2020/06/24
ONLINE / INNOQ Technology Lunch

OpenSSL

Was Du schon immer darüber wissen wolltest

INNOQ

CHRISTOPH ISERLOHN
SENIOR CONSULTANT

Was Du schon immer über OpenSSL wissen wolltest?

Alles? In knapp 45 Minuten?

SEE ALSO

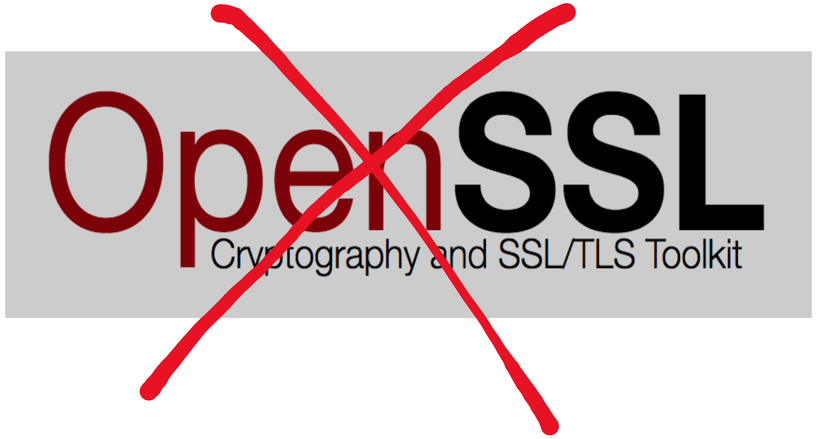
openssl(1), crypto(7), CRYPTO_get_ex_new_index(3), SSL_accept(3), SSL_clear(3),
SSL_connect(3), SSL_CIPHER_get_name(3), SSL_COMP_add_compression_method(3),
SSL_CTX_add_extra_chain_cert(3), SSL_CTX_add_session(3), SSL_CTX_ctrl(3),
SSL_CTX_flush_sessions(3), SSL_CTX_get_verify_mode(3), SSL_CTX_load_verify_locations(3)
SSL_CTX_new(3), SSL_CTX_sess_number(3), SSL_CTX_sess_set_cache_size(3),
SSL_CTX_sess_set_get_cb(3), SSL_CTX_sessions(3), SSL_CTX_set_cert_store(3),
SSL_CTX_set_cert_verify_callback(3), SSL_CTX_set_cipher_list(3), SSL_CTX_set_client_CA_list(3),
SSL_CTX_set_client_cert_cb(3), SSL_CTX_set_default_passwd_cb(3),
SSL_CTX_set_generate_session_id(3), SSL_CTX_set_info_callback(3),
SSL_CTX_set_max_cert_list(3), SSL_CTX_set_mode(3), SSL_CTX_set_msg_callback(3),
SSL_CTX_set_options(3), SSL_CTX_set_quiet_shutdown(3), SSL_CTX_set_read_ahead(3),
SSL_CTX_set_security_level(3), SSL_CTX_set_session_cache_mode(3),
SSL_CTX_set_session_id_context(3), SSL_CTX_set_ssl_version(3), SSL_CTX_set_timeout(3),
SSL_CTX_set_tmp_dh_callback(3), SSL_CTX_set_verify(3), SSL_CTX_use_certificate(3),
SSL_alert_type_string(3), SSL_do_handshake(3), SSL_enable_ct(3), SSL_get_SSL_CTX(3),
SSL_get_ciphers(3), SSL_get_client_CA_list(3), SSL_get_default_timeout(3), SSL_get_error(3),
SSL_get_ex_data_X509_STORE_CTX_idx(3), SSL_get_fd(3), SSL_get_peer_cert_chain(3),
SSL_get_rbio(3), SSL_get_session(3), SSL_get_verify_result(3), SSL_get_version(3),
SSL_load_client_CA_file(3), SSL_new(3), SSL_pending(3), SSL_read_ex(3), SSL_read(3),
SSL_rstate_string(3), SSL_session_reused(3), SSL_set_bio(3), SSL_set_connect_state(3),
SSL_set_fd(3), SSL_set_session(3), SSL_set_shutdown(3), SSL_shutdown(3), SSL_state_string(3),
SSL_want(3), SSL_write_ex(3), SSL_write(3), SSL_SESSION_free(3), SSL_SESSION_get_time(3),
d2i_SSL_SESSION(3), SSL_CTX_set_psk_client_callback(3), SSL_CTX_use_psk_identity_hint(3),
SSL_get_psk_identity(3), DTLSv1_listen(3)

