How broken is TLS?

A tale of BEAST, CRIME, Lucky Thirteen and Heartbleed

Hanno Böck, https://hboeck.de

2014-04-19

- Hanno Böck, freelance journalist, lives in Berlin
- Often writes about cryptography for Golem.de and others
- Also runs webhosting (`https://schokokeks.org`)

- TLS is **the** most important crypto protocol
- There are problems on multiple layers (certificate authorities, software, cryptography, protocols)
- BEAST, CRIME, Lucky Thirteen, Heartbleed, ...

- SSL 1 - only internal (Netscape)
- 1994: SSL 2 (Netscape, severe security issues, disabled)
- 1996: SSL 3 (Netscape, no extensions, still used)
- 1999: TLS 1 (IETF standard, problems with CBC, til today de facto standard)
- 2006: TLS 1.1 (half fix for CBC-problems)
- 2008: TLS 1.2 (introduces authenticated encryption with GCM and SHA-2)

- X.509 certificate authority signs host-certificate, host-certificate used to get a session key for TLS
- Two protocols (X.509 and TLS)
- So we have software, CAs, X.509 and TLS itself

- Heartbleed is only a memory read error, just a software bug, nothing with cryptography
- So pretty boring, right?

- SSL libraries have terrible code quality
- OpenSSL Valhalla Rampage - have fun and be scared
- Don't blame OpenSSL. The others are just as bad

- Who needs the heartbeat extension?
- What other extensions are there in TLS and who needs them?
- And who needs GOST, DSA, SEED, IDEA, Brainpool curves or Camellia? (GOST is only a draft, OpenSSL still has code for it)
- TLS also supports DES or RC2 with 40 Bit

- It is extremely difficult to write secure code in C (buffer overflows etc.)
- Which programming language is better? Not too exotic, not too much overhead and usable on many different platforms, but still memsafe?
- And by the way: Constant time implementations needed
- Can we do memsafe C? (Softbount/CETS, OpenBSD malloc)

- Main problem: Every certificate authority can do Man-in-the-Middle-attacks on every website
- You automatically trust all CAs in your browser and all sub CAs
- Weird: It doesn't matter if your CA is trustworthy, only the least trustworthy CA matters

- CA-disaster 2011
- Diginotar, Comodo, Türktrust and others
- EFF SSL Observatory finds many valid certificates that shouldn't exist (e.g. 512 Bit EV certificate)
- Diginotar is the only case where it had consequences
- Comodo issued fake certificates for mail.google.com, www.google.com, login.yahoo.com, login.skype.com, addons.mozilla.org and login.live.com erstellt

- CRL doesn't scale, OCSP not privacy friendly
- OCSP useless in Firefox and IE, disabled in Chrome
- OCSP Stapling could fix things a bit
- OCSP Stapling Required - only a draft
- Revocation does not work right now

- Heartbleed: StartSSL charges for revocation
- This is a problem: It gives incentives to do the wrong thing
- Cynics might say: Doesn't matter, it's broken anyway
- More general problem: It's expensive and difficult to be secure (certificates) - it should be the other way round

- Convergence: distributed access (MitM still possible if attacker near server)
- Sovereign Keys: EFF, complicated, uses append-only log
- TACK: draft for TLS to pin certificates, shares many ideas with Sovereign Keys
- HTTP Key Pinning: draft from Google, only HTTPS
- Certificate Transparency: from Google, append-only log, make sure MitM gets detected

- Three public key types for X.509 certificates: RSA, DSA, ECDSA
- Everyone uses RSA and that's a good thing
- DSA/ECDSA have severe problems with bad randomness

- MD5 signatures were used until someone showed they are really broken
- SHA1 signatures are de facto standard
- Few CAs do SHA256 signatures (about to change)
- RSA signatures use old PKCS #1 1.5 scheme, RSA-PSS (PKCS #1 2.1) would be better, lacks software support

- TLS supports a lot of algorithm combinations
- Example: ECDHE-RSA-AES256-GCM-SHA384
- Key exchange with elliptic curves, signature with RSA, encryption with AES, key size 256 bit, block mode GCM (Galois/Counter Mode), MAC algorithm SHA385

- Until TLS 1.0 AES CBC used an implicit initialization vector
- This led to the BEAST attack
- Fixed in TLS 1.1, but who implements better protocols with more security if they don't have to?

