Bettercrypto - Applied Crypto Hardening for Sysadmins

Aaron Kaplan kaplan@cert.at, Aaron Zauner azet@azet.org,
David Durvaux david@autopsit.org

2015-06-15

Part 1a: Intro to the project

bettercrypto·org

Overview

1. **Part 1a:** Intro to bettercrypto & Motivation (aaron)
2. How we got started, how we work, what's there, what's missing, how to use the guide
3. Brainstorming round: what's needed for the organisation?
4. **Part 1b:** Background (david)
5. History of Crypto in a nutshell
6. Theory
7. Practical settings
8. **Part 1c:** Testing, tools, finding a process for improvement
9. Brainstorming round: finding an internal testing strategy
10. **Part 2:** introduction to and survey of TLS Security (azet)

Prerequisites

- Participants should have a basic knowledge of System administration and be familiar with configuring Apache, nginx, etc.
- know git/github
- a basic knowledge of cryptography will help.

Motivation


NATIONAL SECURITY AGENCY
PEEPING WHILE YOU'RE SLEEPING
The NSA
The only part of government
that actually listens.

Motivation (2)

This was a wake up call.

Whatever you might think about the Snowden leaks, please note:

- the leaks also revealed to all countries worldwide precise recipies on how to do country wide or even Internet-wide surveillance, traffic inspection and -modification, etc.
- If politicians in other countries did not know how to do this, now they know!
- If criminals did not know how to do this, now they know!
- The leaks revealed to **every one** "tradecraft" secrets. This makes the whole world an unsafer place!
- => What can we still do then? => Crypto++!

Motivation (3)

Ende-zu-Ende-Verschlüsselung ist das Einzige, das funktioniert. Leitungen verschlüsseln ist auch schwierig, es gibt keine Standards. Und das Problem mit ausländischen Anbietern bleibt. Bereits beim Endkunden verschlüsseln, alles andere wird nicht helfen.

— Klaus Landefeld, Beirat DE-CIX Management GmbH 2015-03-26, NSA Untersuchungsausschuß im deutschen Bundestag

The reaction

Don't give them anything for free

It's your home, you fight!

The reaction (2)

But...

- We as humans are used to certain **modes** in communications: spoken words tend to:
  - be forgotten over time ("data expires")
  - get modified/changed whenever "copied" (repeated)
  - get changed/modified over time ("forgetfullness")
  - we tend to be not so harsh about them ("forgive")
  - have a limited geographic range ("town talk")
  - be very decentralized ("accoustic range")
- digital traces/data tends to be:
  - stored for ever. Never modified by default
  - used against you in the future
  - very centralized
  - copied very easily

The reaction (3)

Crypto is the only thing that might still help

a.k.a.:

"*The Bottom Line Is That Encryption Does Work*"

— Edward Snowden

But where is there any advice?

- Ca. August 2013: Adi Kriegisch asks Aaron Kaplan where he could find good recommendations on SSL settings.
- Does that exist? At that time:
  - no ssllabs cookbook
  - only theoretical recommendations (ENISA, eCrypt II, NIST)
  - ioerror's duraconf settings are outdated
  - no practical copy & paste-able settings exist?

Project plan

So we have a plan :)

Project plan (srsly)

- Do at least something against the **Cryptocalypse**
- Check SSL, SSH, PGP crypto Settings in the most common services and certificates:
  - Apache, nginx, lighttpd
  - IMAP/POP servers (dovecot, cyrus, . . .) – openssl.conf
  - Etc.
- Write down our experiences as guide
- Create easy, copy & paste-able settings which are "OK" (as far as we know) for sysadmins.
- Keep the guide short. There are many good recommendations out there written by cryptographers for cryptographers
- Many eyes must check this!
- Make it open source

Why is this relevant for you?

- You run networks and services. These are targets. If you believe it or not.
- You produce code. Make sure it uses good crypto coding practices
- However good crypto is hard to achieve
- Crypto does not solve all problems, but it helps
- There are no secure defaults

Who?

