LibreSSL

Birth of LibreSSL and its current status

Frank Timmers

Consutant, Snow B.V.

Background

What is LibreSSL

- A fork of OpenSSL 1.0.1g
- Being worked on extensively by a number of OpenBSD developers

What is OpenSSL

- OpenSSL is an open source SSL/TLS crypto library
- Currently the de facto standard for many servers and clients
- Used for securing http, smtp, imap and many others

Alternatives

- Netscape Security Services (NSS)
- BoringSSL
- GnuTLS

What is Heartbleed

- Heartbleed was a bug leaking of private data (keys) from both client and server
- At this moment known as “the worst bug ever”
- Heartbeat code for DTLS over UDP
- So why was this also included in the TCP code?
- Not the reason to create a fork

Why did this happen

- Nobody looked
- Or at least didn’t admit they looked

Why did nobody look

- The code is horrible
- Those who did look, quickly looked away and hoped upstream could deal with it

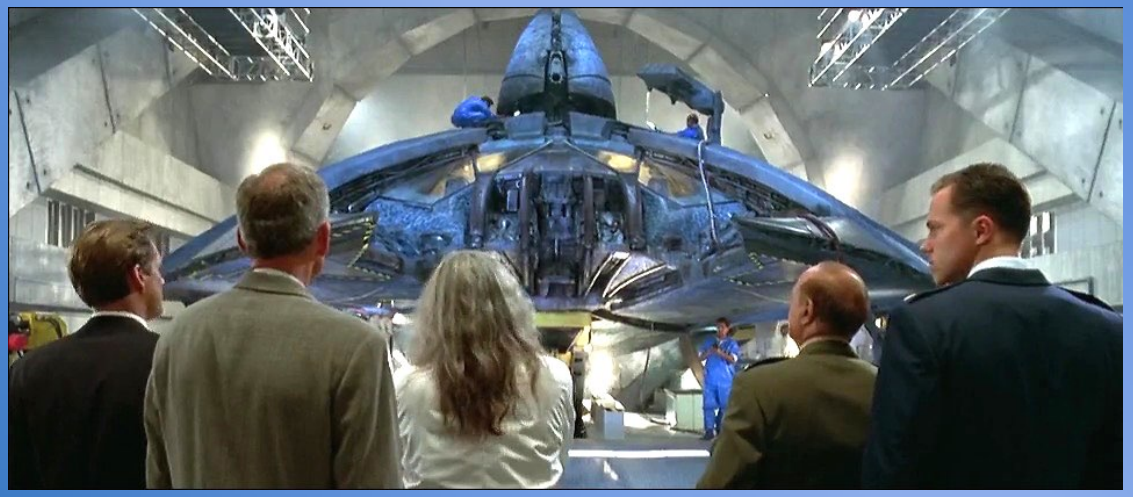
Why was the code so horrible

- Buggy re-implementations of standard libc functions like random() and malloc()
- Forces all platforms to use these buggy implementations
- Nested #ifdef, #ifndefs (up to 17 layers deep) through out the code
- Written in “OpenSSL C”, basically their own dialect
- Everything on by default

Why was it so horrible?

crypto_malloc

- Never frees memory (Tools like Valgrind, Coverity can’t spot bugs)
- Used LIFO recycling (Use after free?)
- Included debug malloc by default, logging private data
- Included the ability to replace malloc/free at runtime



#ifdef trees

- #ifdef, #elif, #else trees up to 17 layers deep
- Throughout the complete source
- Some of which could never be reached
- Hard to see what is or not compiled in

1. #ifdef OPENSSL_WINDOWS
2. #elif defined(OPENSSL_POSIX)
3. #elif defined(OPENSSL_OSX)
4. #elif defined(OPENSSL_VMS)
5. # ifndef OPENSSL_POSIX
6. # else
7. #else
8. #endif

Everything on by default

```
#ifndef OPENSSL_NO_CAMELLIA
#ifndef OPENSSL_NO_CAPIENG
#ifndef OPENSSL_NO_CAST
#ifndef OPENSSL_NO_CMS
#ifndef OPENSSL_NO_COMP
#ifndef OPENSSL_NO_DEPRECATED
#ifndef OPENSSL_NO_DES
#ifndef OPENSSL_NO_DES
#ifndef OPENSSL_NO_DESCBCM
#ifndef OPENSSL_NO_DH
#ifndef OPENSSL_NO_DSA
#ifndef OPENSSL_NO_DTLS1
```

Other examples

- Support for Big Endian amd64 support
- Compiler options NO_OLD_ASN1 and NO_ASN1_OLD
- Backward compatibility for a mistake which was fixed within a month 14 years ago
- Char buf[288+1], tmp[20], str[128+1];
- static const char rnd_seed[] = "**string to make the random number generator think it has entropy**"
- malloc(items*size) -> reallocarray(items, size)
- Socklen_t

So why a fork

- Buggy re-implementations of standard libc functions like random() and malloc()
- Forces all platforms to use these buggy implementations
- Nested #ifdef, #ifndefs (up to 17 layers deep) through out the code
- Written in “OpenSSL C”, basically their own dialect
- Everything on by default

- Serious bug report sitting on RT for 4 years with one liner fix
- Fixes provided to the upstream do not get merged

Who is to blame

OpenSSL ?

