LibreSSL

Giovanni Bechis
<giovanni@openbsd.org>

CeBIT 2015

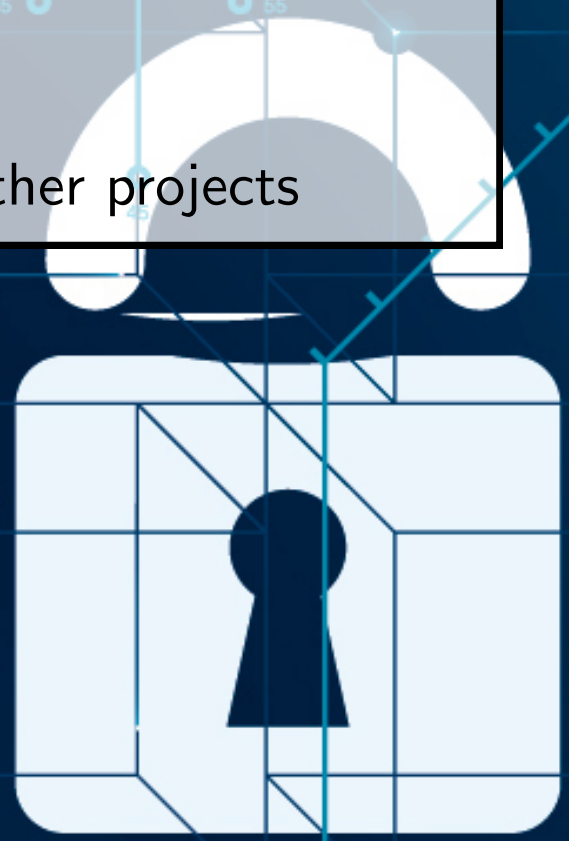
SNB Information Technology & Web Solutions

About Me

- sys admin and developer @SNB

About Me

- sys admin and developer @SNB
- OpenBSD developer
- Open Source developer in several other projects



CVE-2014-0160

- 17% of https web servers use OpenSSL as SSL/TLS library and have heartbeat extension enabled

CVE-2014-0160

- 17% of https web servers use OpenSSL as SSL/TLS library and have heartbeat extension enabled
- at least Cisco, Fortinet, Oracle and Siemens products has been affected

CVE-2014-0160

- 17% of https web servers use OpenSSL as SSL/TLS library and have heartbeat extension enabled
- at least Cisco, Fortinet, Oracle and Siemens products has been affected
- bug was introduced on December 2011 and fixed on April 2014

CVE-2014-0160

- 17% of https web servers use OpenSSL as SSL/TLS library and have heartbeat extension enabled
- at least Cisco, Fortinet, Oracle and Siemens products has been affected
- bug was introduced on December 2011 and fixed on April 2014
- exploitation of this bug does not leave any trace

CVE-2014-0160

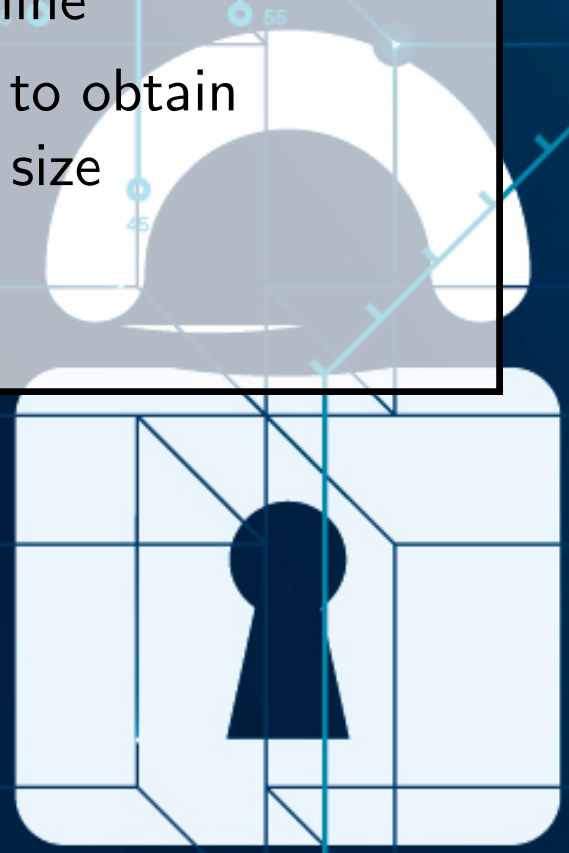
- 17% of https web servers use OpenSSL as SSL/TLS library and have heartbeat extension enabled
- at least Cisco, Fortinet, Oracle and Siemens products has been affected
- bug was introduced on December 2011 and fixed on April 2014
- exploitation of this bug does not leave any trace
- eWEEK estimated a total cost of $500 million

