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

Compass Security Schweiz AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55 214 41 60
Fax +41 55 214 41 61
team@csnc.ch
www.csnc.ch

Alternative title:

My Heart Is Bleeding…

Content

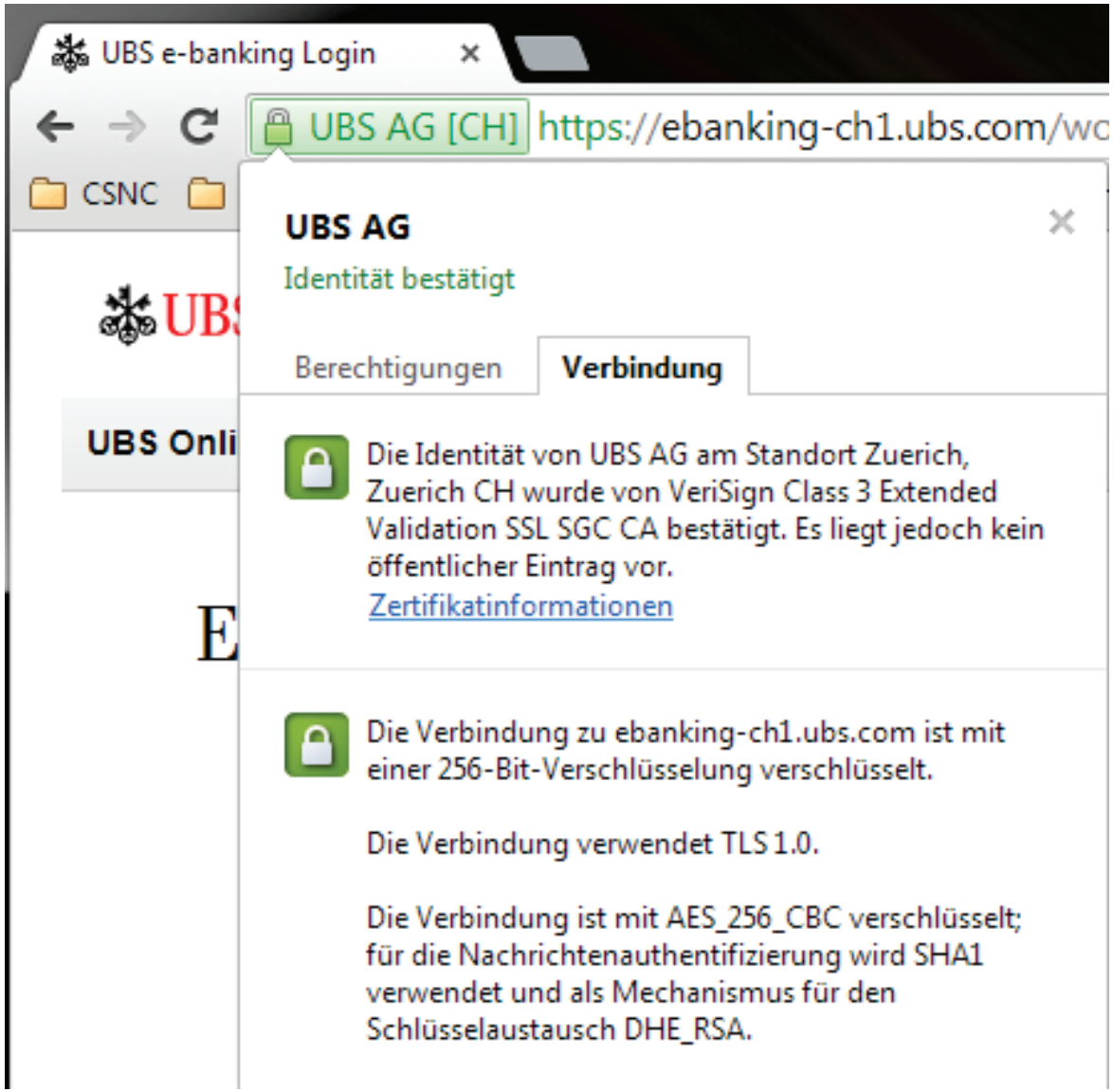
- SSL/TLS Introduction
- SSL/TLS Attacks (BEAST, CRIME, ..)
- Perfect Forward Secrecy (PFS)
- PRISM
- Heartbleed
- The CA Problem
- Conclusion

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 aquired 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/



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, ..)

