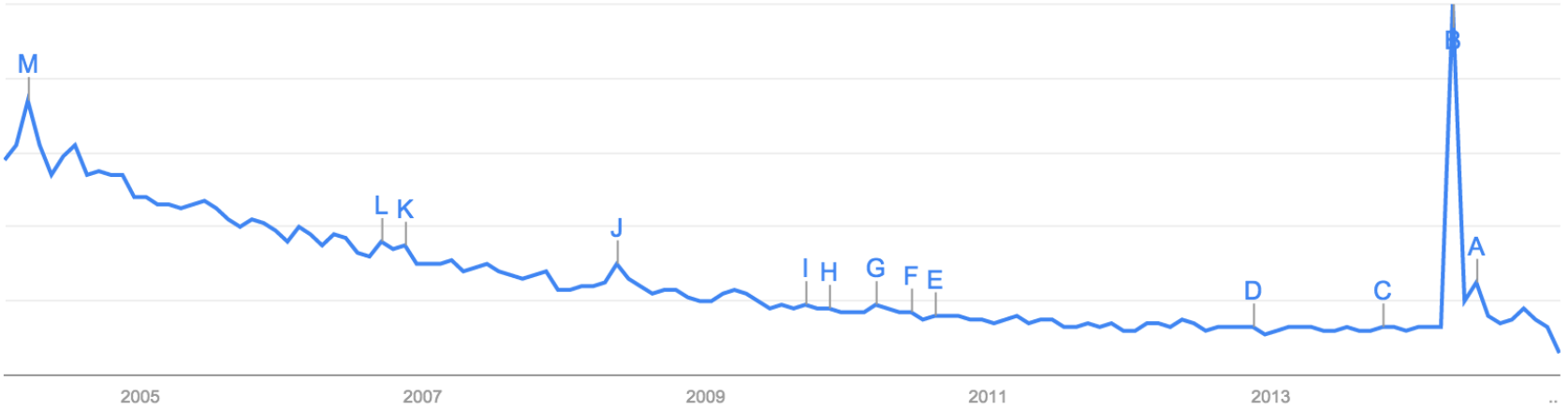
We ❤ SSL

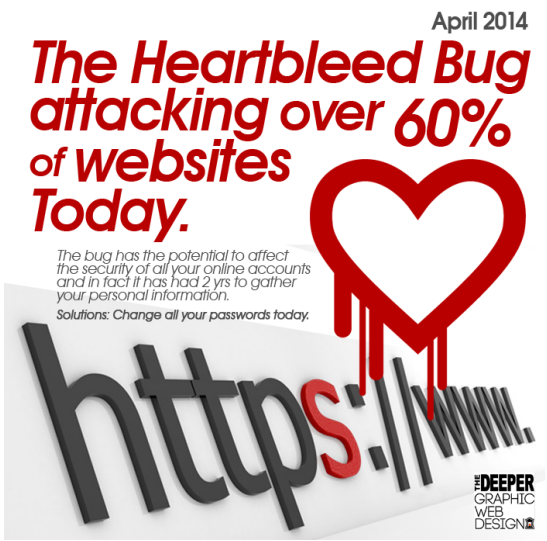
Emilia Käsper
OpenSSL / Google

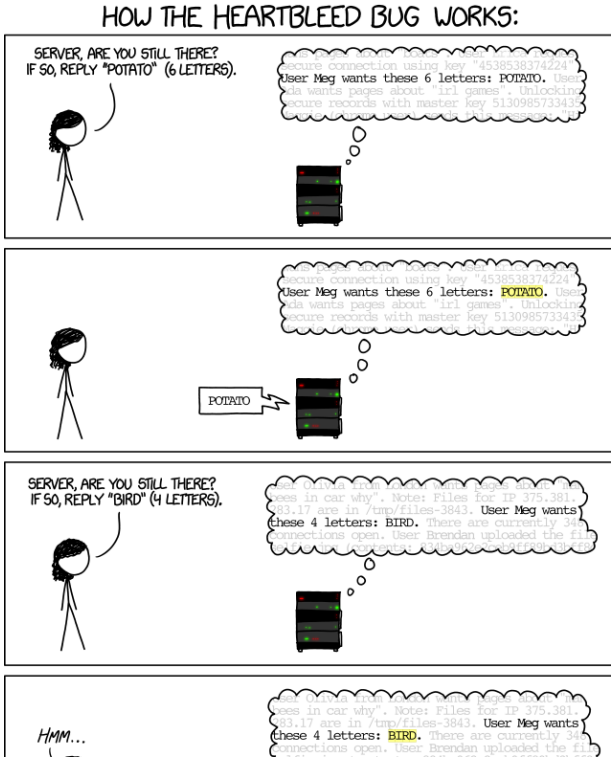
Let’s start with a guessing game...

