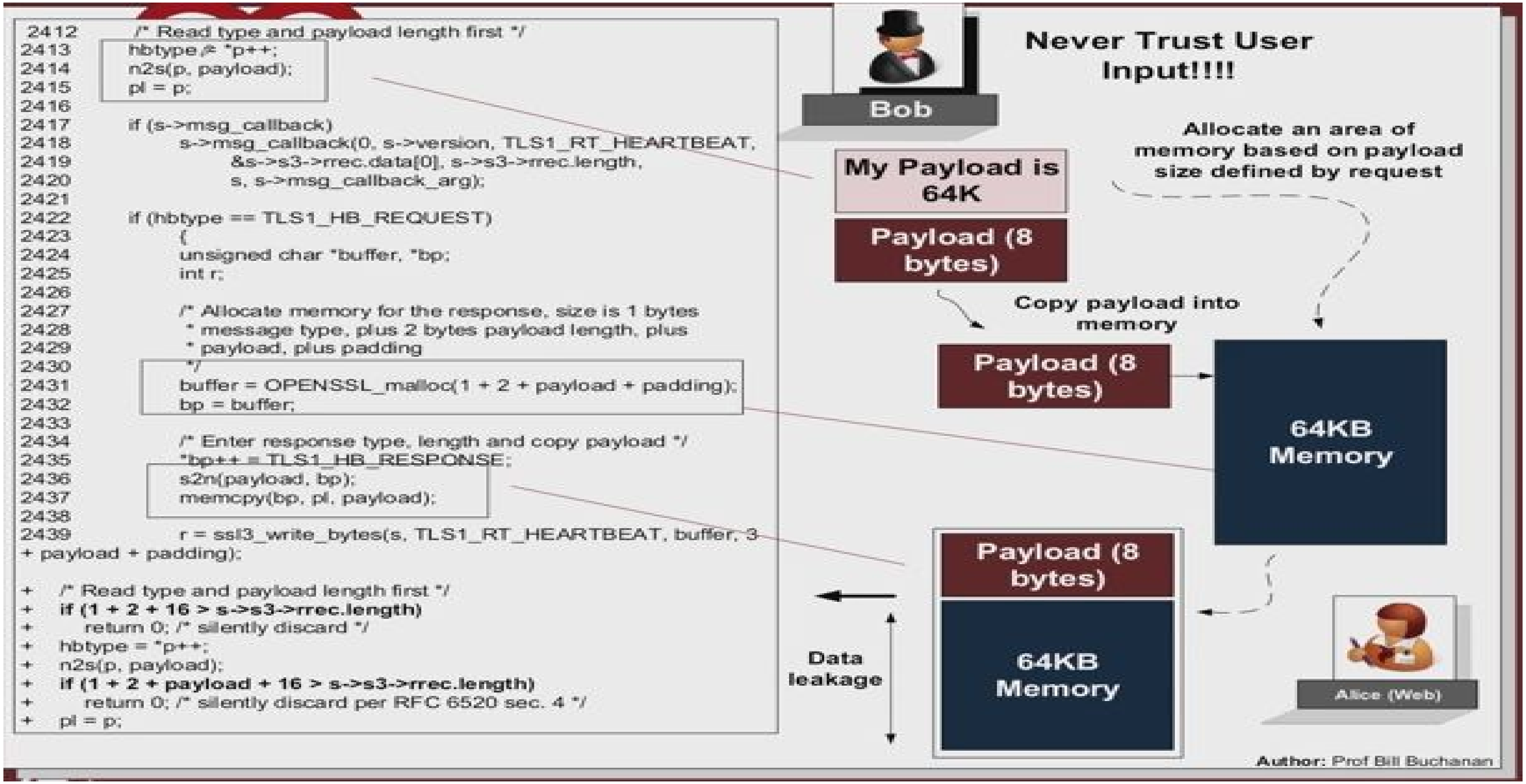
Heartbeat protocol is used to keep a TLS connection alive without the need to constantly renegotiate the SSL session.