How the Heartbleed bug works:

- attacker can dump up to 64k of memory near the SSL heartbeat of the attacked machine

How the Heartbleed bug works:

- attacker can dump up to 64k of memory near the SSL heartbeat of the attacked machine
- attack can be repeated many times to obtain different memory allocations of 64k size

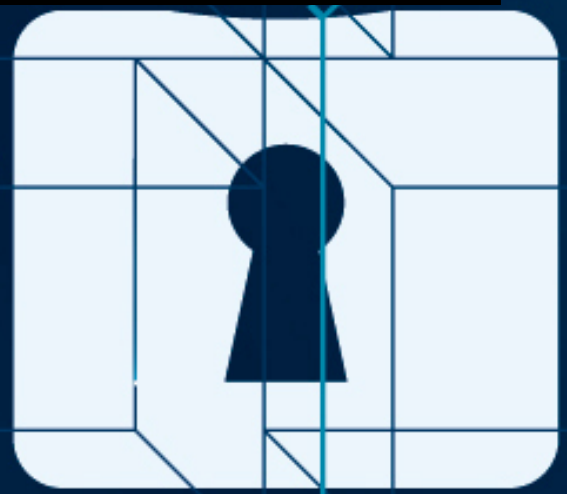


How the Heartbleed bug works:

- attacker can dump up to 64k of memory near the SSL heartbeat of the attacked machine
- attack can be repeated many times to obtain different memory allocations of 64k size
- memory stolen could reveal any kind of data: passwords, credit card numbers, personal data, ...

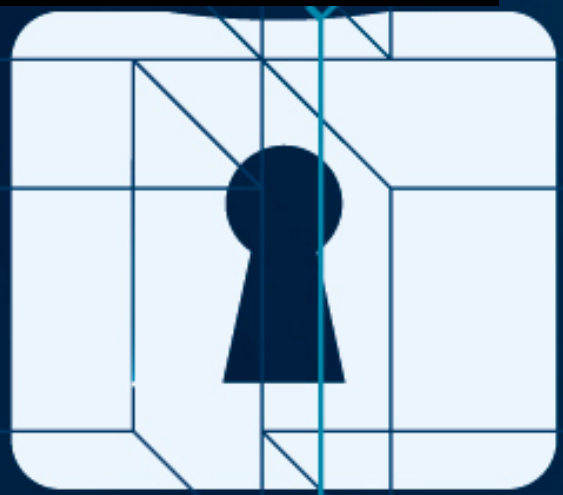
Why Heartbleed happened ?

- code too complex and intricated



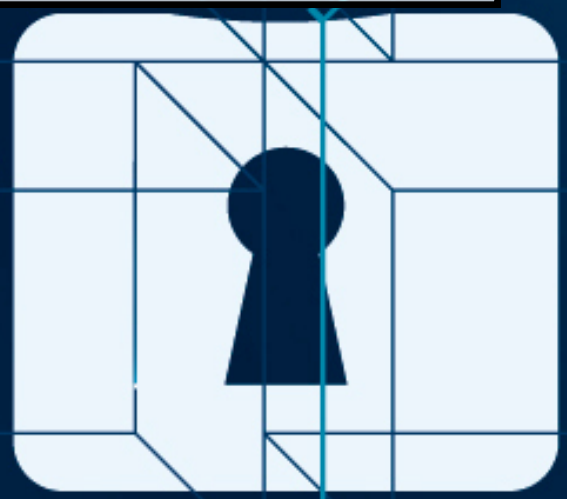
Why Heartbleed happened ?

- code too complex and intricated
- developers mostly interested in adding features, not fixing code



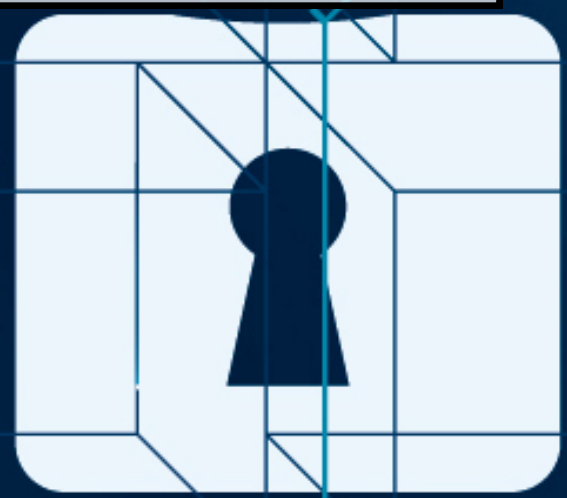
Why Heartbleed happened ?