Wolfgang Breyha (uni VIE), David Durvaux, Tobias Dussa (KIT-CERT), L. Aaron Kaplan (CERT.at), Christian Mock (coretec), Daniel Kovacic (A-Trust), Manuel Koschuch (FH Campus Wien), Adi Kriegisch (VRVis), Ramin Sabet (A-Trust), Aaron Zauner (azet.org), Pepi Zawodsky (maclemon.at), IAIK, A-Sit, and many more. . .

Contents so far

- Intro
- Disclaimer
- Methods
- Theory
  - Elliptic Curve Cryptography
  - Keylengths
  - Random Number Generators
  - Cipher suites – general overview & how to choose one
- Recommendations on practical settings
- Tools
- Links
- Appendix

Methods and Principles

C.O.S.H.E.R principle:

- **C**ompletely
- **O**pen
- **S**ource
- **H**eaders
- **E**ngineering and
- **R**esearch

Methods:

- Public review
- commits get **discussed**
- recommendations **need** references (like wikipedia)
- Every commit gets logged & we need your review!

How to commit

- `https://git.bettercrypto.org` (master, read-only)
- `https://github.com/BetterCrypto/` (please clone this one & send PRs)

How?

1. discuss the changes first on the mailinglist
2. clone
3. follow the templates
4. send pull requests
5. **split the commit into many smaller commits**
6. don't be cross if something does not get accepted.
7. be ready for discussion

Feedback from professional cryptographers

- multiple times mentioned in talks by Dan J. Bernstein & Tanja Lange
  (ONE Conference, CCC, . . .)
- good initial feedback from Vincent Rijmen (inventor of AES)
- got invited to the IETF STRINT workshop 2014
- As of May 2015: Bettercrypt is the basis for RFC7525

Your situation

A large organisation has its own needs

Brainstorming: what's needed in your organisation?

(interactive session)

Some points to get us thinking:

- What are the issues you have encounted with running more crypto?
- Which legacy systems exist? Can they be updated ?
- How can you test all of your systems if they use strong crypto?
- Is there an inventory of all services and servers?
- Which services can be tested from outside?
- Which only from inside?
- Which interfaces exist to outside organisations using crypto?
- ..... interfaces to ... , which should use crypto?

What's needed in your organisation? (2)

(continued)

- Are there any protection rings / classifications on different sets of information?
- Are there any automatic processes using SSL/crypto which can get disturbed by
  updates?
- How to test these processes if they work?
- ***Key-roll-overs***: are there procedures for this? What happens when upgrading to 4k?
- Do not underestimate the amount of work for key-management

Proposed meta-strategy

- practice key-roll-overs
- practice identifying all services which run crypto
- practice testing them against known good standards automatically (nagios, . . .)
- practive crypto config changes

Turns out, key management as well as crypto management can be seen similarly to regular patch management: it needs periodic attention.

Part 1b: Background

over to David...

History, Theory

History part

Pre-history

- Scytale (7h century BC)
- Caesar
- Vigenère (in a cifra del. Sig. Giovan Battista Bellaso, 1533)

How you can loose your head

- Mary Queen of Scots (1542 - 1587)
  - Queen of Scotland until 1567
  - Try to regain the throne
  - Was found guilty of plotting to assassinate Queen Elizabeth I of England
  - Proven after her code get broken...

How it can change a war

- World War II
  - Enigma in use by German Army
  - Broken by the first computer (Alan Turing)
  - Sign the end of U-Boat supremacy on the sea

A sort of Steganography

- Navajos Code Talkers (Pacific War - US Navy)

Nowadays

- Asymmetric cryptography
  - RSA (Rivest - Shamir - Adleman) - 1977
  - GPG (Phil Zimmerman) - 1991
- AES (Rijndael) - 1998

Famous names

- Cryptography was an hot topic for a lot of people
  - Thomas Jefferson (1790) - ciphering cylinder (used for 150 years)
  - Charles Babbage - break the Vigenère Cipher (1854, unknown until 20th Century)
  - Gilbert S. Vernam (AT&T, 1917) - polyalphabetic cipher with random key without repetition
  - Only cipher suite impossible to break both in theory and in practice!