~~Was Du schon immer über OpenSSL wissen wolltest?~~

Was ich denke, dass Du schon immer über OpenSSL wissen solltest


OpenBSD 6.5


SSH...
OpenSSH
KEEPING YOUR COMMUNIQUÉS SECRET
TOP SECRET


OpenSSL
Cryptography and SSL/TLS Toolkit

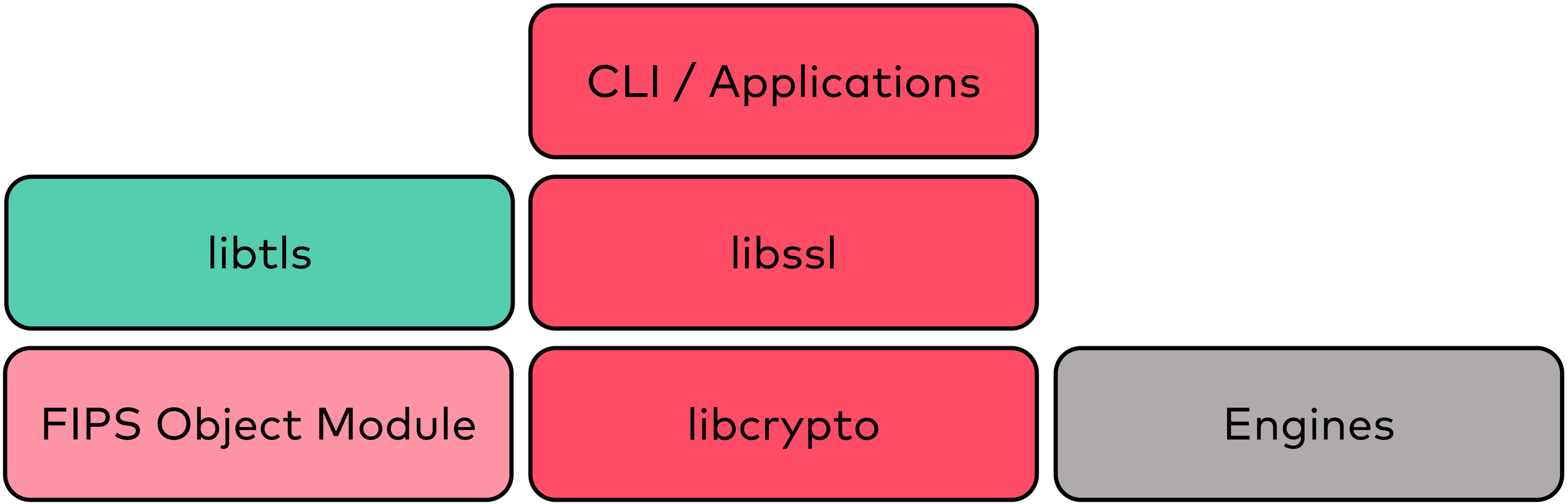

LibreSSL

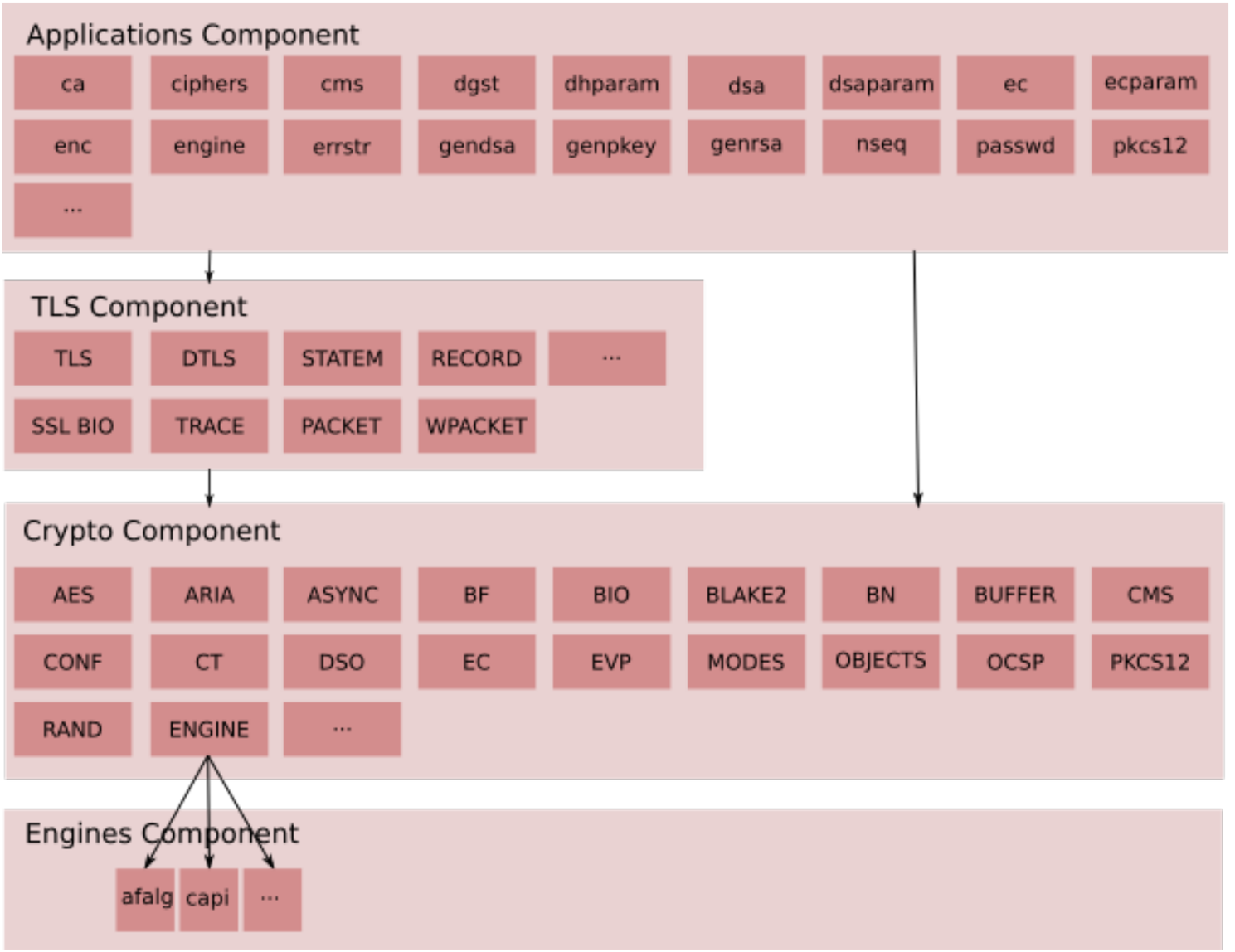
Was ist OpenSSL?

Eine der wichtigsten Software-Bibliotheken überhaupt

Was ist OpenSSL?

- Command line tool
  - **Zertifikatsmanagement: erstellen / signieren / validieren / CSRs erstellen**
  - **Kryptografische Funktionen: Veschlüsseln / Entschlüsseln / Signieren / Signaturprüfung**
- Eine Bibliothek für die Client- und die Server-Seite einer TLS Verbindung
- Eine Bibliothek für kryptografische Funktionen
- Eine Schnittstelle für die kryptografischen Funktionen eines HSM
- Eine halbe PKI


CLI / Applications
libtls
libssl
FIPS Object Module
libcrypto
Engines



https://www.openssl.org/docs/images/AsIsComponent.png

Provokante Analyse: "PGP ist schlecht und sollte verschwinden"

Lesetipp 19.07.2019 09:49 Uhr – Jürgen Schmidt

(Bild: EFAIL-Logo: Jana Runde und Zuzana Somorovska)

Eine Gruppe von Security-Experten erklärt die grundsätzlichen Probleme mit PGP und gibt Tipps für Alternativen im Alltag.

