SSL/TLS jungle
bringing light into the cipher forest

For OWASP.ch
Dobin Rutishauser, dobin.rutishauser@csnc.ch
Compass Security Schweiz AG – www.csnc.ch
10.04.2014, v1.1
Alternative title:

My Heart Is Bleeding...
Content

• SSL/TLS Introduction
• SSL/TLS Attacks (BEAST, CRIME, ..)
• Perfect Forward Secrecy (PFS)
• PRISM
• Heartbleed
• The CA Problem
• Conclusion
About me

• Development of a distributed stealth portscanner for IRC friends in 2001 (dscan) – nuff said

• >3 years at Compass Security Schweiz AG.

• Web App Hacking, Penetration Testing, Exploit Writing, Linux User

• Somehow acquired knowledge about SSL during Compass audits

• Current project: Burp Sentinel
  • Plugin for Burp, soon ZAP too
  • Helps finding vulnerabilities
  • https://github.com/dobin/BurpSentinel
What’s SSL/TLS?
https://ebanking-ch1.ubs.com:443/

Die Identität von UBS AG am Standort Zürich, Zürich CH wurde von VeriSign Class 3 Extended Validation SSL SGC CA bestätigt. Es liegt jedoch kein öffentlicher Eintrag vor. 
Zertifikatinformationen

Die Verbindung zu ebanking-ch1.ubs.com ist mit einer 256-Bit-Verschlüsselung verschlüsselt.

Die Verbindung verwendet TLS 1.0.

Die Verbindung ist mit AES_256_CBC verschlüsselt; für die Nachrichtenauthentifizierung wird SHA1 verwendet und als Mechanismus für den Schlüsselaustausch DHE_RSA.
What does SSL/TLS do?

Provides secure tunnel for insecure protocols

- Confidentiality
- Integrity
- Authenticity

Often used with:

- HTTP
- SMTP/IMAP/POP3
- VPN
- SIP
Where is TLS used?

Public Websites
- Online Shopping
- E-Banking
- Often provided by an entry server / WAF (Airlock, SES, F5, ..)

Administration Interfaces
- WAF
- vSphere
- HP Management Service

Technical Communication
- Web Frontend -> Backend (SOAP, REST, …)
- WLAN PEAP-TLS
- VPN
SSL Handshake

Client

Available Cipher List

Server Heo

Selected Cipher

Server Certificate

Client Key Exchange

Encrypt_pubkey(sessionkey)

Alrighty…

Server
### OpenSSL Ciphers Suites Example

```
$ openssl ciphers MEDIUM -v
DHE-RSA-SEED-SHA   SSLv3  Kx=DH    Au=RSA  Enc=SEED(128)  Mac=SHA1
DHE-DSS-SEED-SHA   SSLv3  Kx=DH    Au=USS  Enc=SEED(128)  Mac=SHA1
ADH-SEED-SHA       SSLv3  Kx=DH    Au=None Enc=SEED(128)  Mac=SHA1
SEED-SHA           SSLv3  Kx=RSA    Au=RSA  Enc=SEED(128)  Mac=SHA1
IDEA-CBC-SHA       SSLv3  Kx=RSA    Au=RSA  Enc=IDEA(128)  Mac=SHA1
IDEA-CBC-MD5       SSLv2  Kx=RSA    Au=RSA  Enc=IDEA(128)  Mac=MD5
RC2-CBC-MD5        SSLv2  Kx=RSA    Au=RSA  Enc=RC2(128)   Mac=MD5
ECDHE-RSA-RC4-SHA  SSLv3  Kx=ECDH   Au=RSA  Enc=RC4(128)  Mac=SHA1
ECDHE-ECDSA-RC4-SHA SSLv3  Kx=ECDH   Au=ECDSA Enc=RC4(128) Mac=SHA1
AEDH-RC4-SHA       SSLv3  Kx=ECDH   Au=None Enc=RC4(128)  Mac=SHA1
ADH-RC4-MD5        SSLv3  Kx=DH     Au=None Enc=RC4(128)  Mac=MD5
ECDH-RSA-RC4-SHA   SSLv3  Kx=ECDH/RSA Au=ECDH Enc=RC4(128) Mac=SHA1
ECDH-ECDSA-RC4-SHA SSLv3  Kx=ECDH/ECDSA Au=ECDH Enc=RC4(128) Mac=SHA1
RC4-SHA            SSLv3  Kx=RSA    Au=RSA  Enc=RC4(128)   Mac=SHA1
RC4-MD5            SSLv3  Kx=RSA    Au=RSA  Enc=RC4(128)   Mac=MD5
RC4-MD5            SSLv2  Kx=RSA    Au=RSA  Enc=RC4(128)   Mac=MD5
PSK-RC4-SHA        SSLv3  Kx=PSK    Au=PSK  Enc=RC4(128)   Mac=SHA1
```
SSL/TLS Details
OpenSSL Ciphers – Structure