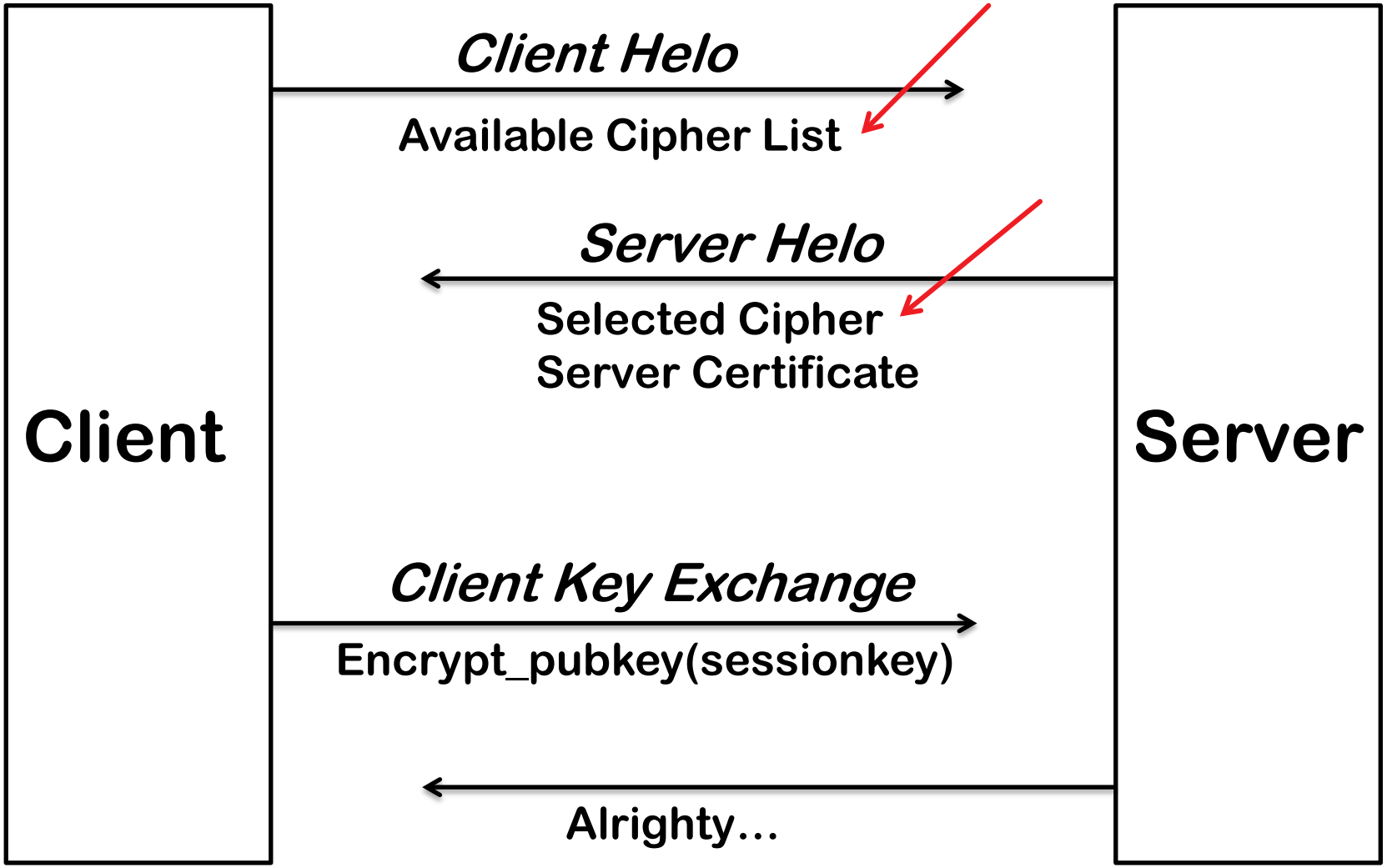
Administration Interfaces

- WAF
- vSphere
- HP Management Service

Technical Communication

- Web Frontend -> Backend (SOAP, REST, …)
- WLAN PEAP-TLS
- VPN

SSL Handshake



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=DSS  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
AECDH-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

