To Make Hearts Bleed

A (Native) Developer's Account On SSL

Daniel Molkentin
daniel@molkentin.de

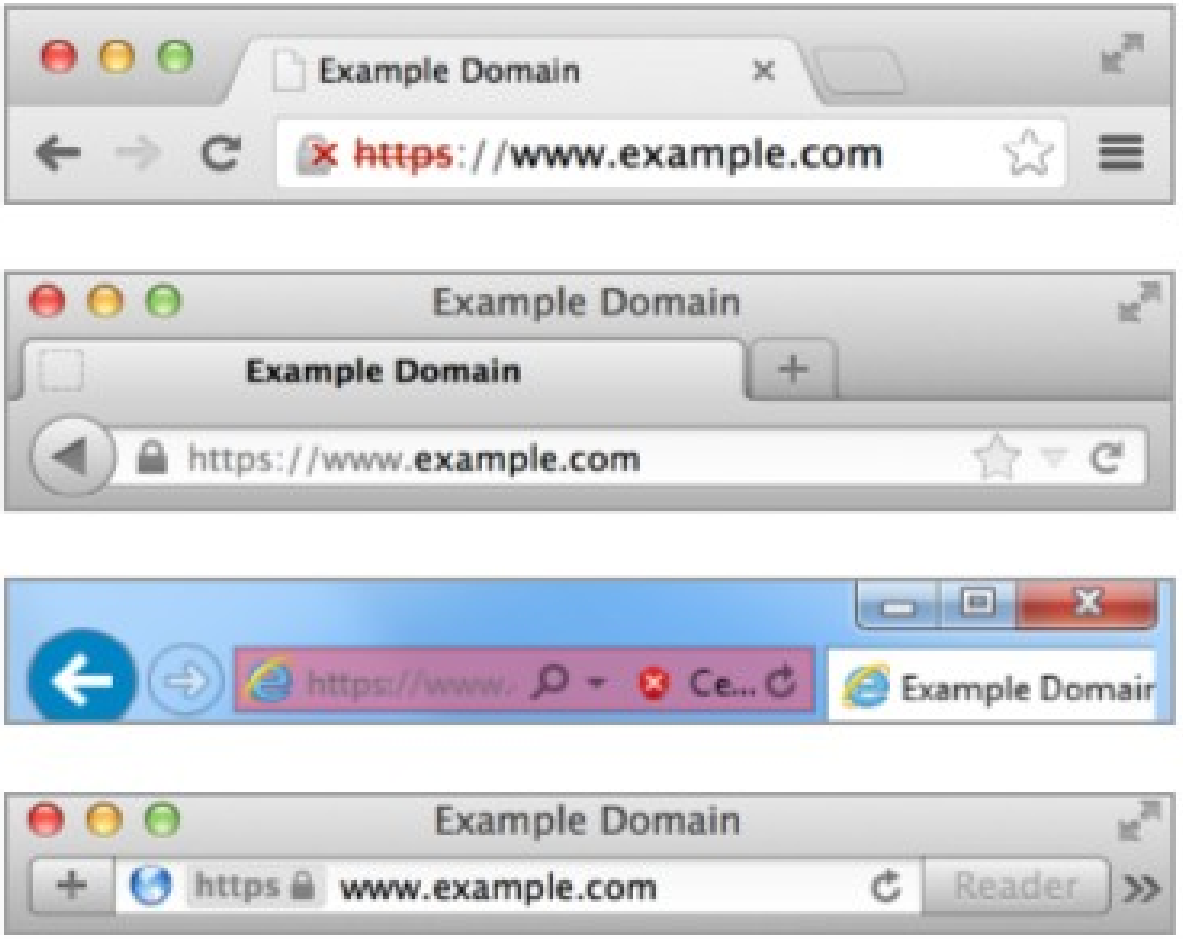
About me

- Daniel Molkentin
- Senior Software Engineer for ownCloud Inc
  - The opinions expressed in this talk are mine
- Working on the desktop client
  - Quite some SSL
- In another/related life: Contributor/Reviewer to the Qt networking stack (Particularly: Qt SSL)
  - Even more SSL
- Guy with a mission, not a “security expert”!

What we need to talk about

- Why the current way developers and sysadmins go about security is suboptimal
- What you can do about it
- What the security community is doing about it
- What nobody is doing (and why)
- How to stop this

Why we need to talk



Most peoples' HTTPS web sites, in different browsers

Scope of this Talk

- Targeted at both software developers and sysadmins
- Not a lecture on computer security
- Tries to explain the details necessary to follow along
- Not a comprehensive overview of all that's wrong
- May destroy all illusions you might have had on SSL (if any)

Please ask questions as they come up, but try to defer discussions until Q&A.