What is this graph about?



Myth: Heartbleed broke the Internet





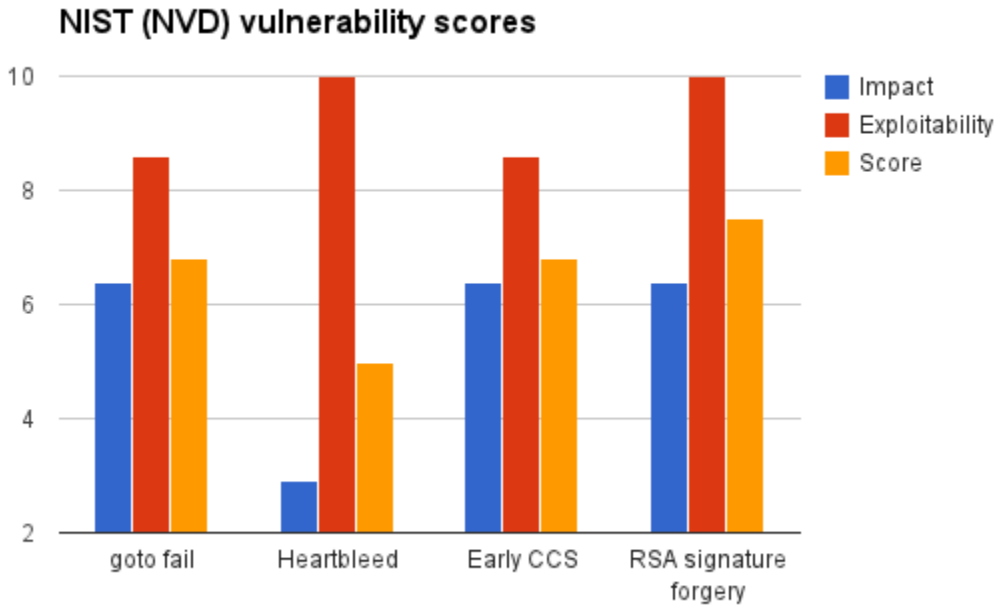
Fact: Internet-breaking bugs are common

- CVE-2011-0014 - infoleak, true impact unknown
- CVE-2012-2110 - possibly arbitrary code execution on reading certificates
- CVE-2012-2333 - buffer over-read, true impact unknown
- CVE-2014-1266 - “goto fail” server spoofing (Apple)
- CVE-2014-0160 - Heartbleed
- CVE-2014-0224 - “early CCS” disables encryption
- CVE-2014-1568 - RSA signature forgery (NSS)

In this talk...

- ~~A history of OpenSSL: the good, the bad and the ugly~~
- Heartbleed in the sea of exploits: why the hype, and what can we learn from this?
- The future of OpenSSL: what we're doing, and how you can help.

Heartbleed - why the attention?



Heartbleed - why the attention?