https://ebanking-ch1.ubs.com:443/

```
$ 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 algorithmns
  - Support GCM and CCM ciphers
  - Supported by all modern browsers!

https://ebanking-ch1.ubs.com:443/

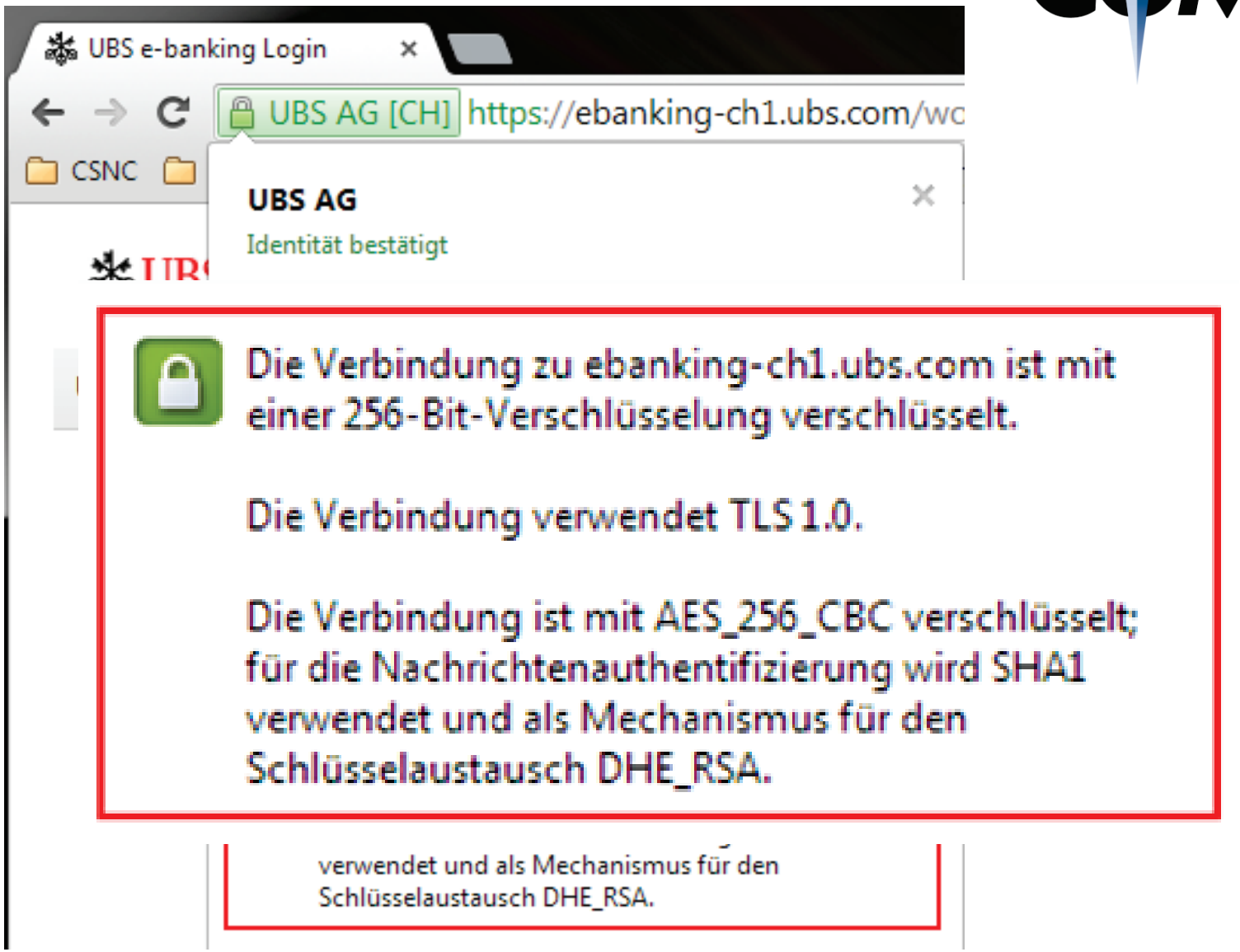
```
* SSLV3 Cipher Suites :
      Preferred Cipher Suite:
        DHE-RSA-AES256-SHA        256 bits
      […]

* 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/



TLS Support in Browsers

SSL/TLS Browser Support 1/2

http://en.wikipedia.org/wiki/Transport_Layer_Security

Browser support for TLS

Browser	Version	Platforms	TLS 1.0	TLS 1.1	TLS 1.2
Google Chrome [notes 2] [notes 3]	0–21	Android, iOS, Linux, Mac OS X, Windows (XP, Vista, 7, 8)	Yes	No	No
	22–29		Yes[32]	Yes	No[32][33][34][35]
	30–		Yes[32]	Yes[32]	Yes[33][34][35]
Mozilla Firefox [notes 3] [notes 4]	1–22 ESR 10, 17	Android, Linux, Mac OS X, Windows (XP, Vista, 7, 8)	Yes[36]	No[28]	No[30]
	23		Yes[36]	Yes, disabled by default[28][37]	No[30]
	24–26 ESR 24		Yes[36]	Yes, disabled by default[28][37]	Yes, disabled by default[30][38]
	27–		Yes[36]	Yes[28][37][39]	Yes[30][38][39]
Internet Explorer [notes 5]	6	Windows (98, 2000, ME, XP)	Yes, disabled by default	No	No
	7–8	Windows XP	Yes	No	No
	7–9	Windows Vista	Yes	No	No
	8–10	Windows 7	Yes	Yes, disabled by default	Yes, disabled by default
	10	Windows 8	Yes	Yes, disabled by default	Yes, disabled by default
	11	Windows 7, 8.1	Yes	Yes[42]	Yes[42]