SSL/TLS in a Nutshell

- Introduced 1994 as SSL 2.0 by Netscape (proprietary, unreviewed, broken design, “export mode”) to encrypt connections, “authenticate” domains
  - Uses X.509 (directory oriented, no extensions)
  - X.509 uses ASN.1 (“Unfortunately Complicated”)
- SSL 3.0: complete rewrite, better, still proprietary (1995)
- TLS 1.0: IETF-controlled, 3DES mandatory, wider range of cipher suites, uses finalized X.509 (1999)
- Fields: Subject, Issuer, Public Key, Extensions (e.g. SAN)
- Signed by CA (or intermediate), trusted by browsers → avoids MITM attacks
- Simple! (In theory….)

ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128:AES256:AES:DES-CBC3-SHA:HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK

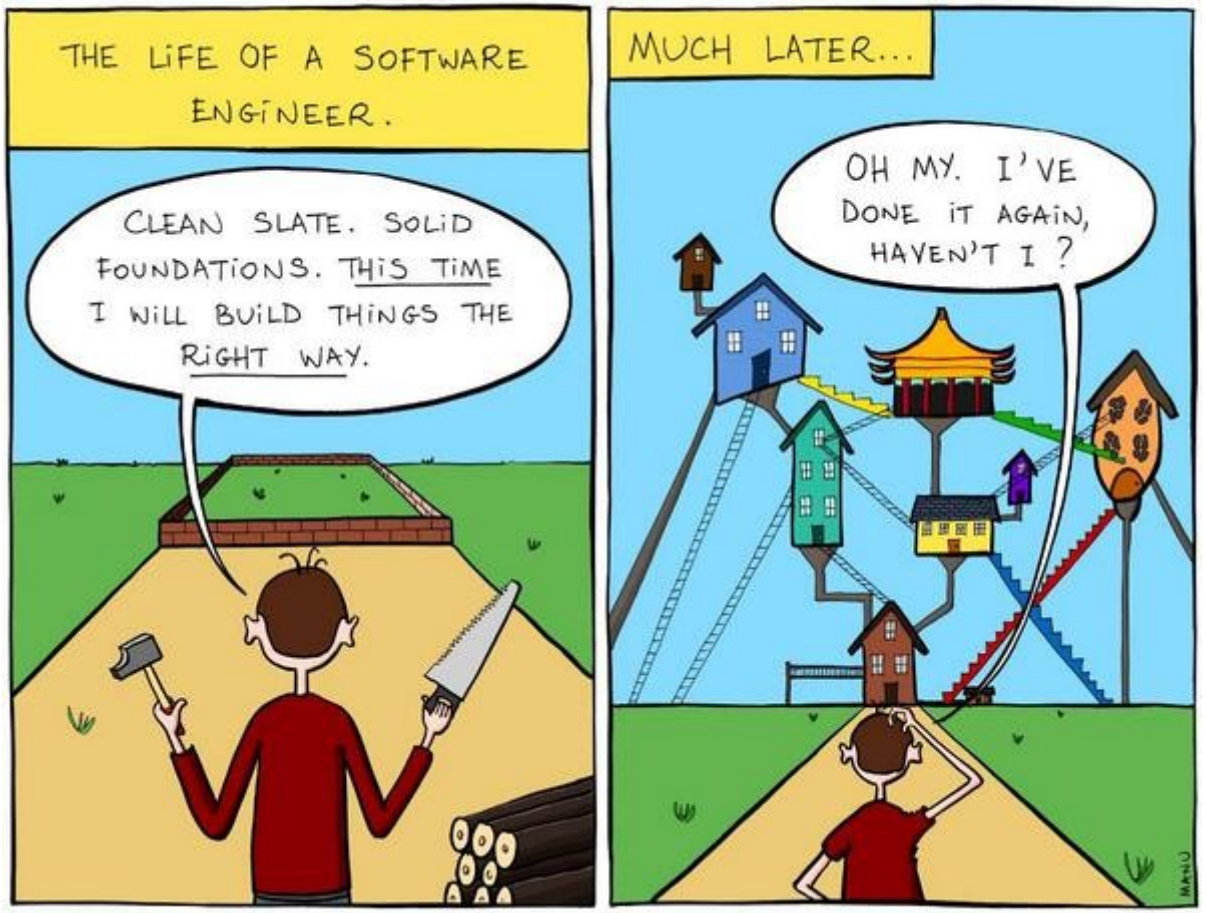
Immediate Issues

- Any CA can sign for *any* domain
  - State Orgs from questionable countries
  - Walt Disney! (`CN=The Walt Disney Company CA`, Fingerprint: `885da88c44`)
  - Pending: “Honest Achmed's Used Cars and Certificates”
- Ooops, the key has leaked
  - Remedies: CRL (several MB already), OCSP, CT, ...
  - Browsers don't check
  - Blacklist, not a whitelist
  - Much has been tried, nothing works **today**
- SSL “Virtual Hosts” still not well understood
- Self-signed certificates (Testing, “Only for me anyway”, <Enter lame excuse>)