Why would you Need a payload Just to check if A machine is alive?

Heartbeat Protocol cont.



Heartbleed Attack why and how



```
        /* Read type and payload length first */
        hbtype = *p++;
        n2s(p, payload);
        pl = p;

        if (s->msg_callback)
                s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
                        &s->s3->rrec.data[0], s->s3->rrec.length,
                        s, s->msg_callback_arg);

        if (hbtype == TLS1_HB_REQUEST)
                {
                unsigned char *buffer, *bp;
                int r;

                /* 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);

                r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3
+ payload + padding);

+   /* 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;
```

Author: Prof Bill Buchanan

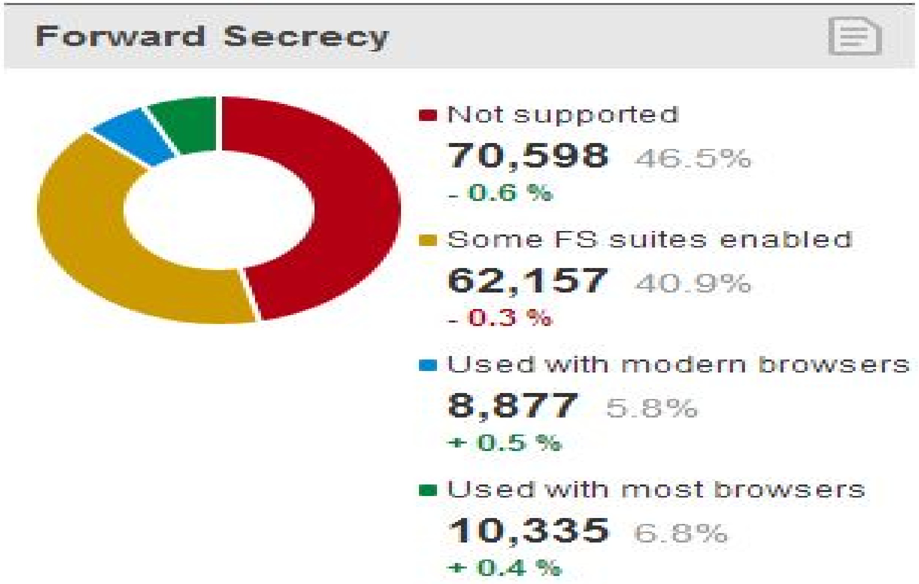
Memory disclosure: what exactly can an attacker get?

- Private crypto keys - the keys to the server.
- Usernames and Passwords
- Session identifiers
- Private data – data payloads
- Meta data for the SSL session, programming structure pointers - may defeat other exploit protections.

Could the Bug have been Prevented or Detected?

- Prevented – No.
- Detected to some extend - through the use of Perfect Forward Secrecy ciphersuites
- If Incorporated into browsers would have prevented clients from repeated Heartbleed