SSL/TLS Browser Support 2/2

http://en.wikipedia.org/wiki/Transport_Layer_Security

Opera [notes 6] [notes 7]	5–7	Android,[citation needed] iOS,[citation needed] Linux, Mac OS X, Windows	Yes[46]	No	No
	8–9		Yes	Yes, disabled by default[47]	No
	10–12		Yes	Yes, disabled by default	Yes, disabled by default
	14–16		Yes	Yes[48]	No[48]
	17–		Yes	Yes[49]	Yes[49]
Safari [notes 8]	1–6	Mac OS X –10.8[notes 9]	Yes	No	No
	7	Mac OS X 10.9 [notes 10]	Yes	Yes	Yes
	3–5	iPhone OS 1–3, iOS 4.0[notes 11][notes 9]	Yes[56]	No	No
	5–6	iOS 5–6[notes 11][notes 9]	Yes	Yes	Yes
	7	iOS 7[notes 11][notes 9]	Yes	Yes	Yes
	3–5	Windows	Yes	No	No

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

http://en.wikipedia.org/wiki/Transport_Layer_Security

Cipher security against publicly known feasible attacks

Cipher	Protocol version				
	SSL 2.0	SSL 3.0 [note 1][note 2][note 3]	TLS 1.0 [note 1][note 3]	TLS 1.1 [note 1]	TLS 1.2 [note 1]
AES CBC[note 4]	N/A	N/A	Depends	Secure	Secure
AES GCM[18][note 5]	N/A	N/A	N/A	N/A	Secure
AES CCM[19][note 5]	N/A	N/A	N/A	N/A	Secure
Camellia CBC[20][note 4]	N/A	N/A	Depends	Secure	Secure
Camellia GCM[21][note 5]	N/A	N/A	N/A	N/A	Secure
SEED CBC[22][note 4]	N/A	N/A	Depends	Secure	Secure
ChaCha20+Poly1305[23][note 5]	N/A	N/A	N/A	N/A	Secure
IDEA CBC[note 4][note 6]	Insecure	Depends	Depends	Secure	N/A
Triple DES CBC[note 4][note 7]	Insecure	Depends	Depends	Depends	Depends
DES CBC[note 4][note 6]	Insecure	Insecure	Insecure	Insecure	N/A
RC2 CBC[note 4][note 6]	Insecure	Insecure	Insecure	Insecure	N/A
RC4[note 8]	Insecure	Insecure	Insecure	Insecure	Insecure

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

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

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
```

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 stie!
  - Insert your site into browser HSTS list!

What if they issue a fake certificate?

- Use certificate pinning

Best Attack Vector: Implementation errors

Past implementation erros:

- 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

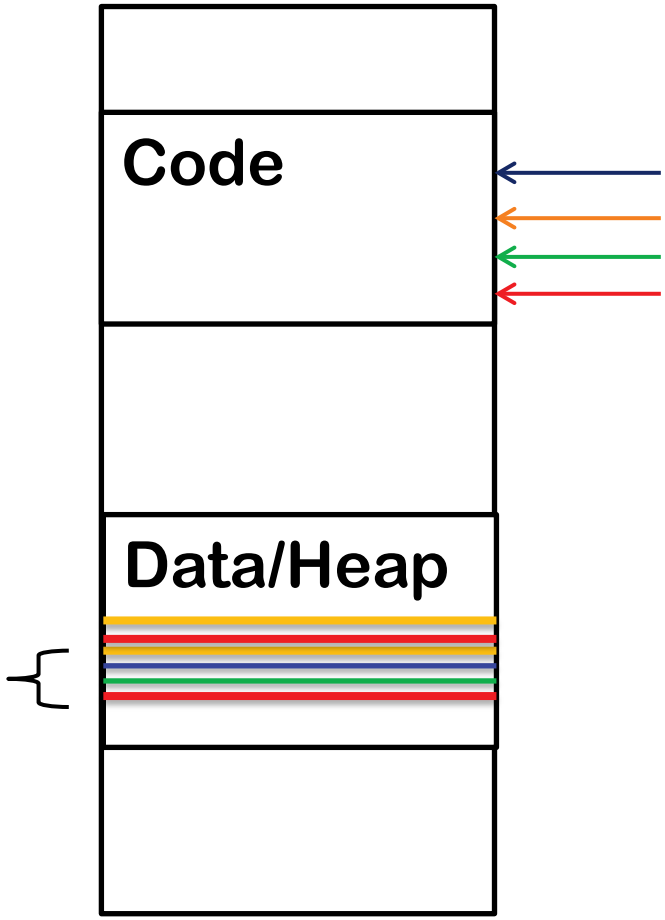
Can be repeated indefinetly

Discloses:

- Sensitive User Data
- Cookies
- Private Keys
- PFS Session Keys
- …

Exploit is public

- Heap Feng Shui?



Apache + OpenSSL Process

Heartbleed