- code too complex and intricated
- developers mostly interested in adding features, not fixing code
- fixes not merged upstream



Why Heartbleed happened ?

- code too complex and intricated
- developers mostly interested in adding features, not fixing code
- fixes not merged upstream
- bugs (and fixes) sleep for years in the bug tracker



OpenSSL malloc replacement

- it never frees memory (tools cannot spot bugs)

OpenSSL malloc replacement

- it never frees memory (tools cannot spot bugs)
- it uses LIFO recycling (no 'use after free' problems)

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

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

What's wrong with OpenSSL code ?

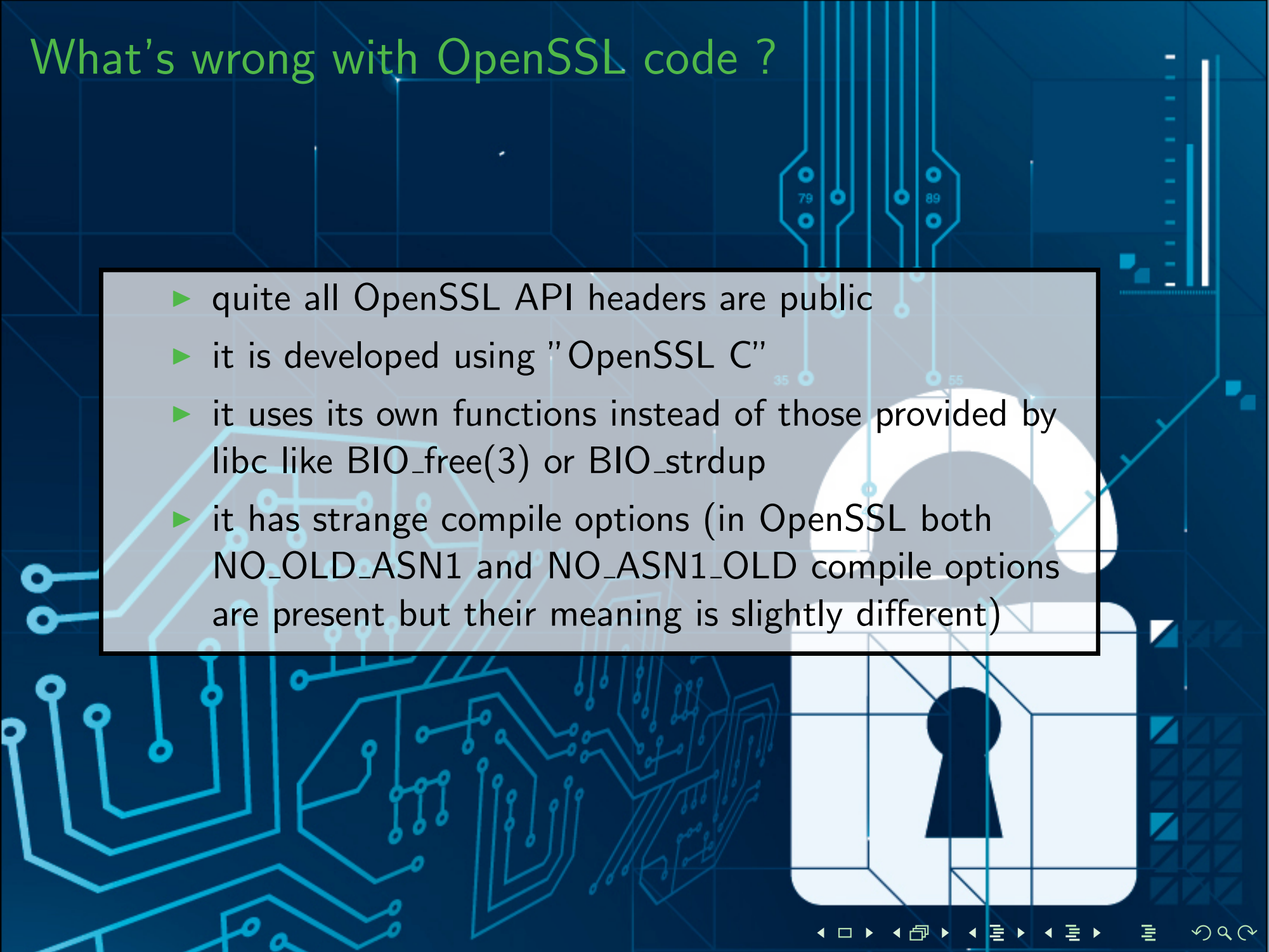
- quite all OpenSSL API headers are public
- it is developed using "OpenSSL C"