$ openssl ciphers -v LOW
EDH-RSA-DES-CBC-SHA  SSLv3  Kx=DH  Au=RSA  Enc=DES(56)  Mac=SHA1
EDH-DSS-DES-CBC-SHA  SSLv3  Kx=DH  Au=DSS  Enc=DES(56)  Mac=SHA1

- SSL/TLS Version
  - SSLv2, SSLv3, TLS1.0, TLS1.1, TLS1.2

- Key Exchange Mechanism
  - RSA, DH, DHE/EDH, ECDHE, …

- Authentication Mechanism
  - RSA, …

- Encryption Algorithm
  - RC4, DES, AES, IDEA, SEED, …
OpenSSL Ciphers – Encryption Strength

«Really Bad»
- NULL, EXP (EXPORT), ADH

LOW:
- DES-CBC

MEDIUM:
- SEED, IDEA, RC2
- RC4-MD5?

High:
- AES, AES-GCM, DES3
- CAMELIA?
OpenSSL Ciphers - Key Exchange

- **RSA**
  - Client encrypts session key with public key of server certificate

- **DH**
  - Diffie Hellman key exchange
  - NO REAL DH KEY EXCHANGE!
    - Uses static data from certificate for key exchange
    - No perfect forward secrecy (PFS)

- **DHE/EDH/ECDHE**
  - Ephemeral Diffie Hellman
    - Provides PFS
$ sslyze -regular ebanking-ch1.ubs.com:443

* TLSv1 Cipher Suites:
  Preferred Cipher Suite:
    DHE-RSA-AES256-SHA  256 bits
  Accepted Cipher Suite(s):
    DHE-RSA-AES256-SHA  256 bits
    AES256-SHA         256 bits
    EDH-RSA-DES-CBC3-SHA 168 bits
    DES-CBC3-SHA      168 bits
    DHE-RSA-AES128-SHA 128 bits
    AES128-SHA        128 bits

* SSLv3 Cipher Suites:
  Preferred Cipher Suite:
    DHE-RSA-AES256-SHA  256 bits
  Accepted Cipher Suite(s):
    DHE-RSA-AES256-SHA  256 bits
    AES256-SHA         256 bits
    EDH-RSA-DES-CBC3-SHA 168 bits
...
SSL Versions - Weaknesses

SSLv2
- No No No!
- Length extension attacks, truncation attacks, downgrade attacks, vulnerable to Man-in-the-Middle attacks, …
- Patched-out in Ubuntu (without updating man page)

SSLv3
- Released in 1996…
- Weaker key derivation than TLS 1.0
- Cannot be validated under FIPS 140-2
- There have been various attacks on SSLv3 implementations
- Vulnerable to certain protocol downgrade attacks
TLS Version - Advantages

- **TLS 1.0**
  - Released in 1999 (!!)
  - Cannot downgrade to SSL 3.0
  - Uses MD5 AND SHA1 at the same time

- **TLS 1.1**
  - Added protection against CBC attacks

- **TLS 1.2**
  - Enhancement of client side preferred hash/sign algorithms
  - Support GCM and CCM ciphers
  - Supported by all modern browsers!
SSLv3 Cipher Suites:
Preferred Cipher Suite:
   DHE-RSA-AES256-SHA  256 bits
   [...]

