LibreSSL

Giovanni Bechis
giovanni@openbsd.org

Università degli studi di Udine,
Nov 29, 2014
About Me

- sys admin and developer @ SNB
- OpenBSD developer
- Open Source developer in several other projects
How the Heartbleed bug works:

SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).

Sees in car why". Note: Files for 18-31.341 at 22:17 are in /tmp/files-3613. User Meg wants these 4 letters: BIRD. There are currently 614 connections open. User Brendan uploaded the file.

HMM...

BIRD

SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).

User Meg wants these 500 letters: "Hat". Lucas requests the "issued connections" page. Eve (administrator) wants to set server's master key to "1618-350-3653". Isabel wants pages about snakes but not too long. User Karen wants it.

Hat: Lucas requests the "issued connections" page. Eve (administrator) wants to set server's master key to "1618-350-3653". Isabel wants pages about snakes but not too long. User Karen wants it.
why Heartbleed happened?

- horrible code, once you look at it, you’ll go somewhere else
- developers only interested in adding features, not fixing code
- fixes are not merged upstream
- bugs (and fixes) sleep for years in the bug tracker
OpenSSL malloc replacement

- it never frees memory (tools cannot spot bugs)
- it uses LIFO recycling (no 'use after free' problems)
- includes a debugging malloc that send private info to logs
- includes the ability to replace the malloc/free at runtime
What’s wrong with OpenSSL code?

- quite all OpenSSL API headers are public
- it is developed using “OpenSSL C”
- it uses its own functions instead of those provided by libc
  - BIO_free(3), BIO_strdup
- crazy compile options (in OpenSSL both NO_OLD_ASN1 and NO_ASN1_OLD compile options are present but their meaning is slightly different)
#include "des_locl.h"

/* HAS BUGS! DON’T USE - this is only present for use in des.c */
void DES_3cbc_encrypt(DES_cblock *input, DES_cblock *output, long length,
                     DES_key_schedule ks1, DES_key_schedule ks2, DES_cblock *iv1,
                     DES_cblock *iv2, int enc)
Index: e_aes.c
===================================================================
RCS file: /cvs/src/lib/libssl/src/crypto/evp/e_aes.c,v
retrieving revision 1.13
retrieving revision 1.14
diff -u -p -u -p -r1.13 -r1.14
--- e_aes.c 8 May 2014 15:13:06 -0000 1.13
+++ e_aes.c 8 May 2014 15:29:00 -0000 1.14
@@ -56,7 +56,6 @@
 #include <assert.h>
 #include <openssl/aes.h>
 #include "evp_locl.h"
-#ifndef OPENSSL_FIPS
 #include "modes.locl.h"
-#ifndef OPENSSL_FIPS
 #include "openssl/rand.h"
@@ -692,11 +691,6 @@
 case EVP_CTRL_GCM_SET_IVLEN:
     if (arg <= 0)
         return 0;
-#ifdef OPENSSL_FIPS
- if (FIPS_module_mode() &&
- !(c->flags & EVP_CIPH_FLAG_NON_FIPS_ALLOW) && arg < 12)
-     return 0;
-#endif

@@ -692,11 +691,6 @@
 case EVP_CTRL_GCM_SET_IVLEN:
     if (arg <= 0)
         return 0;
-#ifdef OPENSSL_FIPS
- if (FIPS_module_mode() &&
- !(c->flags & EVP_CIPH_FLAG_NON_FIPS_ALLOW) && arg < 12)
-     return 0;
-#endif