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 like BIO_free(3) or BIO_strdup

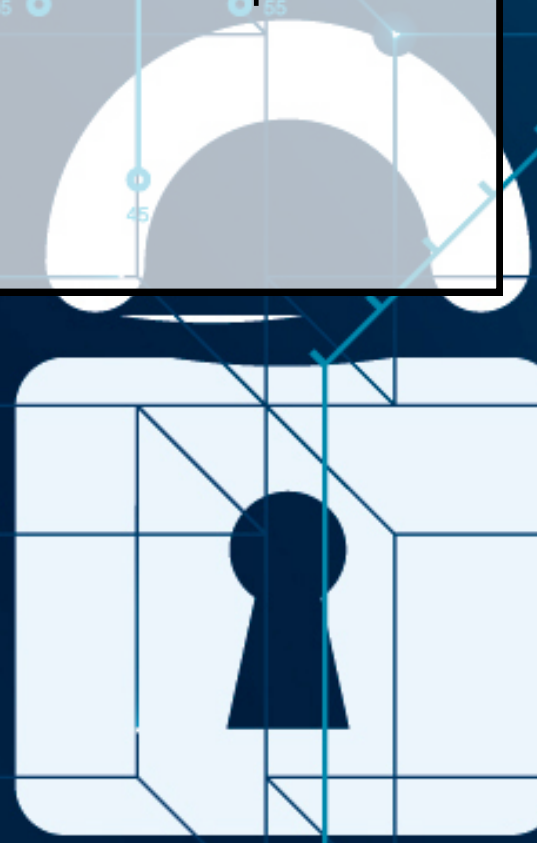
What's wrong with OpenSSL code ?

- quite all OpenSSL API headers are public
- it is developed using "OpenSSL C"
- it uses its own functions instead of those provided by libc like BIO_free(3) or BIO_strdup
- it has strange compile options (in OpenSSL both NO_OLD_ASN1 and NO_ASN1_OLD compile options are present but their meaning is slightly different)