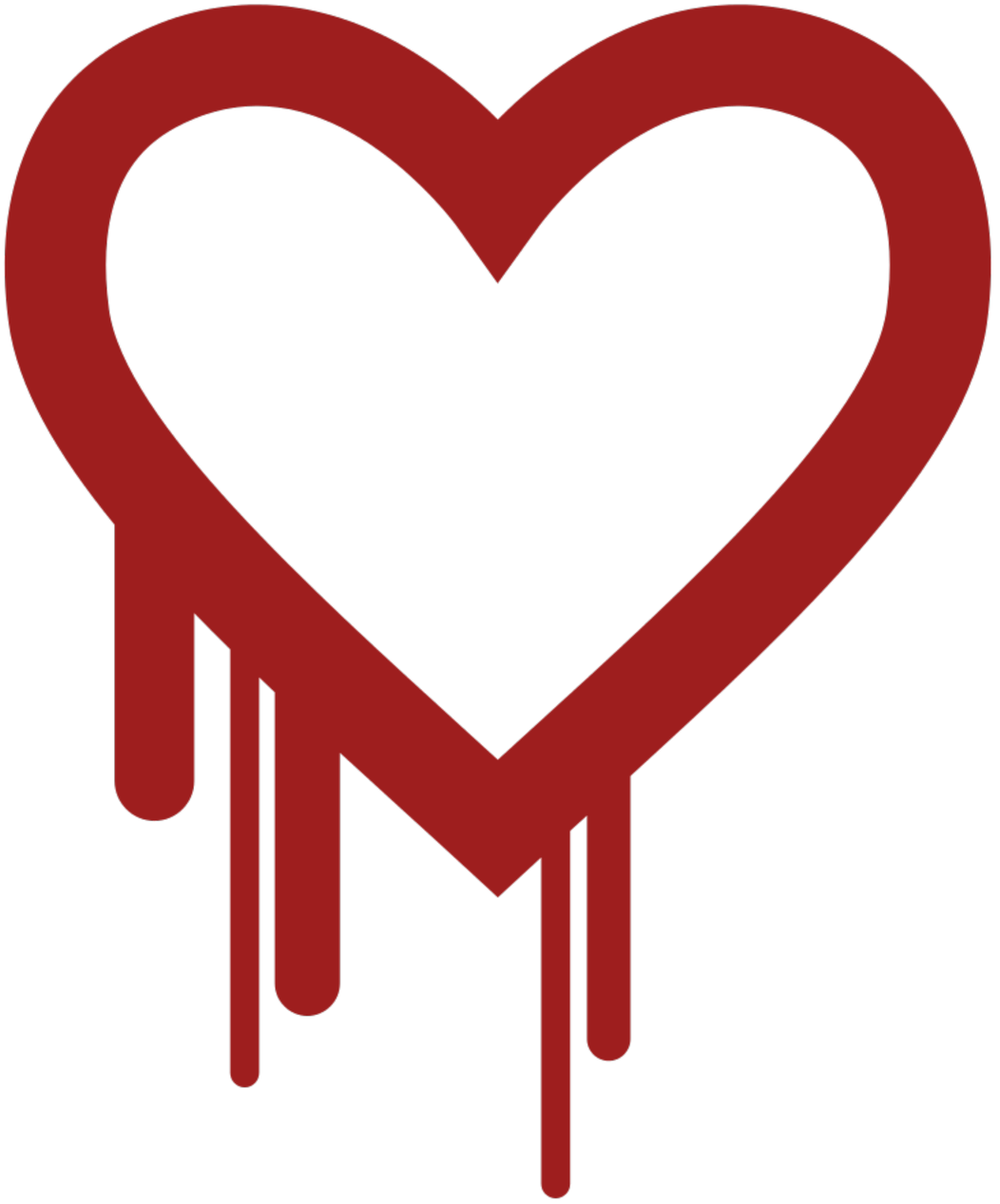
Das OpenSSL Problem

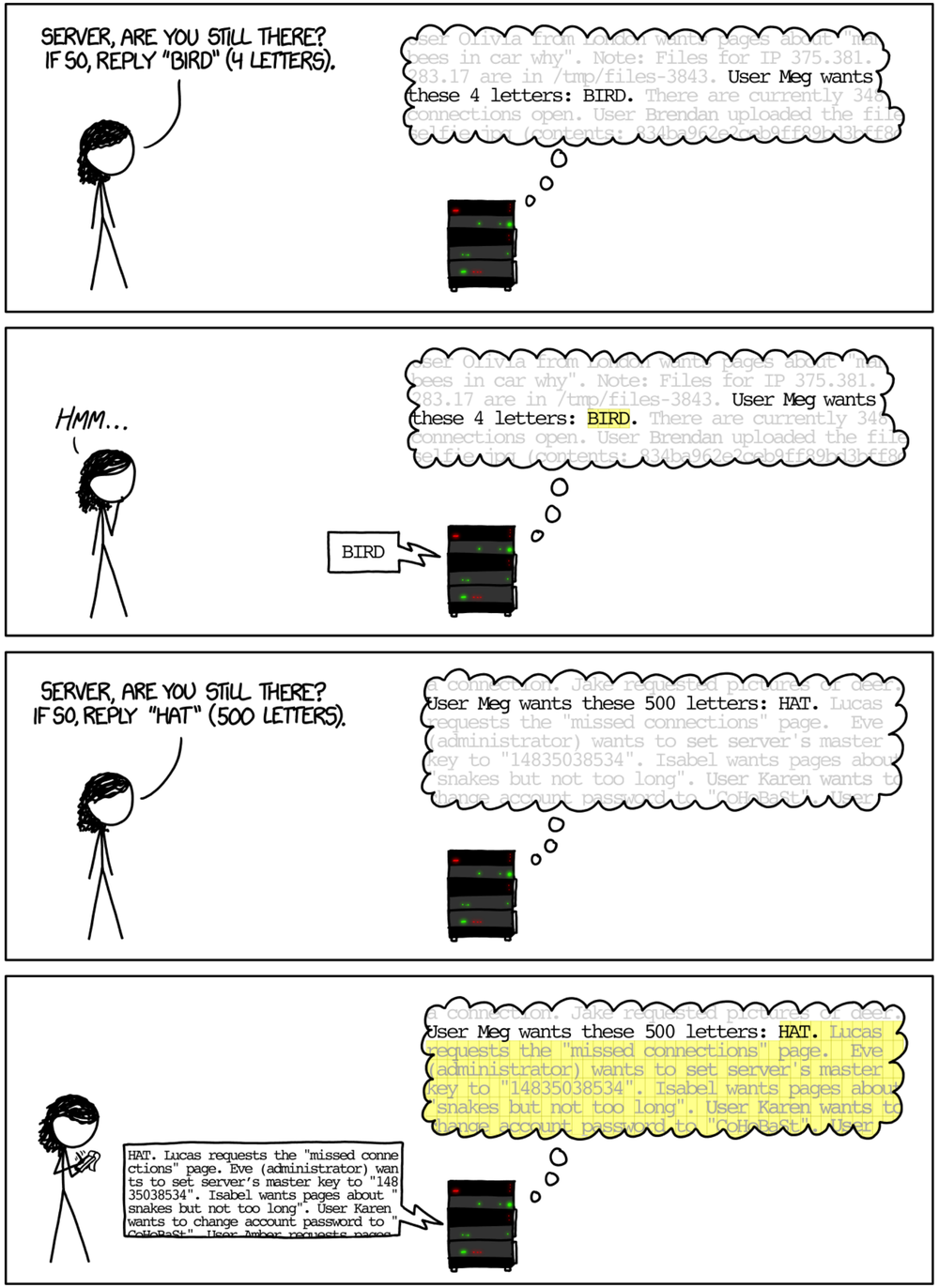
OpenSSL ist schlecht und sollte verschwinden

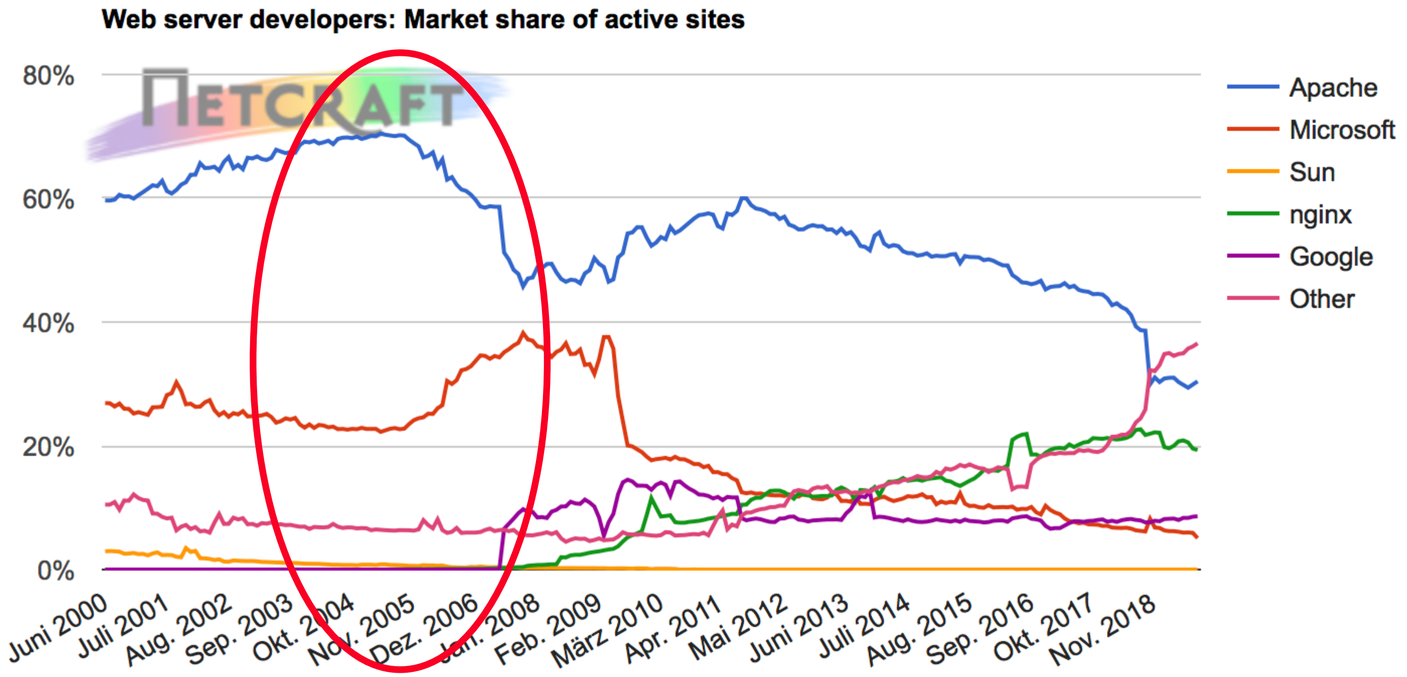
Das OpenSSL Problem

OpenSSL in seiner aktuellen Form ist schlecht und sollte verschwinden

1995 SSLeay
1998 OpenSSL
… ???







Developer	July 2019	Percent	August 2019	Percent	Change
Apache	56,937,841	29.91%	55,570,824	30.42%	0.51
nginx	37,218,850	19.55%	35,295,177	19.32%	-0.23
Google	16,266,687	8.54%	15,748,171	8.62%	0.08
Microsoft	11,308,526	5.94%	9,274,322	5.08%	-0.86

https://news.netcraft.com/archives/2019/08/15/august-2019-web-server-survey.html

Heartbleed Nachwehen

- Riesige Passwortwechselaktionen – weltweit
- Unzählige Zertifikate müssen zurückgezogen werden
- Certification revocation Mechanismen sind kaputt
- Forks wie *LibreSSL* und *BoringSSL*
- Gründung der *Core Infrastructure Initiative*