Theory

$$\hbar \frac{\partial}{\partial t} \Psi = \hat{H} \Psi$$

Some thoughts on ECC

- Currently this is under heavy debate
- Trust the Math
  - eg. NIST P-256 (`http://safecurves.cr.yp.to/rigid.html`)
  - Coefficients generated by hashing the unexplained seed c49d3608 86e70493 6a6678e1 139d26b7 819f7e90.
- Might have to change settings tomorrow
- Most Applications only work with NIST-Curves
- Bottom line: we leave the choice of ECC yes or no to the reader. You might have to adapt again.
- However, many server operators tend towards ECC for speed

Keylengths

- `http://www.keylength.com/`
- Recommended Keylengths, Hashing algorithms, etc.
- Currently:
  - RSA: >= 3248 bits (Ecrypt II)
  - ECC: >= 256
  - SHA 2+ (SHA 256,...)
  - AES 128 is good enough

AES 128? Is that enough?

„On the choice between AES256 and AES128: I would never consider using AES256, just like I don't wear a helmet when I sit inside my car. It's too much bother for the epsilon improvement in security."

— Vincent Rijmen in a personal mail exchange Dec 2013

- Some theoretical attacks on AES-256

(Perfect) Forward Secrecy

Problem:

- Three letter agency (TLA) records all encrypted traffic
- Someday TLA gains access to private-key (Brute Force, Physical Force)
- TLA can decrypt all recorded traffic

Solution:

- **Ephemeral** session keys via Diffie Hellman (**DHE**)

Review of Diffie Hellman

Let g be a primitive root mod p. p is a Prime.

Alice to Bob:

$$X = g^x \mod p$$

Bob to Alice:

$$Y = g^y \mod p$$

Alice calculates:

$$k_1 = Y^x \mod p$$

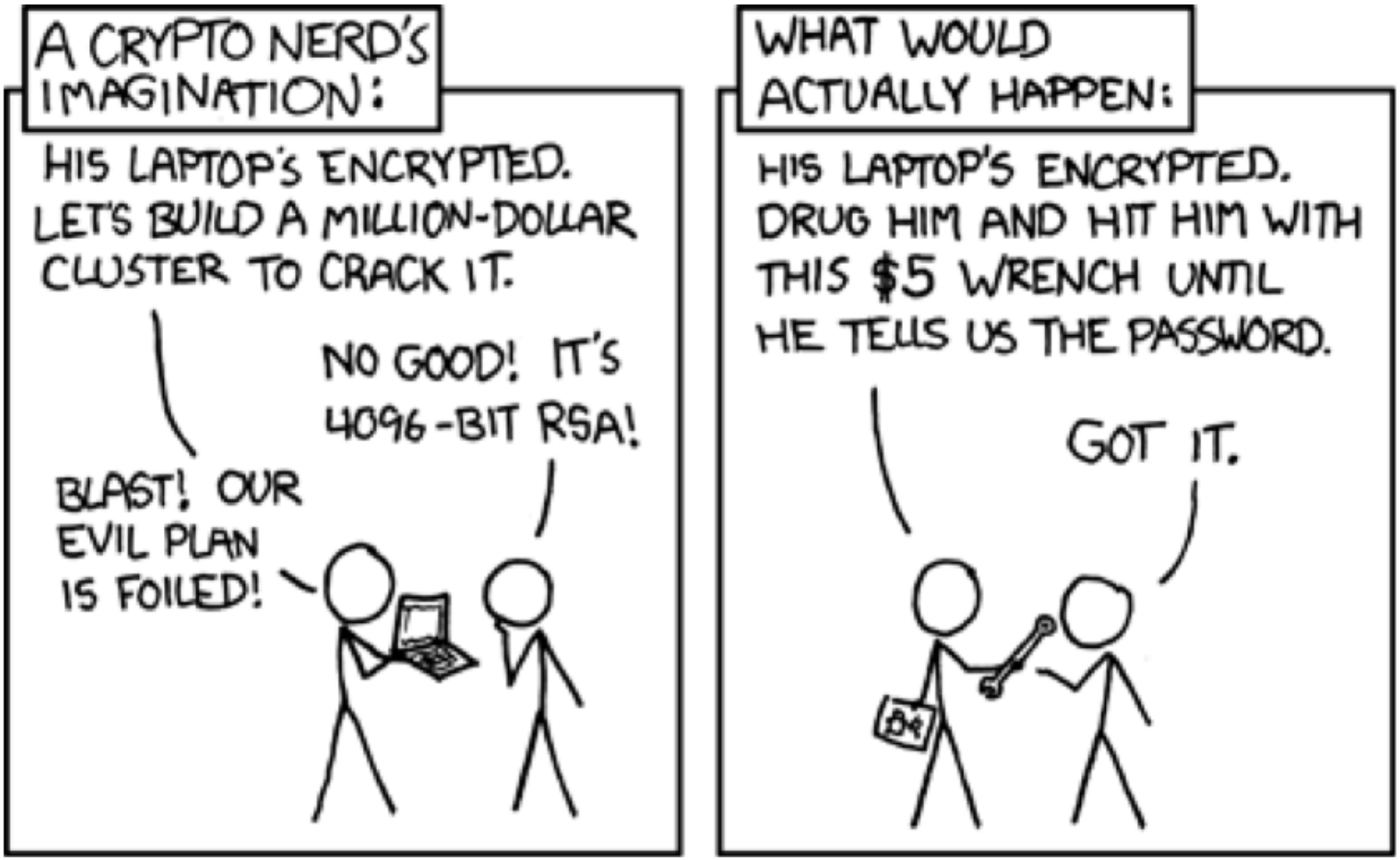
Bob calculates:

$$k_2 = X^y \mod p.. \text{ Therefore, } k_1 = k_2$$

Proof:

$$k_1 = Y^x = (g^y)^x = g^{(x \times y)} = (g^x)^y = X^y = k_2 \mod p \quad \square$$

Reality


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Well...

We recommend perfect forward secrecy.

- Ephemeral: new key for each execution of a key exchange process
- TLS private-key only for authentication
- Alternative new private key every x days/months
- Pro:
  - Highest security against future attacks
- Contra:
  - Elliptic Curves needed for broad client support
  - slight processing overhead

(P)RNGs

- (P)RNGs **are** important!
- Nadia Heninger et al / Lenstra et al
  "... to identify apparently vulnerable devices from 27 manufacturers."

	Our TLS Scan		Our SSH Scans	
Number of live hosts	12,828,613	(100.00%)	10,216,363	(100.00%)
...using repeated keys	7,770,232	(60.50%)	6,642,222	(65.00%)
...using vulnerable repeated keys	714,243	(5.57%)	981,166	(9.60%)
...using default certificates or default keys	670,391	(5.23%)		
...using low-entropy repeated keys	43,852	(0.34%)		
...using RSA keys we could factor	64,081	(0.50%)	2,459	(0.03%)
...using DSA keys we could compromise			105,728	(1.03%)
...using Debian weak keys	4,147	(0.03%)	53,141	(0.52%)
...using 512-bit RSA keys	123,038	(0.96%)	8,459	(0.08%)
...identified as a vulnerable device model	985,031	(7.68%)	1,070,522	(10.48%)
...model using low-entropy repeated keys	314,640	(2.45%)		