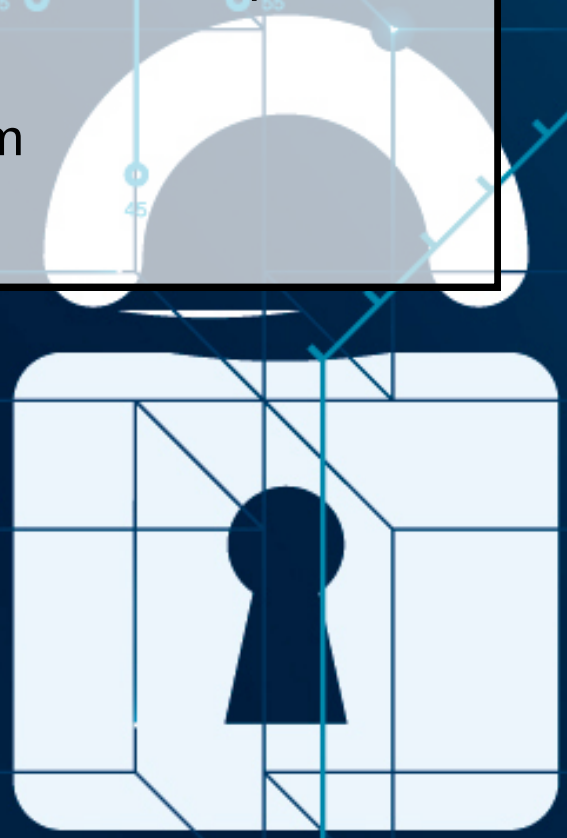
LibreSSL

- very young project, development started on April 2014



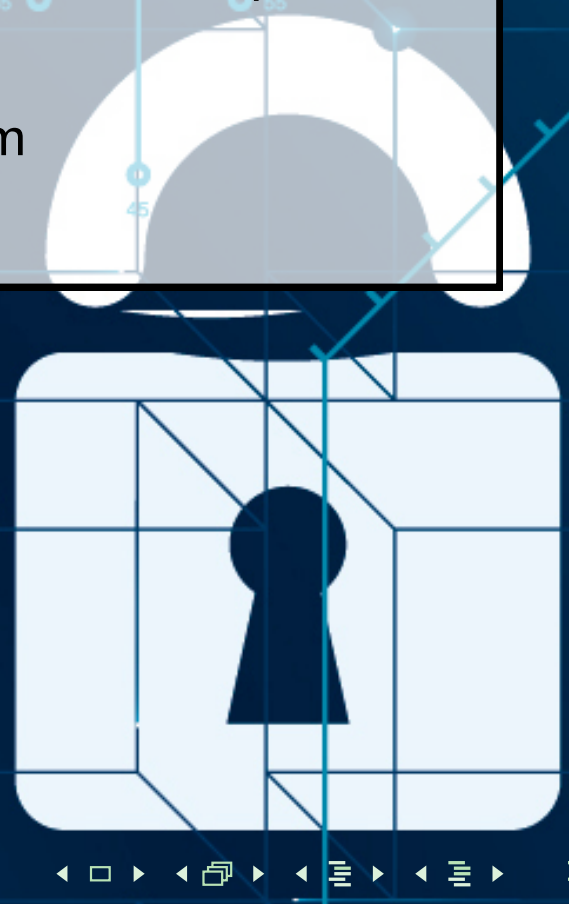
LibreSSL

- very young project, development started on April 2014
- mostly developed by OpenBSD team



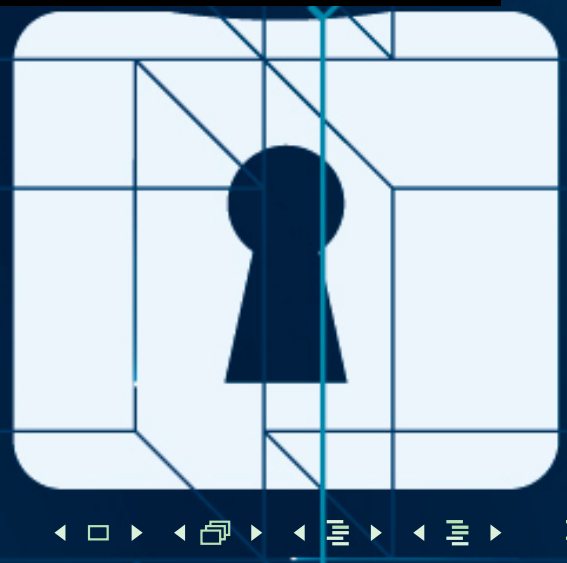
LibreSSL

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



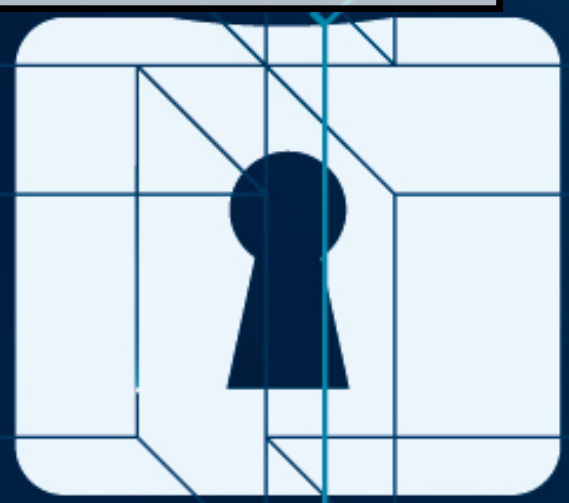
LibreSSL goals

- preserve API/ABI compatibility with OpenSSL



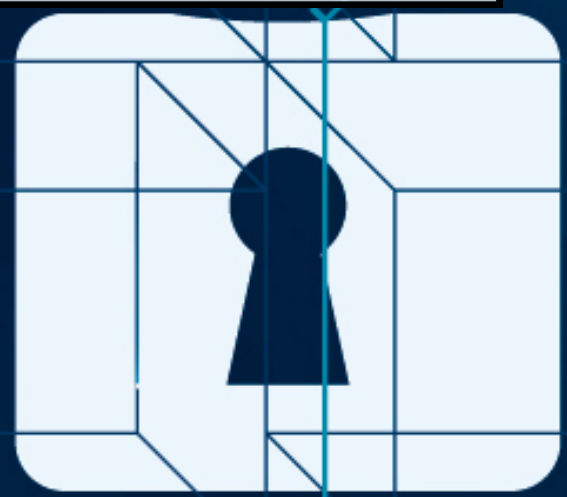
LibreSSL goals

- preserve API/ABI compatibility with OpenSSL
- bring more people into working with the codebase (+KNF, -#ifdef)