```
209  00000cf0  14 6a eb f8 00 00 00 00  14 6a ed 38 14 6a ed 28  |.j.......j.8.j.(|
210  00000d00  13 e7 6b 64 13 e7 69 1c  13 e7 6b a4 13 e7 65 fc  |..kd..i...k...e.|
211  00000d10  fe 90 5d d8 13 8f 27 a8  00 08 3a a8 fe 90 75 b8  |..]...'...:...u.|
212  00000d20  09 88 1f 10 14 6a ed 88  fe 90 6f 9c 14 6a ed 2c  |.....j....o..j.,|
213  00000d30  14 6a eb 00 14 6a ee b8  14 6a ed 78 14 6a ed 68  |.j...j...j.x.j.h|
214  00000d40  13 8f 29 ac 13 8f 29 1c  13 8f 29 ec 13 8f 25 fc  |..)...)...)...%.|
215  00000d50  00 00 00 00 00 00 00 00  fe 90 3a a8 00 00 00 00  |..........:.....|
216  00000d60  14 6a ed 20 00 a0 42 b0  fe 90 3a a8 00 00 00 00  |.j. ..B...:.....|
217  00000d70  14 6a ed 20 00 37 00 38  fe 90 5d d8 14 6a ed 20  |.j. .7.8..]..j. |
218  00000d80  00 18 ee 38 fe 90 75 b8  09 88 1f 10 14 6a ed f0  |...8..u......j..|
219  00000d90  fe 90 6f 9c 14 6a ed 94  14 6a ee 50 14 6a ee 50  |..o..j...j.P.j.P|
220  00000da0  14 6a ed e0 14 6a ed d0  13 8f 2a 4c 13 8f 29 1c  |.j...j....*L..).|
221  00000db0  13 8f 2a 84 13 8f 25 fc  00 00 00 00 14 6a ed bc  |..*...%......j..|
222  00000dc0  fe 90 3a a8 00 00 00 00  14 6a ed 88 00 6a ed f8  |..:......j...j..|
223  00000dd0  fe 90 3a a8 00 00 00 00  14 6a ed 88 00 e7 65 fc  |..:......j....e.|
224  00000de0  fe 90 7e 78 14 6a ed 88  04 18 3a a8 14 6a ed f0  |..~x.j....:..j..|