TLSv1 Cipher Suites:
Preferred Cipher Suite:
   DHE-RSA-AES256-SHA  256 bits
   [...]

TLSv1_1 Cipher Suites:
Preferred Cipher Suite: None
Accepted Cipher Suite(s): None

TLSv1_2 Cipher Suites:
Preferred Cipher Suite: None
Accepted Cipher Suite(s): None
https://ebanking-ch1.ubs.com:443/

Die Verbindung zu ebanking-ch1.ubs.com ist mit einer 256-Bit-Verschlüsselung verschlüsselt.

Die Verbindung verwendet TLS 1.0.

Die Verbindung ist mit AES_256_CBC verschlüsselt; für die Nachrichtenauthentifizierung wird SHA1 verwendet und als Mechanismus für den Schlüsselaustausch DHE_RSA.
TLS Support in Browsers
## SSL/TLS Browser Support 1/2


<table>
<thead>
<tr>
<th>Browser</th>
<th>Version</th>
<th>Platforms</th>
<th>TLS 1.0</th>
<th>TLS 1.1</th>
<th>TLS 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Google Chrome</strong></td>
<td>0–21</td>
<td>Android, iOS, Linux, Mac OS X, Windows (XP, Vista, 7, 8)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>22–29</td>
<td></td>
<td>Yes[32]</td>
<td>Yes</td>
<td>No[32][33][34][35]</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td></td>
<td>Yes[36]</td>
<td>Yes, disabled by default[28][37]</td>
<td>Yes, disabled by default[30][38]</td>
</tr>
<tr>
<td></td>
<td>24–26</td>
<td></td>
<td>Yes[36]</td>
<td>Yes, disabled by default[28][37]</td>
<td>Yes, disabled by default[30][38]</td>
</tr>
<tr>
<td><strong>Internet Explorer</strong></td>
<td>6</td>
<td>Windows (98, 2000, ME, XP)</td>
<td>Yes, disabled by default</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>7–8</td>
<td>Windows XP</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>7–9</td>
<td>Windows Vista</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>8–10</td>
<td>Windows 7</td>
<td>Yes</td>
<td>Yes, disabled by default</td>
<td>Yes, disabled by default</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Windows 8</td>
<td>Yes</td>
<td>Yes, disabled by default</td>
<td>Yes, disabled by default</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Windows 7, 8.1</td>
<td>Yes</td>
<td>Yes[42]</td>
<td>Yes[42]</td>
</tr>
</tbody>
</table>
## SSL/TLS Browser Support 2/2


<table>
<thead>
<tr>
<th>Opera</th>
<th>5–7</th>
<th>8–9</th>
<th>10–12</th>
<th>14–16</th>
<th>17–</th>
</tr>
</thead>
<tbody>
<tr>
<td>[notes 6]</td>
<td>Android, [citation needed]</td>
<td>Yes</td>
<td>Yes, disabled by default</td>
<td>Yes, disabled by default</td>
<td>Yes</td>
</tr>
<tr>
<td>[notes 7]</td>
<td>iOS, [citation needed]</td>
<td>Yes</td>
<td>Yes, disabled by default</td>
<td>Yes, disabled by default</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Linux, Mac OS X, Windows</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Safari</th>
<th>1–6</th>
<th>7</th>
<th>3–5</th>
<th>5–6</th>
<th>7</th>
<th>3–5</th>
</tr>
</thead>
<tbody>
<tr>
<td>[notes 8]</td>
<td>Mac OS X –10.8 [notes 9]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Mac OS X 10.9 [notes 10]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>iPhone OS 1–3, iOS 4.0 [notes 11] [notes 9]</td>
<td>Yes</td>
<td>Yes, no</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>iOS 5–6 [notes 11] [notes 9]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>iOS 7 [notes 11] [notes 9]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Windows</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Comparison between RC4 and 3DES

Browsers without AES

- Old browsers may not support AES
- Like IE6 on XP
- RC4 or 3DES should always be offered by the Server

RC4

- + Not vulnerable to BEAST
- - Some say, can be broken in realtime by NSA
- - Microsoft recommends developers to not use it anymore
- - Several vulnerabilities… (broken in $2^{24}$ connections)