openssl code
RCS file: /cvs/src/lib/libssl/src/crypto/sha/sha512.c,v
retrieving revision 1.2
retrieving revision 1.3
diff -u -r1.2 -r1.3
--- src/lib/libssl/src/crypto/sha/sha512.c 2013/12/19 22:09:26 1.2
+++ src/lib/libssl/src/crypto/sha/sha512.c 2014/04/17 21:07:05 1.3
@@ -318,13 +318,11 @@
 : "=r"(ret) \ 
 : "J"(n),"0"(a) \ 
 : "cc"); ret; })
-# if !defined(B_ENDIAN)
+# elif (defined(__i386) || defined(__i386__))
    # define PULL64(x) ({ SHA_LONG64 ret=*((const SHA_LONG64 *)(&(x))); \ 
                  asm ("bswapq %0" \ 
                       : "=r"(ret) \ 
                       : "0"(ret)); ret; })
-# endif
-# elif (defined(__i386) || defined(__i386__)) && !defined(B_ENDIAN)
+# elif (defined(__i386) || defined(__i386__))
    # if defined(I386_ONLY)
    # define PULL64(x) ({ const unsigned int *p=(const unsigned int *)&(x));\ 
                        unsigned int hi=p[0],lo=p[1];
                        asm ("bswapq %0" \ 
                             : "=r"(ret) \ 
                             : "0"(ret)); ret; })
    # endif
    #endif
static struct hostent *GetHostByName(char *name);
static int ssl_sock_init(void);
static int init_server(int *sock, int port, int type);
static int init_server_long(int *sock, int port, char *ip, int type);

--- apps/s_socket.c 19 Apr 2014 13:13:01 -0000 1.31
+++ apps/s_socket.c 19 Apr 2014 16:38:04 -0000 1.32
@@ -77,7 +77,6 @@

h2 = gethostbyname(*host);
if (h2 == NULL) {
    BIO_printf(bio_err, "gethostbyname failure\n");
LibreSSL

- very young project, development started on April 2014
- mostly developed by OpenBSD
- forked from OpenSSL 1.0.1g
LibreSSL goals

- preserve API/ABI compatibility with OpenSSL
- bring more people into working with the codebase (+KNF, -#ifdef)
- fix bugs asap, use modern coding practices
- do portability the right way™
Some differences between LibreSSL and OpenSSL

- 90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9
- does not add FIPS support
- different set of ciphers (-MD2, -SRP, +ChaCha, +poly1305)
- BIO_* functions do the right thing™
- malloc(x*y) has been converted to reallocarray(x,y)
How OpenSSL does portable

- use and abuse of internal functions that behaves "more or less" the same as libc counterpart
- #ifdef and #ifndef everywhere
- support for as many combinations of operating systems and compilers out there
How OpenSSH (and LibreSSL) does portable

- assume a sane target OS (OpenBSD) and code with his standards
- build and maintain code on the main target OS, using modern C
- provide portability code only to provide functions that other OS’s don’t provide
- do not reimplement libc
- put as few #ifdefs as possible in the code
if (inet_pton(AF_INET, host, &addrbuf) != 1 &&
    inet_pton(AF_INET6, host, &addrbuf) != 1) {
    if (SSL_set_tlsext_host_name(ssl, host) == 0) {
        ERR_print_errors_fp(ttyout);
        goto cleanup_url_get;
    }
}
if (SSL_connect(ssl) <= 0) {
    ERR_print_errors_fp(ttyout);
    if (ressl_connect_socket(ssl, s, host) != 0) {
        fprintf(ttyout, "SSL failure: \%s\n", resssl_error(ssl));
        goto cleanup_url_get;
    }
} else if (ssl_verify) {
    X509    *cert;
    cert = SSL_get_peer_certificate(ssl);
    if (cert == NULL) {
        fprintf(ttyout, "%s: no server certificate\n", getprogname());
        goto cleanup_url_get;
    }
    if (ssl_check_hostname(cert, host) != 0) {
        X509_free(cert);
        fprintf(ttyout, "%s: host '%s' not present in" 
            " server certificate\n", getprogname(), host);
        goto cleanup_url_get;
    }
    X509_free(cert);
What should we have learned?

- fix bugs ASAP, do not let them sleep for years
- use as few workarounds as possible in your code
- review your code
- look at your old code and fix horrors sleeping there
- do not reinvent the wheel
Thanks