225  00000df0  00 00 00 00 14 6a ee 10  09 88 1f 08 00 00 00 00  |.....j..........|
226  00000e00  14 6a ee 20 00 00 00 08  00 00 00 17 09 88 1f 08  |.j. ............|
```

Heartbleed

Warren Guy
@WarrenGuy

Folgen

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

Übersetzung anzeigen

Antworten Retweeten Favorisieren Pocket Mehr

Parker Thompson
@m0thran

Folgen

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.

Adam Langley
@agl__

Folgen

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

Heartbleed

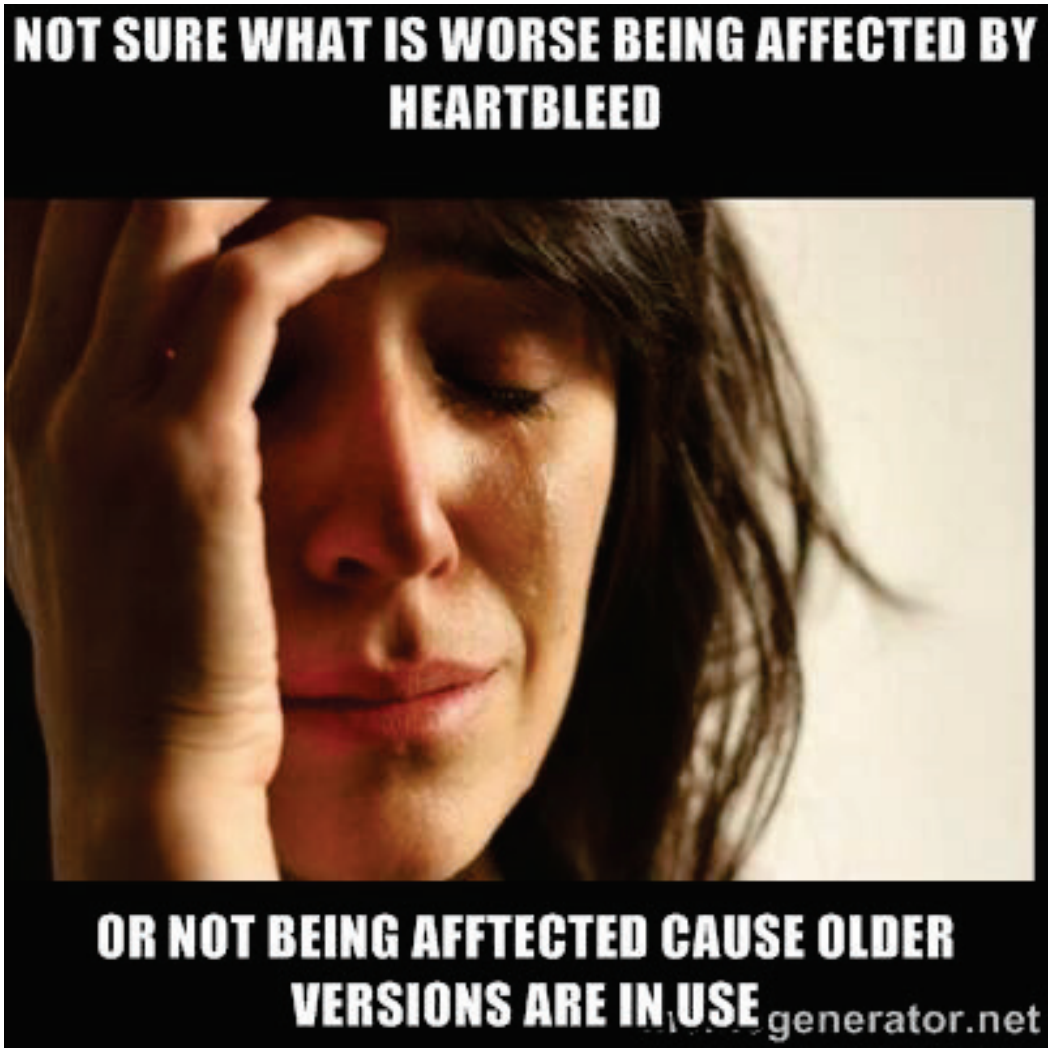
Popular sites which exhibit support for the TLS heartbeat extension include Twitter, GitHub, Yahoo, Tumblr, Steam, DropBox, HypoVereinsbank, PostFinance, Regents Bank,Commonwealth Bank of Australia, and the anonymous search engine DuckDuckGo.

Heartbleed

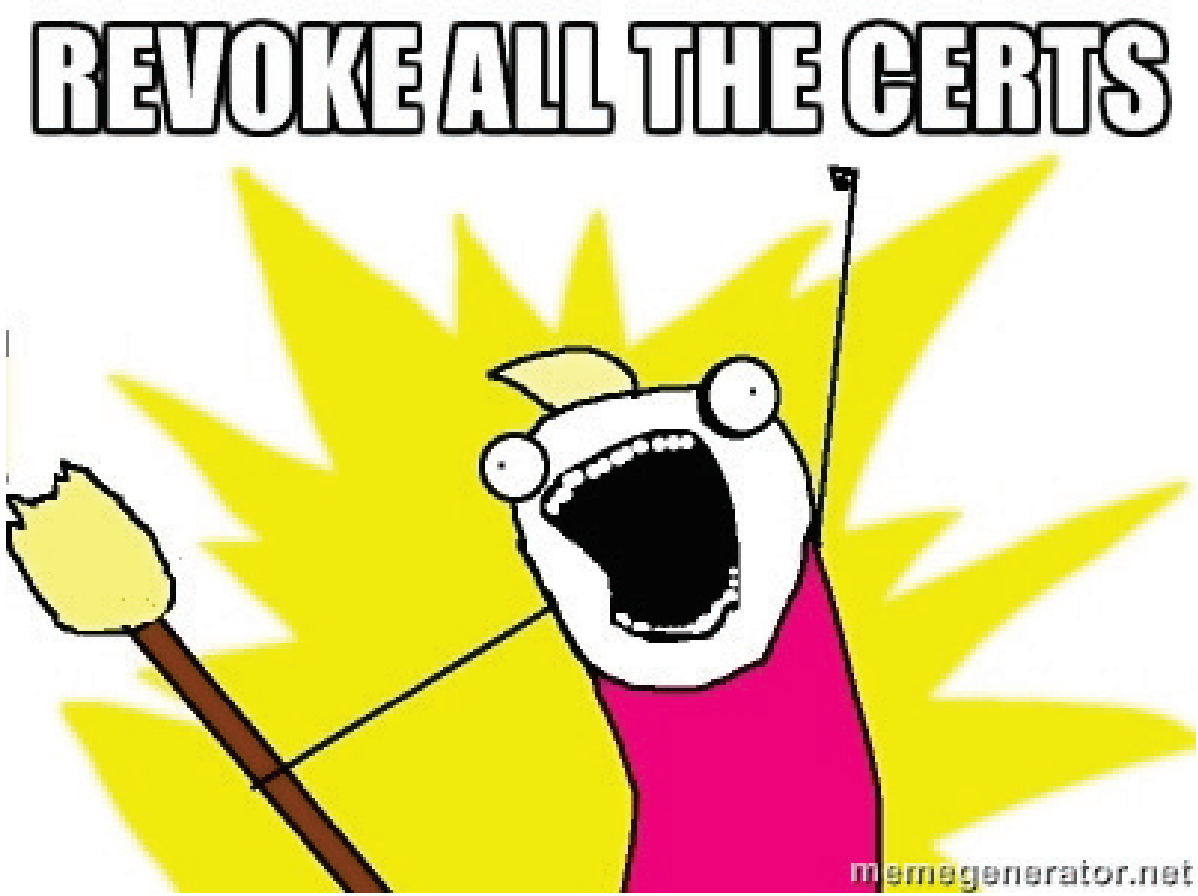
```
-       /* Read type and payload length first */
-       hbtype = *p++;
-       n2s(p, payload);
-       pl = p;
-
        if (s->msg_callback)
                s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
                        &s->s3->rrec.data[0], s->s3->rrec.length,
                        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 AFFTECTED CAUSE OLDER VERSIONS ARE IN USE
generator.net