- TLS needs confidentiality and authenticity
- Most common: Encryption with AES-CBC, authentication with HMAC
- TLS does MAC-then-Pad-then-Encrypt
- Different error messages for padding and MAC errors allow Padding Oracle attack
- "Solution": just one error message

”Canvel et al. [CBCTIME] have demonstrated a timing attack on CBC padding based on the time required to compute the MAC. In order to defend against this attack, implementations MUST ensure that record processing time is essentially the same whether or not the padding is correct. In general, the best way to do this is to compute the MAC even if the padding is incorrect, and only then reject the packet. For instance, if the pad appears to be incorrect, the implementation might assume a zero-length pad and then compute the MAC. **This leaves a small timing channel**, since MAC performance depends to some extent on the size of the data fragment, **but it is not believed to be large enough to be exploitable**, due to the large block size of existing MACs and the small size of the timing signal.” (TLS 1.2, RFC 5246, p. 23)

- To translate this: We know there is a problem, but we don't think it's a real problem
- That was the Lucky Thirteen attack - it was already mentioned in the TLS standard itself

- For all problems with BEAST and Lucky Thirteen there are workarounds in browsers
- But after Lucky Thirteen many wanted to avoid CBC
- Solution: RC4 (only non-CBC-algorithm left before TLS 1.2)
- PCI verification (credit card standard) required RC4 for some time

- RC4 was developed 1994 by Ron Rivest, not for public use
- Source was published in a newsgroup, is only a few lines long and you can copy-and-paste it from Wikipedia
- March 2013: Bernstein et al show an impractical attack on RC4 in TLS, no workarounds
- After Snowden some people speculate that NSA can attack RC4 in real time
- Today many people think RC4 has to die (draft)

- RC4 or CBC with MAC-then-Encrypt? Both are bad, but for CBC we have workarounds
- BEAST, Lucky Thirteen, RC4-attacks are all not very practical, but this is not good
- AES-GCM: Authenticated encryption, only in TLS 1.2, only new browsers
- Encrypt-then-MAC for CBC: draft
- ChaCha20/Poly1394: draft - very fast and probably very secure cipher

- Forward secrecy is great! If you do it right
- but...
- Apache before 2.4.7 only supports Diffie-Hellman (DHE) with 1024 bit, Java until 1.7 the same
- And there are elliptic curves... (ECDHE)

- Idea: Diffie-Hellman and El Gamal/DSA use a "group" (some math structure)
- You can use any group you like
- Assumption: If you use an elliptic curve that's much more secure (the best known attacks on discrete logarithms don't work in elliptic curves)
- There are infinitely many elliptic curves - which one should we use?

- Solution: We ask the NSA - they have lots of skilled cryptographers, they should know
- In 1999 NIST published some good, well tested elliptic curves
- The author works for the NSA
- Where does 3045ae6f c8422f64 ed579528 d38120ea e12196d5 come from?
- It is not very likely that the NSA has a backdoor, but we can't completely rule it out

- Dan Bernstein developed a better elliptic curve: Curve25519, 255 bit
- Draft for TLS
- The SafeCurves project also has longer curves
- ECDSA doesn't work and is bad anyway, there is Ed25519 for Curve25519

- Compression leaks information about content and leads to side channel attacks (CRIME attack)
- Just disable it, your browser won't use it anyway
- Problem: Similar attacks work on HTTP compression, workarounds are tricky and have to happen in the web application (BREACH attack)

- TLS has an internal downgrade protection
- But browsers have a "workaround" for this protection: Try a lower protocol if the higher one fails
- Problem: Broken middleware and load balancers are everywhere
- Downgrades can happen with bad connections (mobile Internet)
- This is a problem for SNI (ServerNameIndication) and other extensions - please disable SSLv3 everywhere

- BIG-IP from F5 is such a story...
- One of the reasons for those "workarounds"
- Also there are devices that don't like TLS handshakes between 256 and 512 bytes, so we now have a TLS padding extension to add some useless data
- But there is other hardware that breaks with the padding extensions (Ironware SMTP servers from Cisco)
- Browsers do workarounds for broken stuff all the time
- There's now a draft for better downgrade protection (a workaround for a workaround for broken stuff)

- Frankencerts - take a bunch of real certificates, change things randomly
- And then check if they are valid
- In a perfect world every library should come to the same conclusion
- In the real world they don't
- This uncovered severe bugs in MatrixSSL and GnuTLS and smaller bugs in various other TLS implementations

- Dual EC DRBG has almost certainly a backdoor from the NSA
- Research project: We replace the backdoor parameters with our own parameters
- This revealed private keys for DSA and ECDSA in TLS (told you so: don't use them)
- RSA BSAFE used Dual EC by default
- The TLS extended random extension makes this attack easier, there exists a draft and code in RSA BSAFE (learn: don't invent extensions without a clear purpose. Oh, we already knew that from Heartbleed)
- Dual EC rarely used, so no real problem, but exciting research