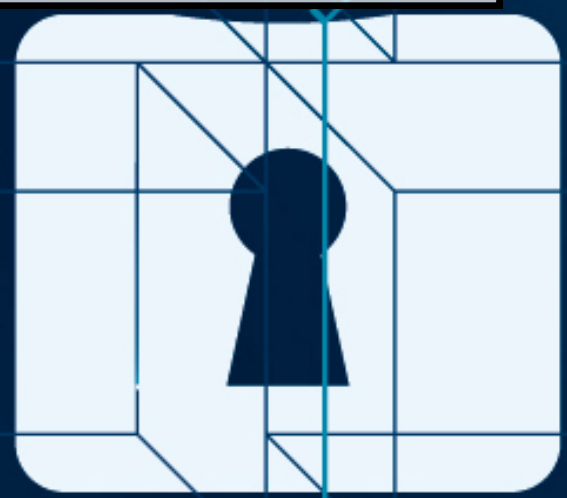
LibreSSL goals

- preserve API/ABI compatibility with OpenSSL
- bring more people into working with the codebase (+KNF, -#ifdef)
- fix bugs asap, use modern coding practices



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

Some differences between LibreSSL and OpenSSL

- ∼90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9

Some differences between LibreSSL and OpenSSL

- ~90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9
- does not have FIPS support

Some differences between LibreSSL and OpenSSL

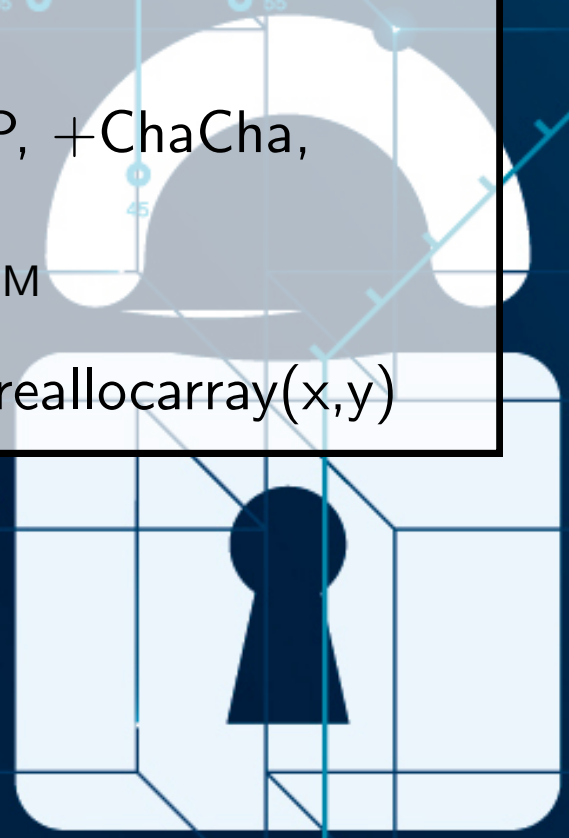
- ∼90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9
- does not have FIPS support
- different set of ciphers (-MD2, -SRP, +ChaCha, +poly1305)

Some differences between LibreSSL and OpenSSL

- ∼90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9
- does not have FIPS support
- different set of ciphers (-MD2, -SRP, +ChaCha, +poly1305)
- BIO_* functions do the right thing^{TM}

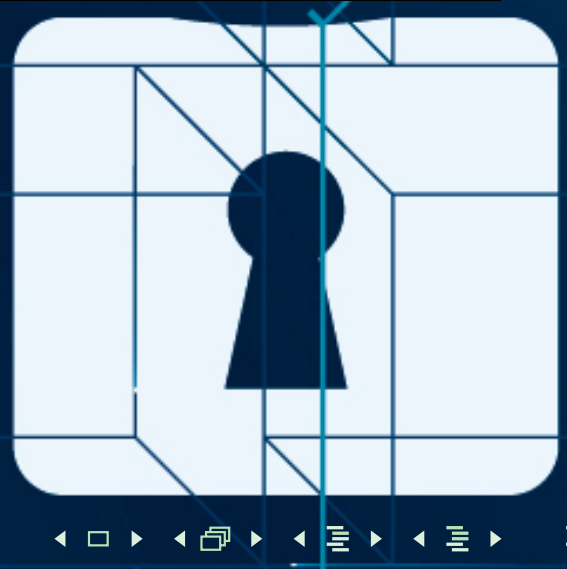
Some differences between LibreSSL and OpenSSL