Heartbleed Nachwehen

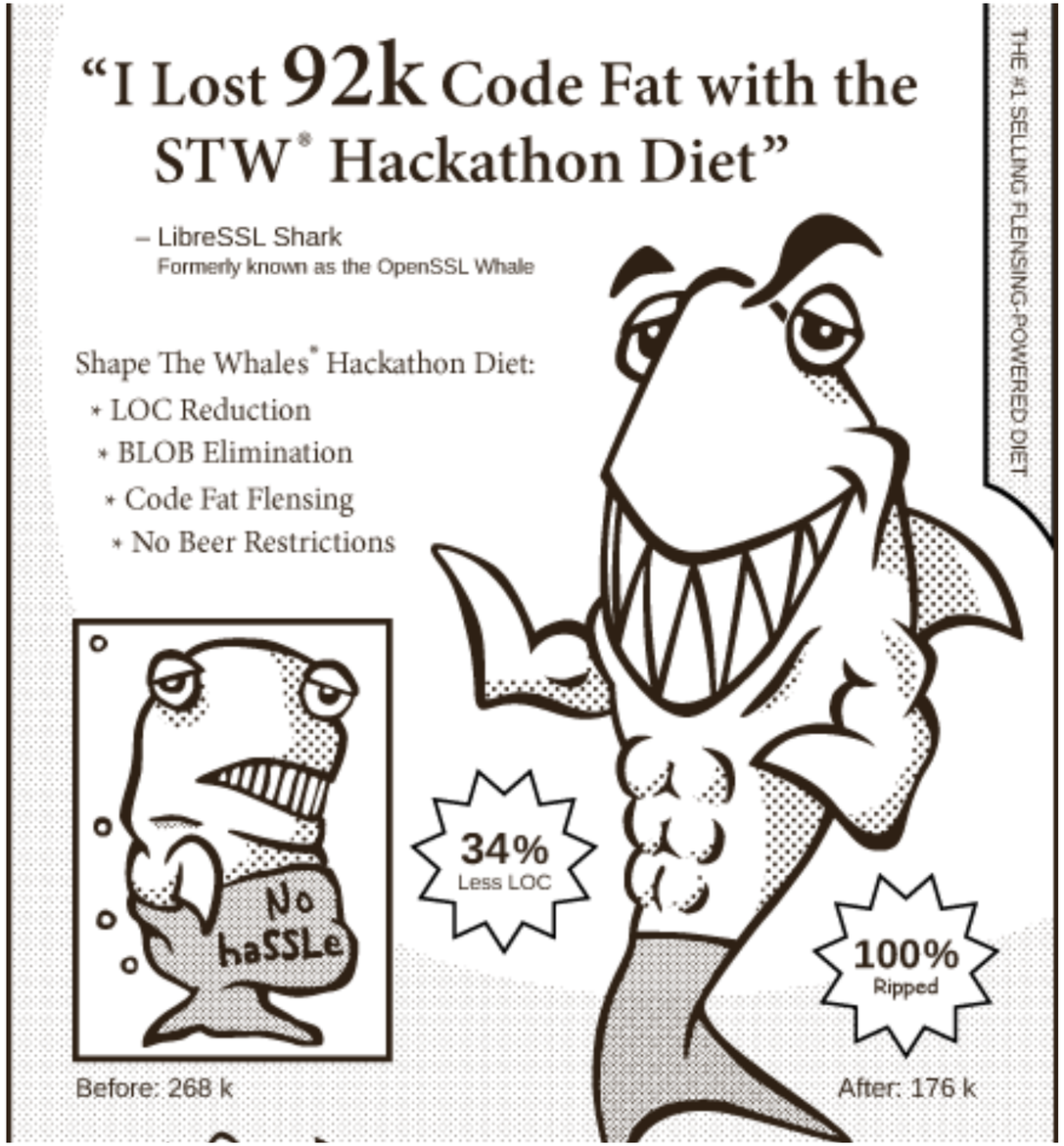
- Sicherheitslücken bekommen schicke Namen und eigenen Webseiten
- Großes Medienecho auch außerhalb der IT
- #heartbleedvirus
- 2017 sind immer noch > 200.000 Server von Heartbleed betroffen – auch heute sind noch betroffenen Systeme zu finden

LibreSSL

- No central architectural authority
- 6,740 `goto` statements
- Inline assembly code
- Multiple different coding styles
- Obscure use of macro preprocessors
- Inconsistent naming conventions
- Far too many selections and options
- Unexplained dead code
- Misleading and incoherent comments

LibreSSL

- Entfernen von alten/unsicheren kryptografischen Funktionen
- Entfernung der Unterstützung von obskuren und alten Betriebssystemen
- Verbesserung der Lesbarkeit des Quelltexts
- Neue Funktionen: aktuelle Verschlüsselungsverfahren
- API-Kompatibel zu OpenSSL 1.0.1
- Eigene (verbesserte) API



Warum ist / war OpenSSL in einem solchen schlechten Zustand?

Kaputtes OpenSSL – warum?

- Spendenaufkommen pro Jahr: ~$2000
- Eine Vollzeitkraft, ein Hand voll Freiwillige
- ~500.000 LOC
- „The mystery is not that a few overworked volunteers missed this bug; the mystery is why it hasn't happened more often.“ –
  Steve Marquess, OpenSSL foundation's president

Hätte man dies nicht auch schon früher erkennen können?