Everyone is guilty

OpenSSL is Open Source, used by many vendors

- OpenBSD
- FreeBSD
- Linux (Redhat/Debian/Ubuntu/etc)
- WindRiver
- HP-UX / AIX / Solaris
- Cisco / Juniper / F5 and other appliance manufacturers
- Microsoft

=> All had access to the source

Everyone is guilty

All had access, but all ran away

LibreSSL the first 30 days

- Fix CRYPTO_malloc
- OpenSSL 1.0.1g was 388,000 lines code
- Removed 90,000 lines of C, about 150.000 lines from all source files
- The unidiff between OpenSSL and LibreSSL aprox 500,000 lines
- Many bug fixed
- Start KNFing the whole thing (man 9 style)
- More readable code, but some scary parts still remain

MAN 9 style

```
STYLE(9)                    OpenBSD Kernel Manual                    STYLE(9)

NAME
     style - Kernel source file style guide (KNF)

DESCRIPTION
     This file specifies the preferred style for kernel source files in the
     OpenBSD source tree.  It is also a guide for preferred user land code
     style.  These guidelines should be followed for all new code.  In
     general, code can be considered ``new code'' when it makes up about 50%
     or more of the file(s) involved.  This is enough to break precedents in
     the existing code and use the current style guidelines.

           /*
            * Style guide for the OpenBSD KNF (Kernel Normal Form).
            */

           /*
            * VERY important single-line comments look like this.
            */

           /* Most single-line comments look like this. */

           /*
            * Multi-line comments look like this.  Make them real sentences.
            * Fill them so they look like real paragraphs.
            */

     Kernel include files (i.e., <sys/*.h>) come first; normally, you'll need
     <sys/types.h> OR <sys/param.h>, but not both!  <sys/types.h> includes
     <sys/cdefs.h>, and it's okay to depend on that.

           #include <sys/types.h>  /* Non-local includes in brackets. */

     If it's a network program, put the network include files next.
```

LibreSSL current state

- Removed even more obsolete code
  - DOS
  - Win16 and other obsolete windows flavors
  - MacOS Classic (Pre OSX)
  - Obscure things you've never heard about
  - Etc. etc. etc
- More code cleanup and KNFing
- More bug fixing (OpenSSL's RT remains a valuable resource)
- Mostly stopped deleting code
- Replaced OpenSSL in OpenBSD 5.6, released 1 Nov 2014
- Replaced OpenSSL in OpenELEC 5.0, released 28 Dec 2014
- H2O HTTP Server 1.2.0 now bundles LibreSSL by default

LibreSSL current state

- Even added some new features (crypto)
  - Brainpool
  - ChaCha
  - Poly1305
  - ANSSI FRP256v1
  - Several new cypher suites based on the above

- Current release 2.1.6, released March 19, 2015
- Put back GHOST and Camellia cipher suite (reworked)
- Initial support for 32 and 64 bit Windows
- Ciphers now default to TLS1.2

LibreSSL Future

- More code cleanup
- With easier to read code, get more developer involvement
- Bug fixes, modern coding practices and standards
- Split libcrypto from libssl
- Do portability right

Portability

How OpenSSL does portability

- Assume the OS provides nothing
- Mazes of #ifdef #ifndef horror
- Own implementations of layers and force all platforms to use it (CRYPTO_malloc, CRYPTO_realloc, BIO_snprintf, OPENSSL_*)
- Assume the world is stuck in 1989

How OpenBSD does portability

- Assume a sane target OS (POSIX, like OpenBSD) – code to that standard.
- Build and maintain code on the above, using modern C
- Provide Portability shims to correctly do things that other OS's don't provide, only for those that need it.

Application Programming Interfaces

- All OpenSSL functions are exposed to the public API and include files
- API's like BIO_snprintf, CRYPTO_malloc can currently not be removed
- Internal library functions now do not use these anymore
- Normal POSIX API: easier and more developer involvement
- Preserve API compatibility with OpenSSL for now
- API will change in the future

Application Programming Interfaces

- New APIs for loading CA keychain and certificates
- Ciphers now default to TLS1.2

Questions


The End