- Entropy after startup: embedded devices and VMs very bad

(P)RNGs - recommendations

- Look out for known weak RNG
  - Dual EC_DRBG is weak (slow, used in RSA-toolkit)
  - Intel RNG ? Recommendation: add System-Entropy (Network). Entropy only increases.
- Use tools (e.g. haveged/HaveGE `http://dl.acm.org/citation.cfm?id=945516`)
- RTFM
  - when is the router key generated
  - Default Keys?
- Re-generate keys regularly
- Do not generate keys on fresh VMs.
- Always generate new keys when refreshing certificates

Cipher suites

- What is a SSLCipherSuite?
- vs. SSLProtocol
- Example:
  SSLProtocol All -SSLv2 -SSLv3
  SSLCipherSuite
  'EDH+CAMELLIA:EDH+aRSA:EECDH+aRSA+AESGCM:EECDH+a
  SHA:AES256-SHA:CAMELLIA128-SHA:AES128-SHA'
  Names are not consistent between official IANA strings and most libraries. They are easily mixed up, always double check.

Some general thoughts on settings

- General:
  - Disable SSL 2.0 (weak protocol and algorithms)
  - Disable SSL 3.0 (BEAST, POODLE)
  - Disable RC4 cipher (RFC7465)
  - Disable EXPORT suites (FREAK Attack)
  - Enable TLS 1.0 or better
  - Disable TLS-Compression (SSL-CRIME Attack)
  - Implement HSTS (HTTP Strict Transport Security)
  - Implement OCSP stapling (Security and performance improvement)
- Variant A: fewer supported clients
- Variant B: more clients, weaker settings

Attacks only get better.

Variant **A**

EECDH+aRSA+AES256 : EDH+aRSA+AES256 : ! SSLv3

ID	OpenSSL Name	Version	KeyEx	Auth	Cipher	Hash
0xC030	ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH	RSA	AESGCM(256)	AEAD
0xC028	ECDHE-RSA-AES256-SHA384	TLSv1.2	ECDH	RSA	AES(256)	SHA384
0x009F	DHE-RSA-AES256-GCM-SHA384	TLSv1.2	DH	RSA	AESGCM(256)	AEAD
0x006B	DHE-RSA-AES256-SHA256	TLSv1.2	DH	RSA	AES(256)	SHA256

Compatibility:

Only clients which support TLS1.2 are covered by these cipher suites (Chrome 30, Win 7 and Win 8.1, Opera 17, OpenSSL >= 1.0.1e, Safari 6/iOS 5, Safari 7/OS X 10.9)
Excellent for controlled environments, like intranet.