Enhance EVP code to generate random symmetric keys of the

```
author     Dr. Stephen Henson <steve@openssl.org>
           Sun, 28 Mar 2004 19:38:00 +0200 (17:38 +0000)
committer  Dr. Stephen Henson <steve@openssl.org>
           Sun, 28 Mar 2004 19:38:00 +0200 (17:38 +0000)
```

```
appropriate form, for example correct DES parity.

Update S/MIME code and EVP_SealInit to use new functions.

PR: 700
```

```
+#ifdef EVP_CHECK_DES_KEY
+        if (DES_set_key_checked(&deskey[0],&data(ctx)->ks1)
+                !! DES_set_key_checked(&deskey[1],&data(ctx)->ks2))
+                return 0;
+#else
```

Remove EVP_CHECK_DES_KEY

author Emilia Kasper <emilia@openssl.org>
Wed, 14 Oct 2015 18:32:38 +0200 (18:32 +0200)
committer Emilia Kasper <emilia@openssl.org>
Wed, 14 Oct 2015 18:45:33 +0200 (18:45 +0200)

Thanks to the OpenBSD community for bringing this to our attention.

Reviewed-by: Rich Salz <rsalz@openssl.org>

CHANGES	patch | blob | history
crypto/evp/e_des.c	patch | blob | history
crypto/evp/e_des3.c	patch | blob | history

diff --git a/CHANGES b/CHANGES

index ec55dc3..3d9c183 100644 (file)

```
--- a/CHANGES
+++ b/CHANGES
@@ -3,6 +3,8 @@
  ------------------

  Changes between 1.0.2 and 1.1.0  [xx XXX xxxx]
+  *) Remove EVP_CHECK_DES_KEY, a compile-time option that never compiled.
+     [Emilia Käsper]
```

Enhance EVP code to generate random symmetric keys of the

```
author    Dr. Stephen Henson <steve@openssl.org>
          Sun, 28 Mar 2004 19:38:00 +0200 (17:38 +0000)
committer Dr. Stephen Henson <steve@openssl.org>
          Sun, 28 Mar 2004 19:38:00 +0200 (17:38 +0000)
```

```
appropriate form, for example correct DES parity.

Update S/MIME code and EVP_SealInit to use new functions.

PR: 700
```

```
+#ifdef EVP_CHECK_DES_KEY
+       if (DES_set_key_checked(&deskey[0],&data(ctx)->ks1)
+               !! DES_set_key_checked(&deskey[1],&data(ctx)->ks2))
+               return 0;
+#else
```

Subject: Random number generator, uninitialised data and valgrind.
Date: 2006-05-01 19:14:00

When debbuging applications that make use of openssl using
valgrind, it can show alot of warnings about doing a conditional
jump based on an unitialised value. Those unitialised values are
generated in the random number generator. It's adding an
unintialiased buffer to the pool.

The code in question that has the problem are the following 2
pieces of code in crypto/rand/md_rand.c:

```
247:
                MD_Update(&m,buf,j);

467:
#ifndef PURIFY
                MD_Update(&m,buf,j); /* purify complains */
#endif
```

Because of the way valgrind works (and has to work), the place
where the unitialised value is first used, and the place were the
error is reported can be totaly different and it can be rather
hard to find what the problem is.

...

What I currently see as best option is to actually comment out
those 2 lines of code. But I have no idea what effect this
really has on the RNG. The only effect I see is that the pool
might receive less entropy. But on the other hand, I'm not even
sure how much entropy some unitialised data has.

What do you people think about removing those 2 lines of code?

List: openssl-dev
Subject: Re: Random number generator, uninitialised data and valgrind.
Date: 2006-05-01 22:34:12

> What I currently see as best option is to actually comment out
> those 2 lines of code. But I have no idea what effect this
> really has on the RNG. The only effect I see is that the pool
> might receive less entropy. But on the other hand, I'm not even
> sure how much entropy some unitialised data has.
>
Not much. If it helps with debugging, I'm in favor of removing them.
(However the last time I checked, valgrind reported thousands of bogus
error messages. Has that situation gotten better?)

List: openssl-dev
Subject: Re: Random number generator, uninitialised data and valgrind.
Date: 2006-05-02 6:08:10

> Not much. If it helps with debugging, I'm in favor of removing them.
> (However the last time I checked, valgrind reported thousands of bogus
> error messages. Has that situation gotten better?)

I recently compiled vanilla OpenSSL 0.9.8a with -DPURIFY=1 and on Debian
GNU/Linux 'sid' with valgrind version 3.1.1 was able to debug some
application using both TLS/SSL as S/MIME without any warning or error
about the OpenSSL code. Without -DPURIFY you're indeed flooded with
warnings.

So yes I think not using the uninitialized memory (it's only a single
line, the other occurrence is already commented out) helps valgrind.

Wie sieht es heute aus?

Der Stand heute

- Etliche Altlasten wurden entfernt
- Der Coding Style wurde vereinheitlicht
- Der Code hat mehrere Audits hinter sich
- Der Entwicklungsprozess ist formalisiert worden

Kryptografie ist schwierig

- Korrekte Implementation
- Absicherung gegen Seitenkanalangriffe
- So schnell schein

Kryptografie ist schwierig

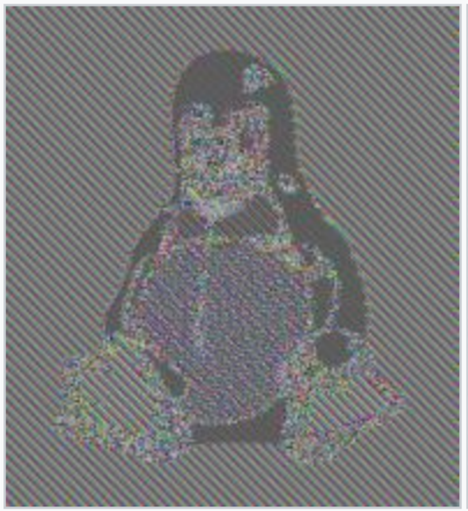
- Welcher Cipher, welche Hash-Funktion?
- Symmetrisch oder Asymmetrisch?
- Welche Modus?
- Block- oder Stream-Cipher?
- Schlüssellängen?
- Welches nonce, welcher IV?
- MAC-then-encrypt oder encrypt-then-MAC oder AEAD?
- Authenticated Encryption with Associated Data?