3DES

- + Old (1977) – but still strong
- - But only 112 bits. No! Only 108 bits…
- - CBC, so possible vulnerable against Lucky 13 attacks
Cipher Security


<table>
<thead>
<tr>
<th>Cipher</th>
<th>SSL 2.0</th>
<th>SSL 3.0 [note 1][note 2][note 3]</th>
<th>TLS 1.0 [note 1][note 3]</th>
<th>TLS 1.1 [note 1]</th>
<th>TLS 1.2 [note 1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES CBC[note 4]</td>
<td>N/A</td>
<td>N/A</td>
<td>Depends</td>
<td>Secure</td>
<td>Secure</td>
</tr>
<tr>
<td>AES GCM[18][note 5]</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Secure</td>
</tr>
<tr>
<td>AES CCM[19][note 5]</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Secure</td>
</tr>
<tr>
<td>Camellia CBC[20][note 4]</td>
<td>N/A</td>
<td>N/A</td>
<td>Depends</td>
<td>Secure</td>
<td>Secure</td>
</tr>
<tr>
<td>Camellia GCM[21][note 5]</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Secure</td>
</tr>
<tr>
<td>SEED CBC[22][note 4]</td>
<td>N/A</td>
<td>N/A</td>
<td>Depends</td>
<td>Secure</td>
<td>Secure</td>
</tr>
<tr>
<td>ChaCha20+Poly1305[23][note 5]</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Secure</td>
</tr>
<tr>
<td>IDEA CBC[note 4][note 6]</td>
<td>Insecure</td>
<td>Depends</td>
<td>Depends</td>
<td>Secure</td>
<td>N/A</td>
</tr>
<tr>
<td>Triple DES CBC[note 4][note 7]</td>
<td>Insecure</td>
<td>Depends</td>
<td>Depends</td>
<td>Depends</td>
<td>Depends</td>
</tr>
<tr>
<td>DES CBC[note 4][note 6]</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>N/A</td>
</tr>
<tr>
<td>RC2 CBC[note 4][note 6]</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>N/A</td>
</tr>
<tr>
<td>RC4[note 8]</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
</tr>
</tbody>
</table>
Attacks on TLS/SSL
SSL Attacks

BEAST (2011)
• In TLS < 1.1
• CBC madness
• Needs Man in the Middle
• Needs Content Injection + Same Origin Policy Violation
• Sending a large amount of requests
• Fixed client or server side (stream ciphers like RC4, TLS 1.1, 1.2)

CRIME (2012)
• Uses TLS compression to find cookie
• Needs to sniff traffic
• Needs the user to click malicious link
• Fixed by disabling TLS compression

BREACH (2013)
• Similar to CRIME, but uses HTTP compression
SSL Attacks

Padding Oracle / Lucky 13
- Trickery with CBC block sizes
- Leaking session id’s
- Fixed with «authenticated encryption algorithm»
  - TLS1.2: AES GCM, AES CCM
- Fixed with RC4…
- Fixed with implementation fixes

RC4 Bias
- First few bytes of RC4 stream cipher are biased
PFS

Perfect Forward Secrecy
Whats PFS?

Short-Term Keys are not dependant on Long-Term Keys

Recorded communication and stolen private key:
- Without PFS: Decrypt ALL past communication in no time
- With PFS: Need to brute force every single connection!

PFS helps against compromised certificates

But not much against compromised ciphers
- Even if cipher is broken, still need to crack each connection individually

Not helpful against Man-in-the-Middle attacks with stolen cert
PFS Pitfalls

Session Resumption with Session ID’s
- Re-use SSL Session ID to shortcut handshake

Session Resumption with Session Tickets
- Send SSL state encryption with a server key to client
- Client sends the encrypted blob to server upon resumption
- How to distribute key to all LB’s?
Some other stuff
Other SSL Vulnerabilities…

**Insecure Renegotiation**
- From 2009
- Possible to insert plaintext at beginning of a SSL protected connection
- Fixed with «Secure Renegotiation»