- Branding => press coverage, pop culture
- Changed awareness: Snowden
- Simplicity of exploit
- Remote code executions aren't concrete enough
- Offensive institutions are much better at judging bug impact. Recall…
  - CVE-2011-0014 - infoleak, true impact unknown
  - CVE-2012-2333 - buffer over-read, true impact unknown

Lesson #1: we need code review

```
617         hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
618     hashOut.length = SSL_SHA1_DIGEST_LEN;
619     if ((err = SSLFreeBuffer(&hashCtx, ctx)) != 0)
620         goto fail;
621
622     if ((err = ReadyHash(&SSLHashSHA1, &hashCtx, ctx)) != 0)
623         goto fail;
624     if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
625         goto fail;
626     if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
627         goto fail;
628     if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
629         goto fail;
630     if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
631         goto fail;
632
633         err = sslRawVerify(ctx,
634                            ctx->peerPubKey,
635                            dataToSign,                 /* plaintext */
636                            dataToSignLen,              /* plaintext length */
637                            signature,
638                            signatureLen);
639         if(err) {
640                 sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
641                     "returned %d\n", (int)err);
642                 goto fail;
643         }
644
645 fail:
646     SSLFreeBuffer(&signedHashes, ctx);
647     SSLFreeBuffer(&hashCtx, ctx);
648     return err;
649
```

```
618         hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
619     hashOut.length = SSL_SHA1_DIGEST_LEN;
620     if ((err = SSLFreeBuffer(&hashCtx)) != 0)
621         goto fail;
622
623     if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
624         goto fail;
625     if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
626         goto fail;
627     if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
628         goto fail;
629     if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
630         goto fail;
631         goto fail;
```

(Draft) 2014/05/14 14:48:15
The second goto fail; is not needed!
Edit

```
632     if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
633         goto fail;
634
635         err = sslRawVerify(ctx,
636                            ctx->peerPubKey,
637                            dataToSign,                 /* plaintext */
638                            dataToSignLen,              /* plaintext length */
639                            signature,
640                            signatureLen);
641         if(err) {
642                 sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
643                     "returned %d\n", (int)err);
644                 goto fail;
645         }
646
647 fail:
648     SSLFreeBuffer(&signedHashes);
649     SSLFreeBuffer(&hashCtx);
650     return err;
651
```

Add support for TLS/DTLS heartbeats.

Description

Add support for TLS/DTLS heartbeats.

Patch Set 1 (edit)

Created: 0 minutes ago

	Unified diffs	Side-by-side diffs	Delta from patch set	Stats (*+605 lines, -19 lines*)	
M	CHANGES	View		1 chunk	+6 lines, -0 lines
M	apps/s_cb.c	View		1 chunk	+20 lines, -0 lines
M	apps/s_client.c	View		1 chunk	+8 lines, -0 lines
M	apps/s_server.c	View		1 chunk	+10 lines, -0 lines
M	crypto/objects/obj_dat.h	View		9 chunks	+15 lines, -10 lines
M	crypto/objects/obj_mac.h	View		2 chunks	+6 lines, -1 line
M	crypto/objects/obj_mac.num	View		1 chunk	+2 lines, -1 line
M	crypto/objects/objects.txt	View		1 chunk	+1 line, -0 lines
M	crypto/rsa/rsa_pmeth.c	View		2 chunks	+16 lines, -2 lines
M	ssl/d1_both.c	View		2 chunks	+151 lines, -1 line
M	ssl/d1_clnt.c	View		1 chunk	+13 lines, -0 lines
M	ssl/d1_lib.c	View		1 chunk	+8 lines, -0 lines
M	ssl/d1_pkt.c	View		1 chunk	+13 lines, -0 lines
M	ssl/d1_srvr.c	View		1 chunk	+13 lines, -0 lines
M	ssl/dtls1.h	View		1 chunk	+1 line, -1 line
M	ssl/s3_clnt.c	View		1 chunk	+12 lines, -0 lines
M	ssl/s3_lib.c	View		1 chunk	+21 lines, -0 lines
M	ssl/s3_pkt.c	View		1 chunk	+13 lines, -0 lines
M	ssl/s3_srvr.c	View		1 chunk	+12 lines, -0 lines
M	ssl/ssl.h	View		6 chunks	+24 lines, -2 lines
M	ssl/ssl3.h	View		2 chunks	+4 lines, -0 lines
M	ssl/ssl_err.c	View		3 chunks	+4 lines, -0 lines
M	ssl/ssl_locl.h	View		1 chunk	+7 lines, -0 lines
M	ssl/t1_lib.c	View		8 chunks	+211 lines, -0 lines
M	ssl/tls1.h	View		2 chunks	+13 lines, -0 lines
M	util/mkdef.pl	View		1 chunk	+1 line, -1 line

```
2403 #ifndef OPENSSL_NO_HEARTBEATS
2404 int
2405 tls1_process_heartbeat(SSL *s)
2406         {
2407         unsigned char *p = &s->s3->rrec.data[0], *pl;
2408         unsigned short hbtype;
2409         unsigned int payload;
2410         unsigned int padding = 16; /* Use minimum padding */
2411
2412         /* Read type and payload length first */
2413         hbtype = *p++;
2414         n2s(p, payload);
2415         pl = p;
2416
2417         if (s->msg_callback)
2418                 s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
2419                         &s->s3->rrec.data[0], s->s3->rrec.length,
2420                         s, s->msg_callback_arg);
2421
2422         if (hbtype == TLS1_HB_REQUEST)
2423                 {
2424                 unsigned char *buffer, *bp;
2425                 int r;
2426
2427                 /* Allocate memory for the response, size is 1 bytes
2428                  * message type, plus 2 bytes payload length, plus
2429                  * payload, plus padding
2430                  */
2431                 buffer = OPENSSL_malloc(1 + 2 + payload + padding);
2432                 bp = buffer;
2433
2434                 /* Enter response type, length and copy payload */
2435                 *bp++ = TLS1_HB_RESPONSE;
2436                 s2n(payload, bp);
2437                 memcpy(bp, pl, payload);
```