Heartbleed



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»

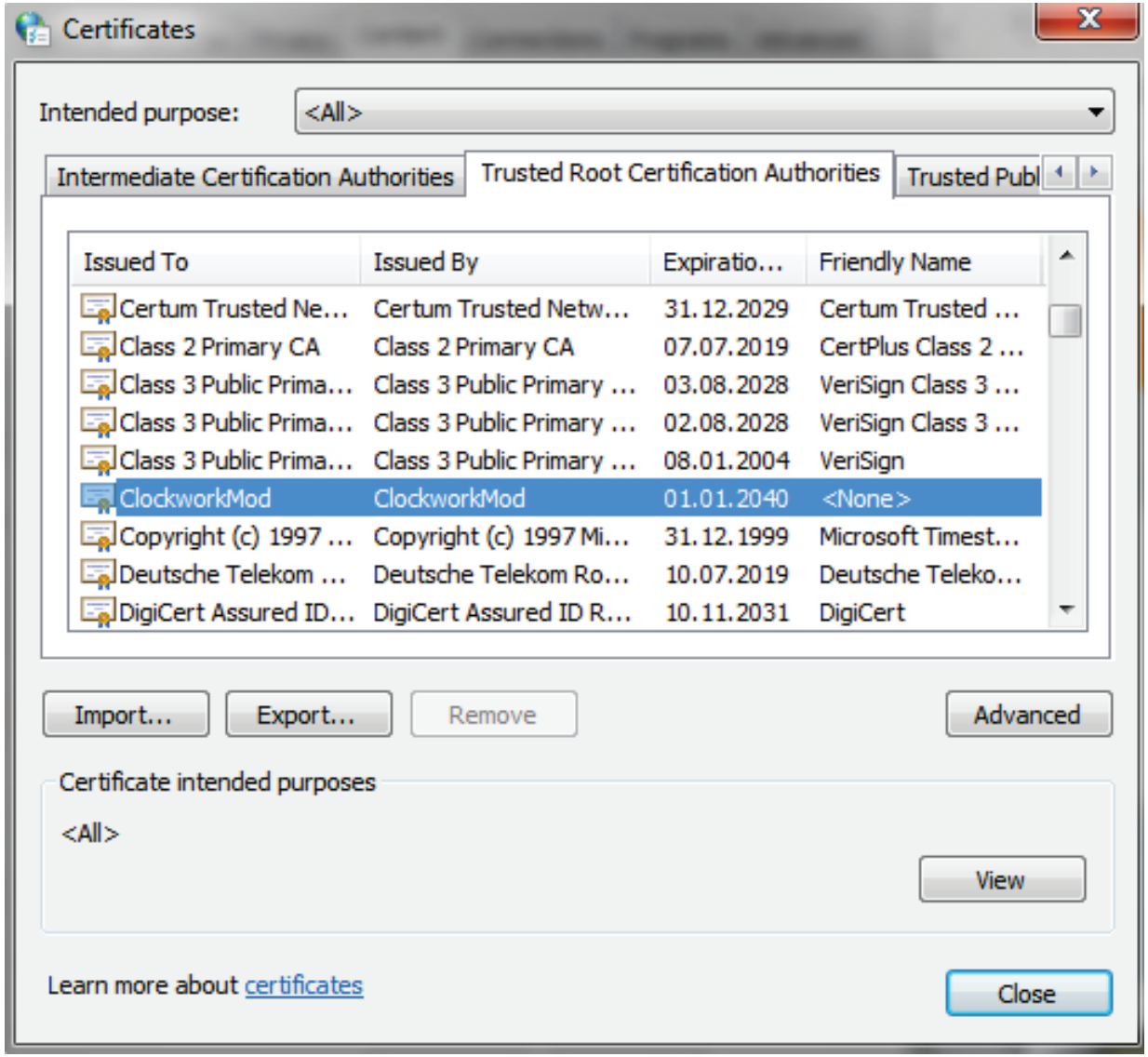
- Lets write A GOOD SSL library
- Now, there are X+1 bad SSL libraries

Source:

- OpenSSL is Open Source
- Pull Request For Heartbeat Support
- No consequent peer review

The CA Problem

The CA Problem



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)
- 11/2011: CA: Einbruch bei KPN

⇒ **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
- DNS Spoofing…

OCSP

- Life check
- What if server is not reachable?
- DNS Spoofing…

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 offical process?