  - Use EasyRSA: https://github.com/OpenVPN/easy-rsa
- Poor Server Configuration (Enables SSLv2, weak ciphers, NULL ciphers)
- Bad practices (Mixed content, partial use of SSL, Session Cookies leak via HTTP)
- Not following Best Practices (HTTP downgrade possible!, Intermediate Certificate missing)

X.509

- ITU Standard, meant to be used with X.500 (today: LDAP)
- Uses ASN.1
  - Encodings: **BER, CER, DER**, XER…
  - Can contain arbitrary fields
  - Fields can contain arbitrary data (e.g. photos, mp3s,…)
  - Extremely easy to get wrong → Security implications
- Common Name / Subject Alternative Name fields
  - Can contain an IP(s), or hostname(s)
  - daniel.molkentin.net, *.molkentin.net
  - 88.198.13.78
  - *.198.13.78 (this used to work!)
  - Should *.foo.bar match baz.fooish.foo.bar?
  - IDN problems all over again ('ü' !=== 'ü')

SSL Sucks! Kill it with Fire!



Fixing the Problem? DANE

```
$  dig +dnssec +noall +answer +multi _443._tcp.www.bund.de TLSA
 _443._tcp.www.bund.de.   900 IN TLSA 3 0 1 (
8F28B062EAA9F917042A63D35D99E017C68D89EAA314
C49A3EF94B6E770B0A49 )
_443._tcp.www.bund.de.    900 IN RRSIG TLSA 7 5 900 (
20140919120001 20140909120001 35264 bund.de.
XBWrIEhXtWX6tJbmhqrVXrtZGiNNyhwdHwRsiuvWxZ7V
jEyVpRNfBhgZK0mG4hC2xyLNT4n2+fW6N42Pb/FwTQXC
9cvYIBb61xiJxl2V2DICf4PGCPUM03hEC8XyZdGVGhPb
CXiA/mi5NIqLnc03YsSaXL7DR87iGUfQPt4Za3M= )
```