XSalsa20+Poly1305
RICHTIG, RICHTIG GEIL!

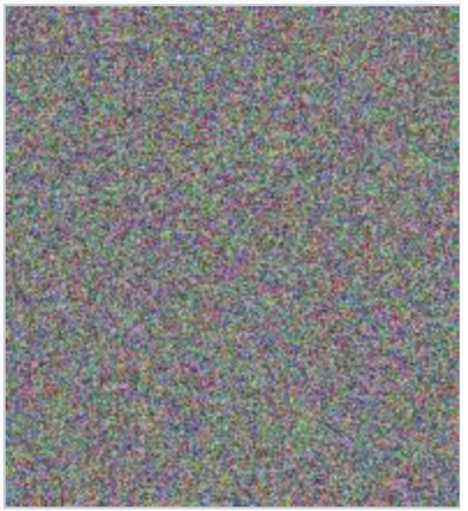

AES256-GCM
SUPERGEIL!


AES-ECB
NICHT SO GEIL!

Original image



Encrypted using ECB mode



Modes other than ECB result in pseudo-randomness

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

TLS ist noch viel schwieriger

- TLS-Protokolle:
  - Record Protokoll
  - Handshake Protokoll
  - Change Cipher Spec Protokoll
  - Alert Protokoll
  - Application Data Protokoll
- X.509 Zertifikate
  - ASN.1
- Server Name Indication (SNI)
- Application-Layer Protocol Negotiation (ALPN)

Angriffe auf TLS

- FREAK
- Logjam
- DROWN
- BEAST
- CRIME
- BREACH
- POODLE
- Lucky Thirteen

Was hat das mit einer Implementierung wie OpenSSL zu tun?

Wie sieht eine vernünftige API aus?

- Geringer Umfang und geringe Komplexität
- The python way – nicht the perl way
- Secure defaults
- Fehlbedienung soweit wie möglich ausschließen
- Gute und strukturierte Dokumentation

Kompatibilität

Ja:

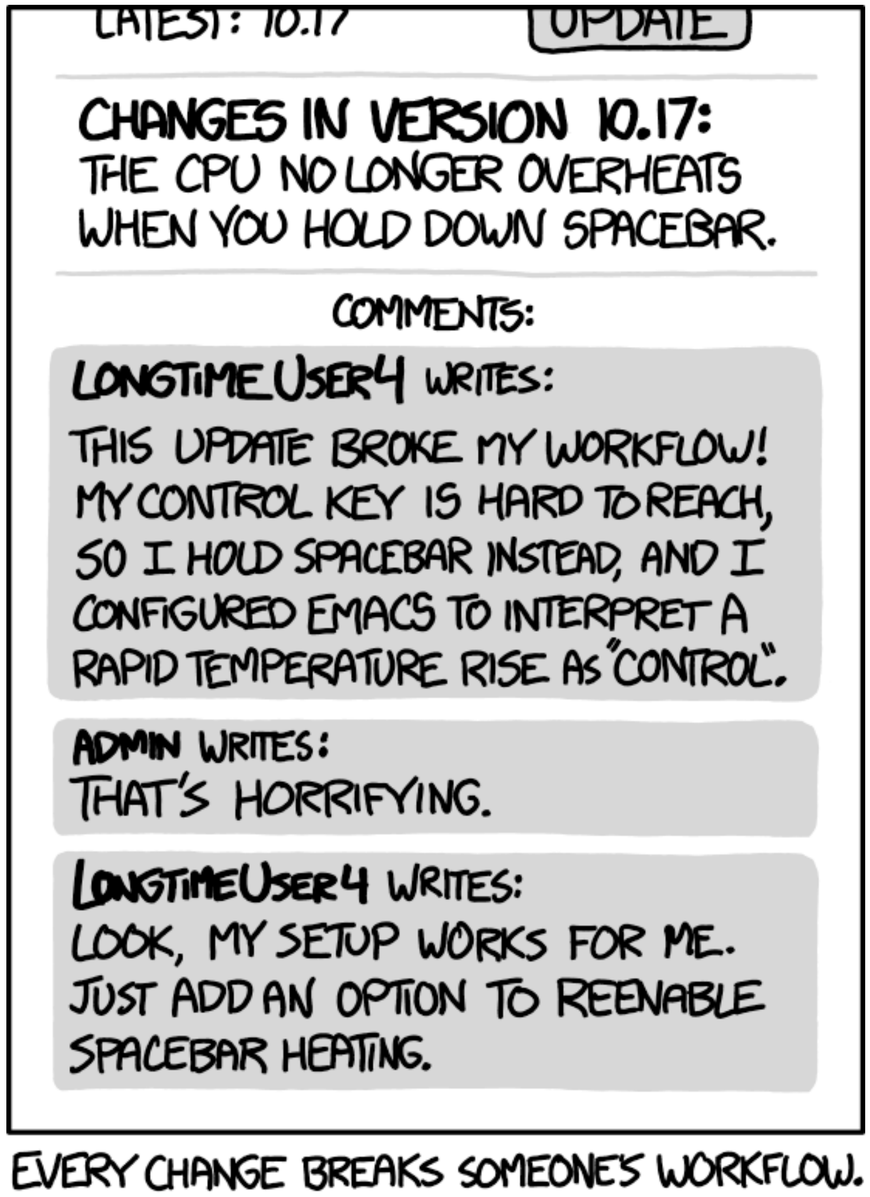
- bei Bugfixes

Nein:

- bei kaputter Krypto
- neuer best practice

Jein:

- mittels build flag - standard: ausgechaltet



https://xkcd.com/1172/

C/C++ Features

- Performance
- Low-level memory control
- Memory corruption bugs

UPDATE 2014-09-04: The above code is not guaranteed to work after all.