**Client Initiated Renegotiation**
- More calculation for the server -> DoS

Independant of each other!
https://ebanking-ch1.ubs.com:443/

$ sslyze --regular ebanking-ch1.ubs.com:443

* Session Renegotiation:
  Client-initiated Renegotiations: Rejected
  Secure Renegotiation: Supported

* Compression:
  Compression Support: Disabled
Other SSL Vulnerabilities…

Browser TLS -> SSL downgrade fallbacks

- TLS 1.2 -> TLS 1.1 -> TLS 1.0 -> SSLv3!
- Just needs man in the Middle
- Fix?
  - «Fake Ciphers»
  - Not really implemented right now
PRISM
How to thwart the NSA

They may be able to break:

- Export, NULL, Low Ciphers
- Medium Ciphers (RC2, RC4, IDEA?, ..)
- and CAMELIA? (HIGH, but who knows…)

But not:

- Ciphers they use themself up and with TOP SECRET
  - AES
- or secured a long time ago, and used by banks:
  - DES
How to thwart the NSA

What if they steal your private keys?
- Use PFS
- Secure your keys! (chmod o-r *.key)

What if they downgrade you to SSLv3?
- Disable it

What if they downgrade you to HTTP?
- Use HSTS header
  - Tell browser to only use HTTPS for this site!
  - Insert your site into browser HSTS list!

What if they issue a fake certificate?
- Use certificate pinning
How to thwart the NSA

Best Attack Vector: Implementation errors

Past implementation errors:
- Apple’s Goto Fail
- Triple Handshake
- GNU TLS Certificate Chain Validation Error
- Heartbleed

That’s just from 2014…

This will not stop
Heartbleed

OpenSSL 1.0.1*

Remotely exploitable

64kb (!) Information Disclosure

Can be repeated indefinitely

Discloses:
- Sensitive User Data
- Cookies
- Private Keys
- PFS Session Keys
- ...

Exploit is public
- Heap Feng Shui?
### Heartbleed

<table>
<thead>
<tr>
<th>Hexadecimal</th>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000cf0</td>
<td>14 6a</td>
<td>eb f8</td>
</tr>
<tr>
<td>000000d00</td>
<td>13 e7</td>
<td>6b 64 13 e7 69 1c</td>
</tr>
<tr>
<td>000000d10</td>
<td>fe 90 5d d8 13 8f 27 a8 00 08 3a a8 fe 90 75 b8</td>
<td>.j.........j.8.j.(</td>
</tr>
<tr>
<td>000000d20</td>
<td>09 88 1f 10 14 6a ed 88 fe 90 6f 9c 14 6a ed 2c</td>
<td></td>
</tr>
<tr>
<td>000000d30</td>
<td>14 6a eb 00 14 6a ee b8</td>
<td></td>
</tr>
<tr>
<td>000000d40</td>
<td>13 8f 29 ac 13 8f 29 1c 13 8f 29 ec 13 8f 25 fc</td>
<td>.]....j.o.j,,</td>
</tr>
<tr>
<td>000000d50</td>
<td>00 00 00 00 00 00 00 00 fe 90 3a a8 00 00 00 00</td>
<td></td>
</tr>
<tr>
<td>000000d60</td>
<td>14 6a ed 20 00 a0 42 b0 fe 90 3a a8 00 00 00 00</td>
<td></td>
</tr>
<tr>
<td>000000d70</td>
<td>14 6a ed 20 00 37 00 38 fe 90 5d d8 14 6a ed 20</td>
<td></td>
</tr>
<tr>
<td>000000d80</td>
<td>00 18 ee 38 fe 90 75 b8 09 88 1f 10 14 6a ed f0</td>
<td>.j....7..]..j</td>
</tr>
<tr>
<td>000000d90</td>
<td>fe 90 6f 9c 14 6a ed 94 14 6a ee 50 14 6a ee 50</td>
<td></td>
</tr>
<tr>
<td>000000da0</td>
<td>14 6a ed e0 14 6a ed d0 13 8f 2a 4c 13 8f 29 1c</td>
<td></td>
</tr>
<tr>
<td>000000db0</td>
<td>13 8f 2a 84 13 8f 25 fc 00 00 00 00 14 6a ed bc</td>
<td></td>
</tr>
<tr>
<td>000000dc0</td>
<td>fe 90 3a a8 00 00 00 00 14 6a ed 88 00 6a ed f8</td>
<td></td>
</tr>
<tr>
<td>000000dd0</td>
<td>fe 90 3a a8 00 00 00 00 14 6a ed 88 00 e7 65 fc</td>
<td></td>
</tr>
<tr>
<td>000000de0</td>
<td>fe 90 7e 78 14 6a ed 88 04 18 3a a8 14 6a ed f0</td>
<td>.~x.j.......j..</td>
</tr>
<tr>
<td>000000df0</td>
<td>00 00 00 00 14 6a ee 10 09 88 1f 08 00 00 00 00</td>
<td>.j.............</td>
</tr>
<tr>
<td>000000e00</td>
<td>14 6a ee 20 00 00 00 08 00 00 00 17 09 88 1f 08</td>
<td>.j.............</td>
</tr>
</tbody>
</table>
Heartbleed