(Draft) 2014/05/12 17:15:07
Please add a bounds check for the payload, ensuring that the total number of bytes written does not exceed the number of bytes available according to s->s3->rrec.length.
Edit

```
2438
2439                 r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload +
     padding);
2440
```

Lesson #2: review != audit

- Code reviewers are not trained to find complex bugs.
- Few people are paid to audit critical codebases defensively.
- Fewer people are paid to turn vulnerabilities into exploits defensively.
- Offensive industry will routinely do this => huge edge in finding full exploit chains.
- You get what you pay for => we ~~need to fix this~~ are fixing this.

Changes in the OpenSSL team

- Expanded development team (3 FTE* + 12 volunteers)
- Mandatory code reviews
- New security policy
- New release strategy
- New blog :)

*https://www.openssl.org/support/acknowledgments.html

New OpenSSL release today!

- Security updates for 1.0.1/1.0.0./0.9.8
- Fixing 8 security vulnerabilities
- We get a lot of reports from academia & industry
- 5th security release since Heartbleed - this is a good thing!

How can the community help?

- Formal verification of crypto code
  - Hitting < 2^{-64} corner cases with unit testing is difficult.
  - New-ish elliptic curve implementations: P-224, P-256, P-521 - fast and constant-time. But are they correct?
  - Regression testing (again!) for bug attacks and oracle attacks.

How can the community help?

- State machine analysis
  - Very old code, not written with adversarial behaviour in mind
  - Individual reports from different research groups…
  - ... => continuous regression testing?

How can the community help?

- Record/message/ASN.1 object layer fuzzing
  - Some open-source tools already available to help:
    - American Fuzzy Lop
    - Frankencert
- Smarter tools for finding/building exploits

How can the community help?

- Constant-time crypto
  - AES, RSA, P-256 quite well covered across platforms
  - But how about a library for implementing common operations (x = condition ? a : b)?
  - … or a constant-time code generator for field operations?
  - Authenticated encryption is brittle => need new primitives.

Questions?

The OpenSSL development team:

Matt Caswell, Mark J. Cox, Viktor Dukhovni, Steve Henson, Tim Hudson, Lutz Jänicke, **Emilia Käsper**, **Ben Laurie**, Richard Levitte, Steve Marquess, Bodo Möller, **Andy Polyakov**, Kurt Roeckx, Rich Salz, Geoff Thorpe

Come talk to us!