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



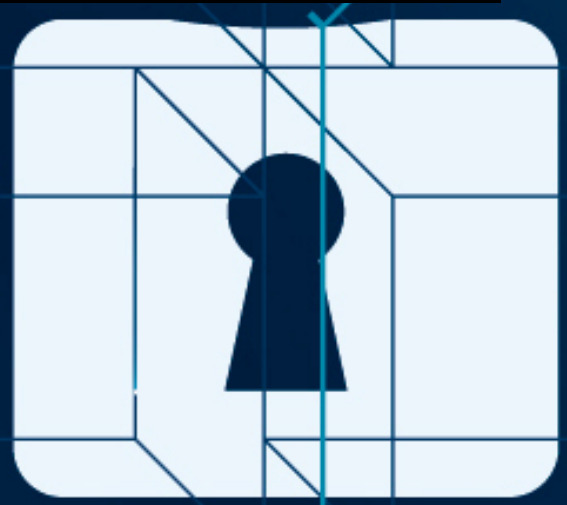
How OpenSSL does portable

- use and abuse of internal functions that behaves "more or less" the same as libc counterpart



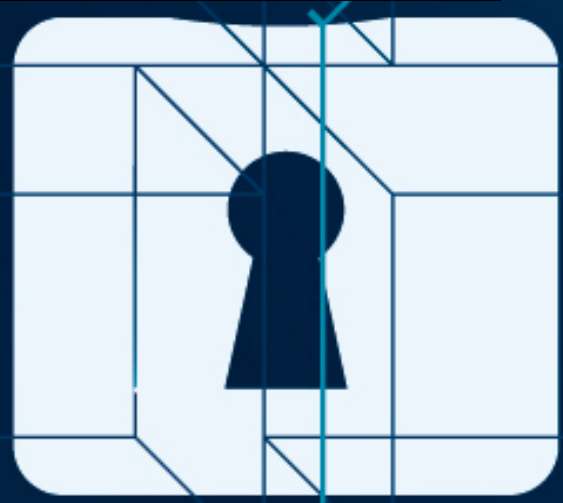
How OpenSSL does portable

- use and abuse of internal functions that behaves "more or less" the same as libc counterpart
- #ifdef and #ifndef everywhere



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

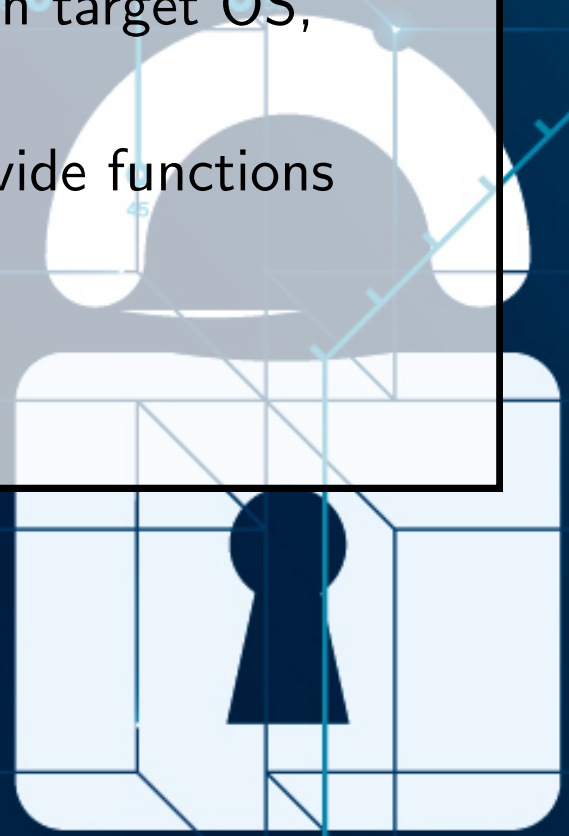
- assume a sane target OS (OpenBSD) and code with his standards