So just in case the graveness of #Heartbleed hasn't been realised by some yet, Yahoo is leaking user credentials
pic.twitter.com/G1v1UBgyiH

The openssl bug #heartbleed is a wonderful exploit. My PoC is even getting files and directories out of mem from what I think is Apache.

When testing the OpenSSL heartbeat fix I never got key material from servers, only old connection buffers. (That includes cookies though.)
Heartbleed

Popular sites which exhibit support for the TLS heartbeat extension include Twitter, GitHub, Yahoo, Tumblr, Steam, DropBox, HypoVereinsbank, PostFinance, Regents Bank, Commonwealth Bank of Australia, and the anonymous search engine DuckDuckGo.
Heartbleed

- /* Read type and payload length first */
- hbtype = *p++;
- n2s(p, payload);
- pl = p;

- if (s->msg_callback)
  s->msg_callback(0, s->version, TLS1_RT_Heartbeat,
  &s->s3->rrec.data[0], s->s3->rrec.length,
  s, s->msg_callback_arg);

+ /* Read type and payload length first */
+ if (1 + 2 + 16 > s->s3->rrec.length)
+   return 0; /* silently discard */
+ hbtype = *p++;
+ n2s(p, payload);
+ if (1 + 2 + payload + 16 > s->s3->rrec.length)
+   return 0; /* silently discard per RFC 6520 sec. 4 */
+ pl = p;
+
Heartbleed

Not sure what is worse being affected by Heartbleed

Or not being affected cause older versions are in use

www.generator.net
Heartbleed

REVOKE ALL THE CERTS

meme generator.net
Heartbleed

Fix:
- Apache no-threads, fork for every connection
  - No more data of other users
- Downgrade to OpenSSL 1.0.0, 0.9.8
- Upgrade to OpenSSL 1.0.1g
- Update all your keys
- PFS helps a bit
- Compile OpenSSL with -DOPENSSL_NO_HEARTBEATS
- HSM? (Hardware Security Module – does not leak private key)

«there are X bad SSL libraries»
- Lets write A GOOD SSL library
- Now, there are X+1 bad SSL libraries

Source:
- OpenSSL is Open Source
- Pull Request For Heartbeat Support
- No consequent peer review
The CA Problem
The CA Problem