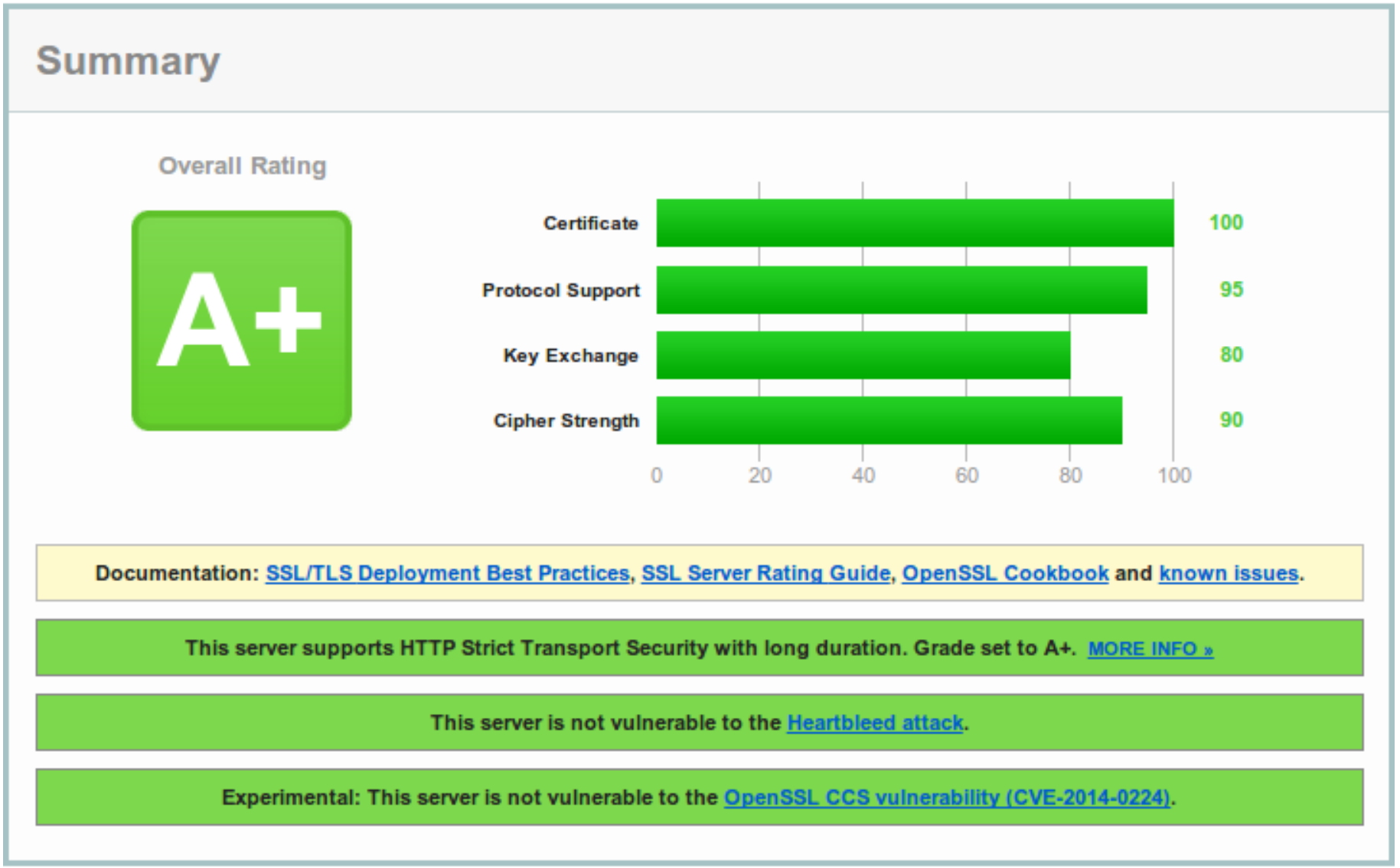
3: Domain Issued Certificate, 0: Full Certificate, 1: SHA256 hash

- Based on DNSSEC: Add certificate hash to (signed) DNS zone
- Tricky (certificate replacement!)
- Few DNS providers / DNS setups have DNSSEC-signed zones
- What about Intranets?
- No browser support (plugin required)
- No convenient API/Framework support

What to do in the Meanwhile?

- In the meanwhile: Get a *valid* certificate (really!). → StartSSL, GlobalSign (for OSS)
- Send ALL the certificates !1!! (incl. intermediates)
- Use good cipher suites: https://wiki.mozilla.org/Security/Server_Side_TLS
- Fix downgrade, HTTP-connect attacks
  - HTTP Strict transport Security header
  - "`Strict-Transport-Security: max-age=31536000; includeSubDomains`"
- Certificate Pinning
  - Only possible for Browser-Vendor (Chrome!) or developers (at the moment)
  - Make sure to pin sensible attributes
  - Static list of a handful of selected sites
  - Flawed: "The 80's called, they want their `hosts.txt` back!"
- Test your configuration: http://ssllabs.com, `cipherscan`, `sslyze`

Your Reward



slllabs.com, showing results for daniel.molkentin.net

What Sysadmins cannot fix

- Security vs. Decree of Availability
- OCSP (Online Certification Status Protocol)
  - Leaks browsing history to CA
  - States: “Good”, “Revoked”, “*Unknown*”
  - Server may not respond
  - Replay attack possible
  - Enter: *OCSP Stapling (Hello Kerberos Tickets!)*
    - Nginx implementation unreliable
    - Only the leaf cert may be stapled (WiP)
    - `OCSP-Must-Staple` not a standard
    - Not available in Apache 2.2, nginx < 1.3.7
    - Needs explicit configuration in later version
      - IIS had this in Windows Server 2008, enabled by default (!)
      - Which brings us right to…

BSDs and Linux “Stable”/”LTS”/ ”Enterprise” Distributions

- Most ship OpenSSL from the medieval ages (0.9.8)
- Only sometimes, the vendor actually relinks against later versions (RHEL/CentOS >= 6.5)
- Only Apache 2.2 available
  - No OCSP stapling (see last slide)
  - 1024 DH only
  - Session Caching, Session Tickets a mess

Call To Action

- Sysadmins
  - Fix your security up to Best Practices
  - Track & Implement improvements to certificate security (OCSP stapling, etc)
  - Don't stick with old distros / Use backports!
- Distributions
  - Promote the use of up-to-date security
  - Upgrade Components & re-certify if necessary

All good now?