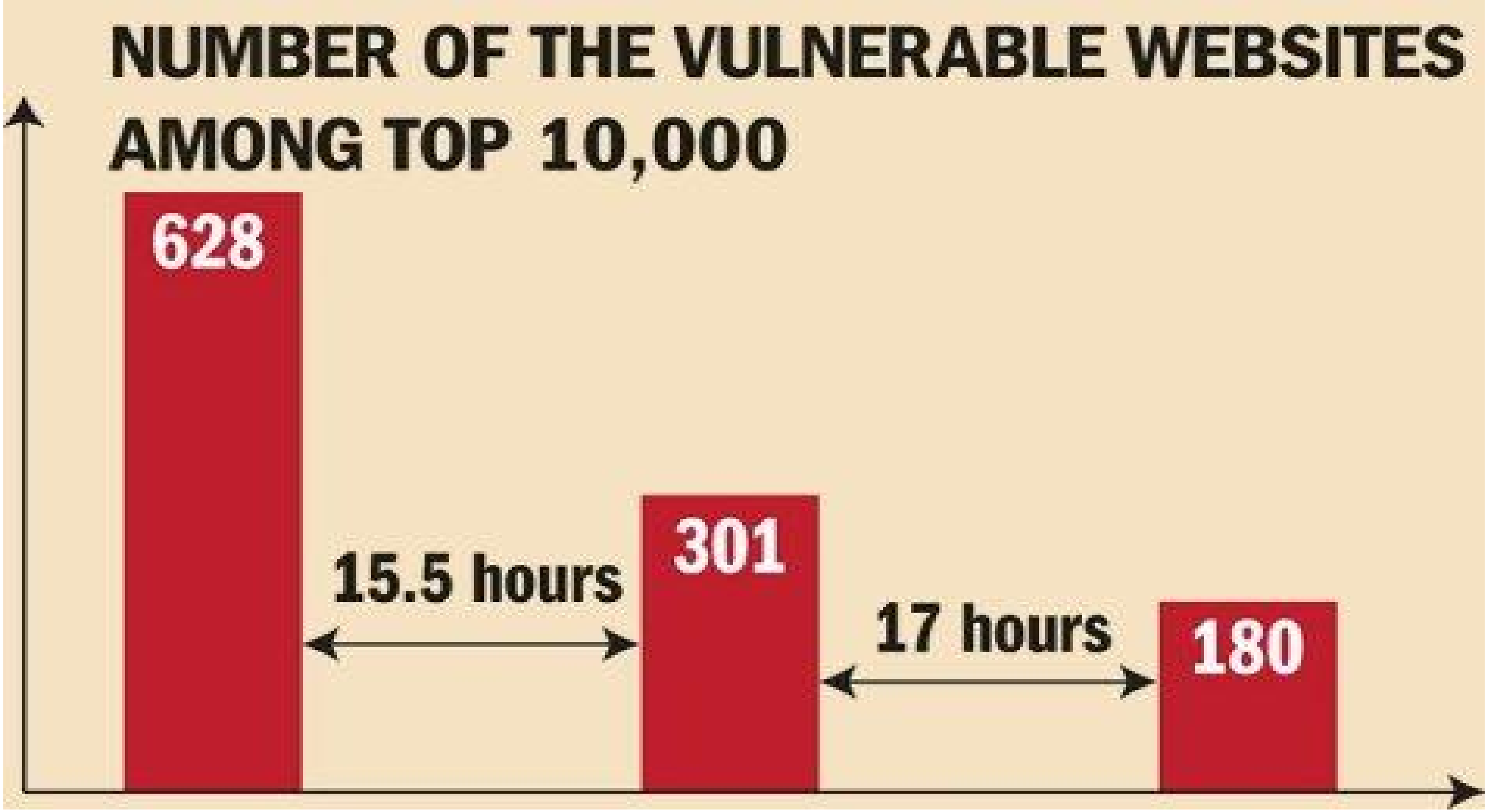
  attacks after server patch.



Why was it Difficult to Detect Heartbleed?

- According to specifications of the heartbeat protocol, the heartbeat response message must be the exact copy of the payload of the heartbeat request message.
- But, this constraint is not enforced in its implementation allowing up to 64K to exfiltrate back to an attacker.
- There is no existence of omniscient automated tools, and the available tools do not possess the reasoning to discern facts like this.

How many sites were vulnerable? (After vulnerability was reported publically)



So what can I do?

- Coordinate with vendors to get vulnerable devices patched or replaced. At a minimum, revoke and reissue vulnerable certificate.
- Change passwords - even if a vendor says their product was not vulnerable, they CANNOT guarantee any business partners products were not vulnerable.
- Monitor carefully for any evidence of identity theft.
- Prepare for phishing and social engineering campaigns leveraging Heartbleed into scaring people into divulging credentials.

End Of Presentation!!!!

Thanks for the Audience!!!