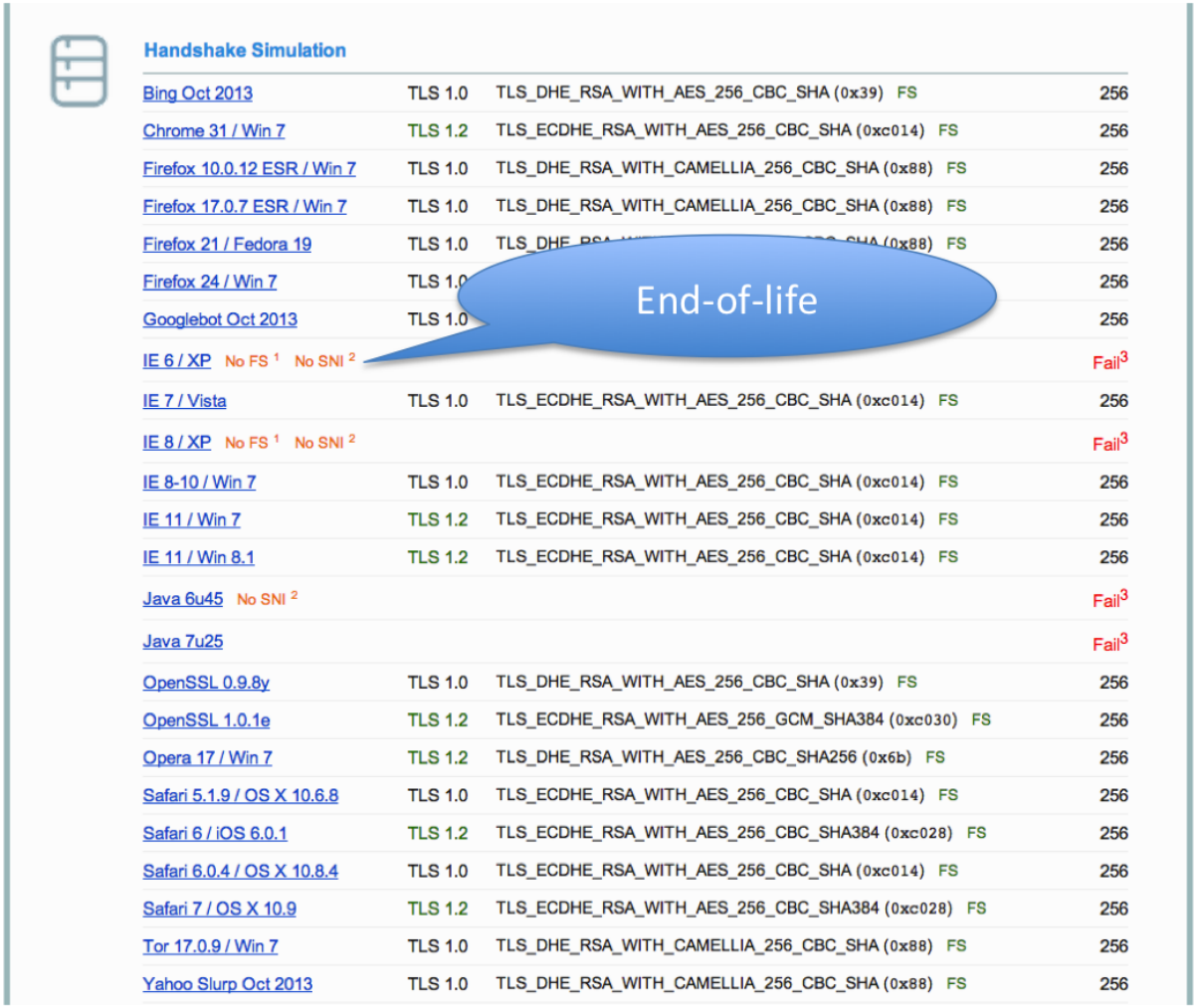
Variant B

- weaker ciphers, broad client support

```
'EECDH+aRSA+AESGCM:EECDH+aRSA+SHA384:EECDH+aRSA+SHA256:EDH+CAMELLIA256:EECDH:
    EDH+aRSA:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!SRP:!DSS:!RC4:!SEED
    :!AES128:!CAMELLIA128:!ECDSA:AES256-SHA'
```

ID	OpenSSL Name	Version	KeyEx	Auth	Cipher	Hash
0xC030	ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH	RSA	AESGCM(256)	AEAD
0xC028	ECDHE-RSA-AES256-SHA384	TLSv1.2	ECDH	RSA	AES(256)	SHA384
0x009F	DHE-RSA-AES256-GCM-SHA384	TLSv1.2	DH	RSA	AESGCM(256)	AEAD
0x006B	DHE-RSA-AES256-SHA256	TLSv1.2	DH	RSA	AES(256)	SHA256
0x0088	DHE-RSA-CAMELLIA256-SHA	SSLv3	DH	RSA	Camellia(256)	SHA1
0xC014	ECDHE-RSA-AES256-SHA	SSLv3	ECDH	RSA	AES(256)	SHA1
0x0039	DHE-RSA-AES256-SHA	SSLv3	DH	RSA	AES(256)	SHA1
0x0035	AES256-SHA	SSLv3	RSA	RSA	AES(256)	SHA1