Now, I'm not the first person to look at this problem, of course, and if you're willing to limit yourself to narrow platforms, you don't need to write `secure_memzero` yourself: On Windows, you can use the `SecureZeroMemory` function, and on C11 (are there any fully C11-compliant platforms yet?) you can use the `memset_s` function. Both of these are guaranteed (or at least specified) to write the provided buffer and to not be optimized away.

There's just one catch: We've been solving the wrong problem.

Zeroing buffers is insufficient

On Thursday I wrote about the problem of zeroing buffers in an attempt to ensure that sensitive data (e.g., cryptographic keys) which is no longer wanted will not be left behind. I thought I had found a method which was guaranteed to work even with the most vexatiously optimizing C99 compiler, but it turns out that even that method wasn't guaranteed to work. That said, with a combination of tricks, it is certainly possible to make *most* optimizing compilers zero buffers, simply because they're not smart enough to figure out that they're not required to do so — and some day, when C11 compilers become widespread, the `memset_s` function will make this easy.

There's just one catch: We've been solving the wrong problem.

With a bit of care and a cooperative compiler, we can zero **a** buffer — but that's not what we need. What we need to do is zero **every** location where sensitive data might be stored. Remember, the whole reason we had sensitive information in memory in the first place was so that we could use it; and that usage almost certainly resulted in sensitive data being copied onto the stack and into registers.

A proactive approach to more secure code

1 Comment / Security Research & Defense / By MSRC Team / July 16, 2019 / Memory Safety, Rust, Safe Systems Programming Languages, Secure Development

What if we could eliminate an entire class of vulnerabilities before they ever happened?

Since 2004, the Microsoft Security Response Centre (MSRC) has triaged every reported Microsoft security vulnerability. From all that triage one astonishing fact sticks out: as Matt Miller discussed in his 2019 presentation at BlueHat IL, the majority of vulnerabilities fixed and with a CVE assigned are caused by developers inadvertently inserting memory corruption bugs into their C and C++ code. As Microsoft increases its code base and uses more Open Source Software in its code, this problem isn't getting better, it's getting worse. And Microsoft isn't the only one exposed to memory corruption bugs—those are just the ones that come to MSRC.

https://msrc-blog.microsoft.com/2019/07/16/a-proactive-approach-to-more-secure-code/

Episode IV

A NEW HOPE

[-] Rustls - a modern TLS library

Rustls is a TLS library that aims to provide a good level of cryptographic security, requires no configuration to achieve that security, and provides no unsafe features or obsolete cryptography.

Results

Performance results are presented in other blog posts as follows:

- Bulk performance.
- Full handshakes.
- Resumed handshakes.
- Memory usage.

See those posts for details and analysis. To summarise the results, though, we can say approximately:

- rustls is 15% quicker to send data.
- rustls is 5% quicker to receive data.
- rustls is 20-40% quicker to set up a client connection.
- rustls is 10% quicker to set up a server connection.
- rustls is 30-70% quicker to resume a client connection.
- rustls is 10-20% quicker to resume a server connection.
- rustls uses less than half the memory of OpenSSL.

Fazit

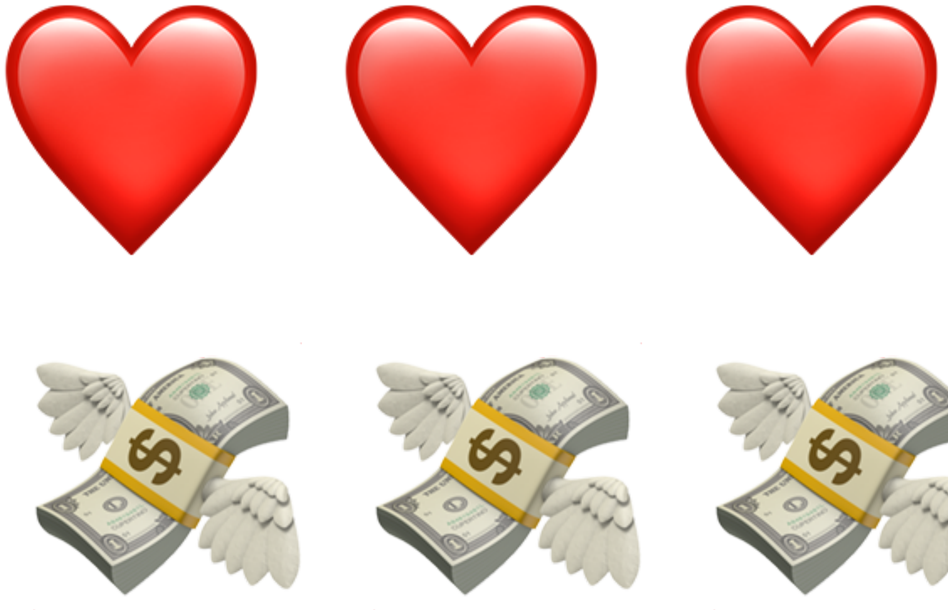
Auch wenn der technische Unterbau stimmt, ist weder die API noch die Dokumentation so, dass OpenSSL leicht und vor allem sicher genutzt werden kann.

Was machen wir jetzt?

Und nun?

- Verschlüsselung: *libsodium*, *cryptography* et al., *libcrypto*
- TLS: *rustls*, *go*, ..., *libtls/libssl*
- Zertifikate/PKI: *cfssl*
- Debugging/CLI: *$ openssl*

Was kannst Du machen?



Vielen Dank!
Fragen? Antworten!

Christoph Iserlohn
christoph.iserlohn@innoq.com

INNOQ
www.innoq.com

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin
Germany
+49 2173 3366-0

Ludwigstr. 180E
63067 Offenbach
Germany
+49 2173 3366-0

Kreuzstr. 16
80331 München
Germany
+49 2173 3366-0

Hermannstrasse 13
20095 Hamburg
Germany
+49 2173 3366-0

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
+41 41 743 0116