In Windows:

- EMET

In Apps:

- Do it yourself! Very easy!
- Dont 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

- https://www.owasp.org/images/5/55/SSL-in-der-Praxis_OWASP-Stammtisch-Muenchen.pdf

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

- https://www.ssllabs.com/projects/best-practices/

ImperialViolet (Google Chrome Developer Blog)

- https://www.imperialviolet.org/

This presentation is based on the following blog entry:

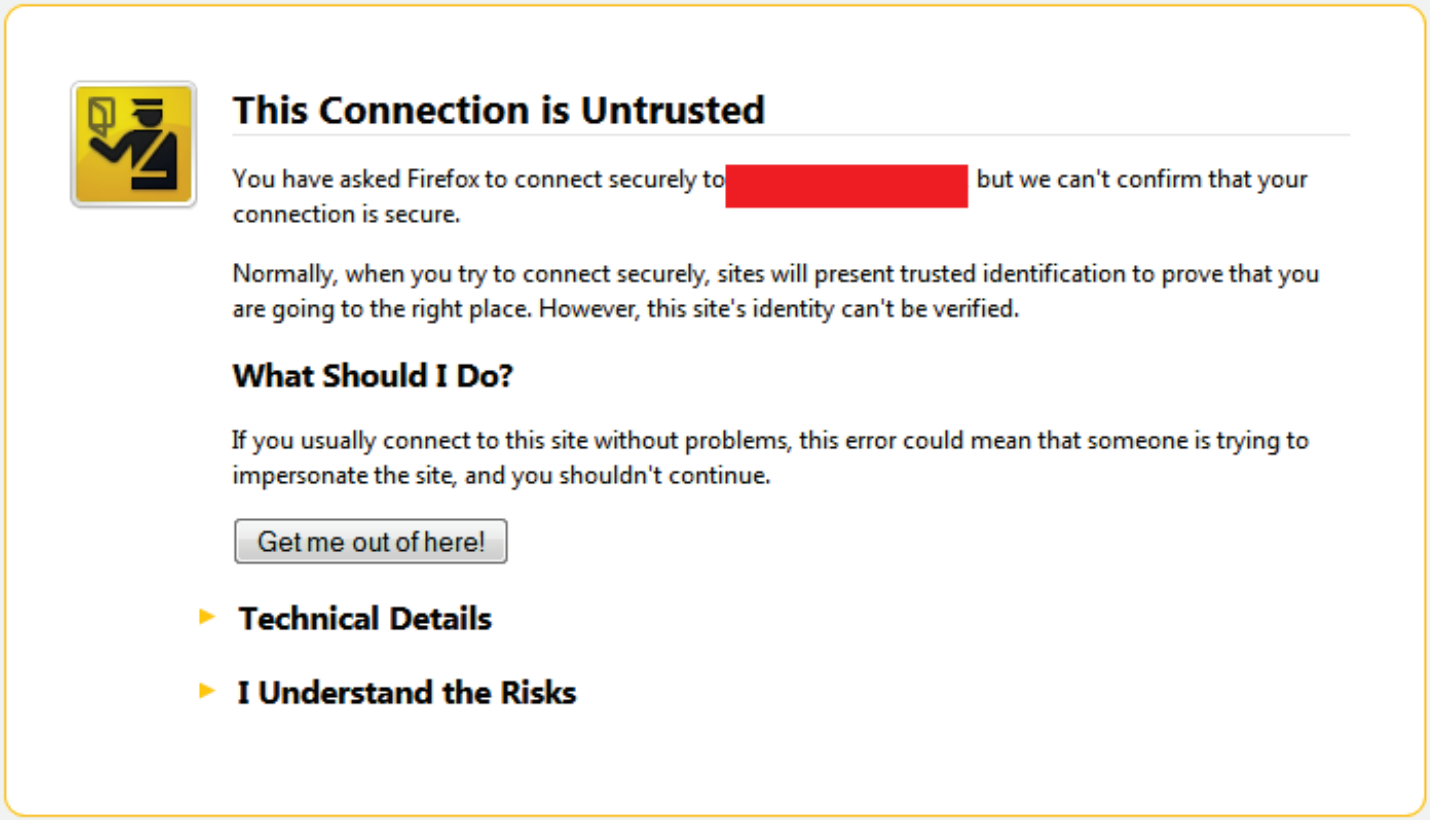
- http://blog.csnc.ch/2013/11/compass-ssltls-recommendations/

Rant:

Browser Indicators

Rant: Browser Indicators

Rant: Browser Indicators



Rant: Browser Indicators