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

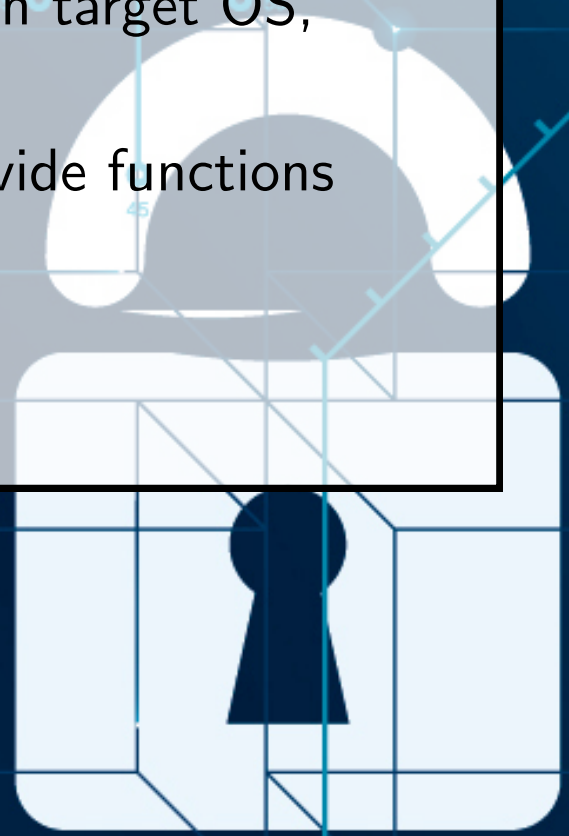
How OpenSSH (and LibreSSL) does portable

- 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



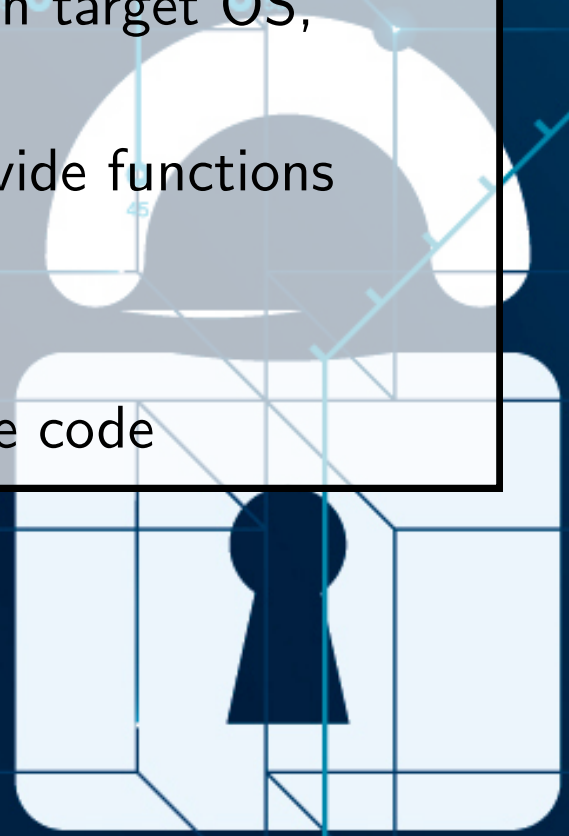
How OpenSSH (and LibreSSL) does portable

- 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



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



LibreSSL API, ftp client

```
-               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", ressl_error(ssl));
                        goto cleanup_url_get;
                }
-               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 fo years

What should we have learned ?

- fix bugs ASAP, do not let them sleep fo years
- use as few workarounds as possible in your code

What should we have learned ?

- fix bugs ASAP, do not let them sleep fo years
- use as few workarounds as possible in your code
- review your code

What should we have learned ?

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

What should we have learned ?

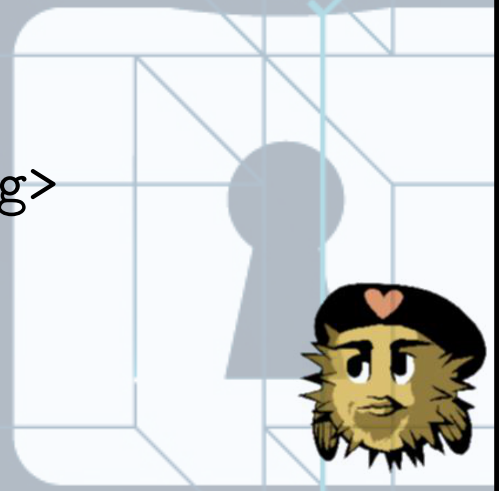
- fix bugs ASAP, do not let them sleep fo years
- use as few workarounds as possible in your code
- review your code
- look at your old code and fix horrors sleeping there
- do not reinvent the wheel

What should we have learned ?

- fix bugs ASAP, do not let them sleep fo 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
- every bug could be a security bug

Questions ?

Giovanni Bechis
<giovanni@openbsd.org>



SNB Information Technology & Web Solutions