- Bug in the logic of TLS handshakes and renegotiations
- Browser workaround, Adam Langley has a check for your browser
- Interesting fact: some browsers accept insecure parameters for Diffie-Hellman
- Diffie-Hellman needs a **large** prime - some browsers think 17 or 15 are large primes

- SSL Stripping: Attacker prevents HTTPS by intercepting links or first HTTP access
- HSTS header forces browser to use TLS and forbids HTTP to same host
- Some browsers have build-in HSTS domains
- WARNING: There is no way to do hybrid HTTPS/HTTP solutions secure. Just protecting the login is wrong

- ASN.1 is the most horrible binary format to parse
- Know any good software to handle it? I don't

——BEGIN CERTIFICATE——
MIICPDCCAaUCED9pHoGc8JpK83P/uUii5N0w
DQYJKoZIhvcNAQEFBQAwXzELMAkG

- Quantum computers kill all public key and key exchange algorithms in TLS
- 2012 Nobel price for research that could help building quantum computers
- Post quantum cryptography is very experimental, no algorithm that could be deployed easily
- Quantum cryptography won't safe you, it is a nice thought experiment, but it is completely impractical

- Use latest Apache
- Support TLS 1.2 with AES-GCM and forward secrecy
- Disable RC4, TLS compression, SSLv3 and all mostly unused algorithms (EXPORT, NULL, SEED, IDEA, ...)
- Enable OCSP stapling
- Use RSA certificates with 2048 or 4096 bit, signed with SHA256
- Use the Qualys SSL test

- a) TLS is broken beyond repair, we need to get rid of it and make something better
- b) TLS is broken on many layers, but we have ideas and fixes for many problems, we need to get them deployed
- Choose your own conclusion - and help making it happen

- I often write about crypto on Golem.de (German) - `http://hboeck.de`
- Matthew Green‘s Blog `http://blog.cryptographyengineering.com/`
- Adam Langley‘s Blog `https://www.imperialviolet.org/`
- Dan Bernstein‘s Blog (DJB) `http://blog.cr.yp.to/`
- Very good online course by Dan Boneh on Coursera `https://www.coursera.org/course/crypto`

- TLS 1.2, RFC 5246 `http://www.ietf.org/rfc/rfc5246.txt`
- Heartbleed `http://heartbleed.com/`
- OpenSSL Valhalla Rampage `http://opensslrampage.org/`
- EFF SSL Observatory `https://www.eff.org/observatory`
- Internet scans `https://scans.io/`
- Sovereign Keys `https://www.eff.org/sovereign-keys`
- Convergence `http://convergence.io/`
- TACK `http://tack.io/`
- HTTP Key Pinning `http://tools.ietf.org/html/draft-ietf-websec-key-pinning-01`
- Certificate Transparency `http://certificate-transparency.org`

- OCSP Stapling Required http://tools.ietf.org/html/draft-hallambaker-tlssecuritypolicy-03
- MD5 broken http://media.ccc.de/browse/congress/2008/25c3-3023-en-making_the_theoretical_possible.html
- RSA-PSS thesis https://rsapss.hboeck.de/
- Padding Oracle http://www.iacr.org/cryptodb/archive/2002/EUROCRYPT/2850/2850.pdf
- BEAST http://ekoparty.org/2011/juliano-rizzo.php
- Lucky Thirteen http://www.isg.rhul.ac.uk/tls/Lucky13.html
- RC4 attack http://www.isg.rhul.ac.uk/tls/
- Prohibiting RC4 draft https://tools.ietf.org/html/draft-popov-tls-prohibiting-rc4-02
- BREACH http://breachattack.com/

- SafeCurves http://safecurves.cr.yp.to/
- Frankencerts https://www.cs.utexas.edu/~shmat/shmat_oak14.pdf
- Dual EC http://dualec.org/ https://projectbullrun.org/dual-ec/
- Triple Handshake https://secure-resumption.com/
- Browser check for Triple Handshake https://www.imperialviolet.org/2014/03/03/triplehandshake.html
- Browser check for DH parameters https://dh.tlsfun.de/
- Post quantum cryptography http://pqcrypto.org/
- TLS – Nuke it from Orbit http://clearcryptocode.org/tls/
- Qualys SSL-Test https://www.ssllabs.com/ssltest/

- Thank you Edward Snowden!