- We only have discussed HTTP servers and browsers
- We did not discuss other servers (mail, ldap, etc)
  - Same principles apply!
- Time to talk about
  - Desktop Applications
  - Apps
  - Your treasured bash scripts (wget, curl!)

Leaving the Browser's Nest

- Modern SSL APIs are designed to write browsers
- They do not *offer the comfort* of a browser
  - You've got to roll your own…
    - Self-signed-cert handling code
    - Warning dialogs
    - Mixed mode handling
  - Auto-Updates
    - Are you using transport security?
    - Are you checking signatures?
  - Cross-Platform without a Toolkit? Outch!
  - Sometimes caught in OSS
  - Closed Source? Who knows…

So you're writing an app for that?

- Chances are that you're doing it wrong
- Pfahl & al, Hannover University
  - Analyzed most popular 13,500 Google play store apps for Android
  - Captured CC#, Twitter, Google, Yahoo, IBM Sametime Accounts
  - Forged Antivirus signature updates (!)
- Common Problems
  - Trusting all Certs
  - Allowing all host names with a valid Cert
  - Trusting all CAs even in ancient Android versions (DigiNotar!)
  - Allows Mixed Mode content

But I'm writing for iOS!

- “Apple reviews apps, broken SSL implementations will be caught in review, right?”
- “I'm so, so sorry” (Pfahl & al strike again, 2013)
  - 1009 cherry-picked apps from the App Store
  - 98 were vulnerable
  - 254 gave false error message when attacked
  - 287 did simply not connect
- Curated App Store is not a guarantee

Causes

- Only basic SSL APIs available
- Improper understanding of underlying technology
- “Programming by stackoverflow”
- Popular Mobile Development Frameworks disable certificate checks by default
- Certificate checks implemented, disabled for development, never re-armed

Remedies

- API / Framework developers need to offer complete solutions, not a basic API
- In the meanwhile: Developers should treat SSL like it was your business domain
  - If you don't understand it, learn about it
  - There is good documentation out there
  - Read it!
- Pen test your application

(Bash) Scripts

- Your Server uses self-signed certs
- You need to retrieve SSL data from a script
- Easy: `curl -k`!
  - Oops, you just opened up widely for MITMs
- Remedy:
  - Create a custom CA, roll out your CA
    - There are a number of good CA software kits
    - There is CACert
  - Get a paid-for certificate
  - For large server farms: Puppet, Chef, CFEngine exist

The Summary

But:

Never surrender,
Never give up,
Or THEY will win.

Daniel Molkentin <daniel@molkentin.de>

Thanks to Richard Moore of Westpoint Security for reviewing these slides

References

- SSL – Paved with good Intentions: http://www.westpoint.ltd.uk/papers/ssl-paved-with-good-intentions.pdf
- 20 Years of SSL/TLS Research: http://www-brs.ub.ruhr-uni-bochum.de/netahtml/HSS/Diss/MeyerChristopher/diss.pdf
- The Case of OCSP-Must-Staple: https://www.ccsl.carleton.ca/paper-archive/sobey-esorics-08.pdf
- Bullet Proof SSL and TLS: https://www.feistyduck.com/books/bulletproof-ssl-and-tls/
- OpenSSL Cookbook: https://www.feistyduck.com/books/openssl-cookbook/
- Rethinking SSL: http://cryptome.org/2014/08/rethinking-ssl.pdf
- An Analysis of Android SSL (In)Security: http://www2.dcsec.uni-hannover.de/files/android/p50-fahl.pdf
- DANE Demonstration (from ICANN 49 DNSSEC Workshop): https://singapore49.icann.org/en/schedule/wed-dnssec/presentation-dnssec-dane-26mar14-en.pdf
- Verisign DANE/TLSA tools: http://dane.verisignlabs.com/

Addendum: FIPS

- Federal Information Processing Standard
- More precisely FIPS 140-2 deals with SSL
- “U.S. government computer security standard used to accredit cryptographic modules”
- Requirement if you want to do business with the US administration
- SecureTransport/SChannel are certified
- OpenSSL: FIPS mode exists, lots of strings attached
- Certification is expensive (50k+)
- OpenSSL drew money from FIPS-related services → reason to keep their codebase as unchanged as possible