<table>
<thead>
<tr>
<th>Issued To</th>
<th>Issued By</th>
<th>Expiration</th>
<th>Friendly Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certum Trusted Net...</td>
<td>Certum Trusted Net...</td>
<td>31.12.2029</td>
<td>Certum Trusted Net...</td>
</tr>
<tr>
<td>Class 2 Primary CA</td>
<td>Class 2 Primary CA</td>
<td>07.07.2019</td>
<td>CertPlus Class 2</td>
</tr>
<tr>
<td>Class 3 Public Primary...</td>
<td>Class 3 Public Primary...</td>
<td>03.08.2028</td>
<td>VeriSign Class 3</td>
</tr>
<tr>
<td>Class 3 Public Primary...</td>
<td>Class 3 Public Primary...</td>
<td>02.08.2028</td>
<td>VeriSign Class 3</td>
</tr>
<tr>
<td>Class 3 Public Primary...</td>
<td>Class 3 Public Primary...</td>
<td>08.01.2004</td>
<td>VeriSign</td>
</tr>
<tr>
<td>ClockworkMod</td>
<td>ClockworkMod</td>
<td>01.01.2040</td>
<td>&lt;None&gt;</td>
</tr>
<tr>
<td>Copyright (c) 1997...</td>
<td>Copyright (c) 1997 Mi...</td>
<td>31.12.1999</td>
<td>Microsoft Time...</td>
</tr>
<tr>
<td>Deutsche Telekom...</td>
<td>Deutsche Telekom Ro...</td>
<td>10.07.2019</td>
<td>Deutsche Telekom</td>
</tr>
<tr>
<td>DigiCert Assured ID...</td>
<td>DigiCert Assured ID R...</td>
<td>10.11.2031</td>
<td>DigiCert</td>
</tr>
</tbody>
</table>

Learn more about certificates
The CA Problem

Exploits: 2011

- 03/2011: CA: Comodo Hack
  → Reaktion: Certificate-Pinning in Chrome
- 07/2011: CA: DigiNotar Hack
  → Reaktion: CA von Regierung übernommen
- 07/2011: PeerJacking (Problem in PHP's cURL) [A-PJ]
- 09/2011: CA: Einbruch bei GlobalSign (Folge von DigiNotar Hack)
- 09/2011: BEAST
  → Reaktion: Empfehlung RC4
- 09/2011: weitere gefälschte Zertifikate (Folge von DigiNotar Hack)
  → 7 Exploits in einem Jahr

Source: SSL in der Praxis, sicher? (Achim Hoffmann)
The CA Problem

How to check for revoked certificates?

CRL
- Offline List
- Replay Attacks
- DNSSpoofing…

OCSP
- Life check
- What if server is not reachable?
- DNSSpoofing…
The CA Problem

Use certificate pinning!

- Ignore the signature hierarchy!
- Check hash of public-key information of the certificate
  - SubjectPublicKeyInfo
- Or, check the issuer CA (always should be issued by Verisign, for example)

In Browser:

- Chrome, IE, FF
- Send them an email to include your site in pinning mechanism
- No official process?

In Windows:

- EMET

In Apps:

- Do it yourself! Very easy!
- Don't forget to push new version before renewal of certificate
Conclusion
Conclusion

- Disable SSLv3 (TLS only)
- Use Ephemeral Ciphers (for PFS)
- Use AES Ciphers
- Do not use RC4
- Disable SSL and HTTP Compression
- Disable Client and insecure Renegotiation
- Update update update!
Conclusion – Web Pages

• Use trustworthy CA
• No wildcard certificates
• EV certificate? Why not…
• Forward :80 -> :443
• Deliver EVERYTHING with HTTPS
• Use «secure» flag on cookies
• Use HSTS header
• Use Certificate Pinning
References

SSL in der Praxis, sicher? achim@owasp.org

SSL CERTIFICATE GOOD PRACTICE GUIDE, Portcullis
- https://labs.portcullis.co.uk/whitepapers/ssl-certificate-good-practice-guide/

SSL/TLS Deployment Best Practices, Qualys SSL LABS

ImperialViolet (Google Chrome Developer Blog)
- https://www.imperialviolet.org/

This presentation is based on the following blog entry:
Rant:

Browser Indicators
Das Sicherheitszertifikat der Website ist nicht vertrauenswürdig!


Fahren Sie nicht fort, insbesondere wenn diese Warnung für diese Website vorher noch nie erschienen ist.

Trotzdem fortfahren  Zurück zu sicherer Website

Mehr Infos dazu
Rant: Browser Indicators

This Connection is Untrusted

You have asked Firefox to connect securely to [redacted] but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

Get me out of here!

▷ Technical Details

▷ I Understand the Risks
There is a problem with this website’s security certificate.

The security certificate presented by this website was not issued by a trusted certificate authority.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

We recommend that you close this webpage and do not continue to this website.

- Click here to close this webpage.
- Continue to this website (not recommended).
- More information