Variant B compatibility



Handshake Simulation

Bing Oct 2013	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) FS	256
Chrome 31 / Win 7	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
Firefox 10.0.12 ESR / Win 7	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256
Firefox 17.0.7 ESR / Win 7	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256
Firefox 21 / Fedora 19	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256
Firefox 24 / Win 7	TLS 1.0		256
Googlebot Oct 2013	TLS 1.0		256
IE 6 / XP No FS [1] No SNI [2]			Fail[3]
IE 7 / Vista	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
IE 8 / XP No FS [1] No SNI [2]			Fail[3]
IE 8-10 / Win 7	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
IE 11 / Win 7	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
IE 11 / Win 8.1	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
Java 6u45 No SNI [2]			Fail[3]
Java 7u25			Fail[3]
OpenSSL 0.9.8y	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) FS	256
OpenSSL 1.0.1e	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) FS	256
Opera 17 / Win 7	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x6b) FS	256
Safari 5.1.9 / OS X 10.6.8	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
Safari 6 / iOS 6.0.1	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) FS	256
Safari 6.0.4 / OS X 10.8.4	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
Safari 7 / OS X 10.9	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) FS	256
Tor 17.0.9 / Win 7	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256
Yahoo Slurp Oct 2013	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256

Choosing your own CipherSuite string

- Rolling your own cipher suite string involves a trade-off between:
  - Compatibility (server <-> client), vs.
  - Known weak ciphers/hashes/MACs
  - The choice ECC or not, vs.
  - Support by different ssl libs (gnutls, openssl,...) vs.
  - Different versions of ssl libs
- In case of ssl lib version issues: do you want to re-compile the whole server for a newer version?
- Be aware of these issues before choosing your own cipher suite. Have test suites!

Choosing your own CipherSuite string (2)

- Complexity
- It is a multi-dimensional optimisation problem
- Consider strong alternatives to de-facto standards (pros/cons - CAMELLIA vs. AES)
- *WISHLIST*: generator for settings? click-dropdown boxes on the webserver -> generate config
- *WISHLIST*: right now we only support OpenSSL CipherSuite names/configs. What about gnutls, etc.?

Practical settings

What we have so far

- Web server: Apache, nginx, MS IIS, lighttpd
- Mail: Dovecot, cyrus, Postfix, Exim
- DBs: Mysql, Oracle, Postgresql, DB2
- VPN: OpenVPN, IPSec, Checkpoint, ...
- Proxies: Squid, Pound
- GnuPG
- SSH
- IM servers (jabber, irc)
- *DANE* (this section is still WIP)
- *Configuration code snippets*

What are we missing

WISHLIST:

- Section on generating CSRs (`-sha256` etc)
- Mail: Exchange, Sendmail
- SIP
- RDP
- Everything as HTML/TXT (easier to copy & paste)
- gnutls settings
- Config generator on the website
- Automatic testing suite

Example Apache

- Selecting cipher suites:

```
SSLProtocol All -SSLv2 -SSLv3
SSLHonorCipherOrder On
SSLCompression off
# Add six earth month HSTS header for all users...
Header add Strict-Transport-Security "max-age=15768000"
# If you want to protect all subdomains, use the following header
# ALL subdomains HAVE TO support https if you use this!
# Strict-Transport-Security: max-age=15768000 ; includeSubDomains

SSLCipherSuite 'EECDH+aRSA+AESGCM:EECDH+aRSA+SHA384:EECDH+aRSA+SHA256:EDH
   +CAMELLIA256:EECDH:EDH+aRSA:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP
   :!PSK:!SRP:!DSS:!RC4:!SEED:!AES128:!CAMELLIA128:!ECDHA:AES256-SHA'
```

- Additionally mod_rewrite:

```
<VirtualHost *:80>
 #...
 RewriteEngine On
      RewriteRule ^.*$ https://%{SERVER_NAME}%{REQUEST_URI} [L,R=
         permanent]
 #...
```

Testing

How to test - Tools

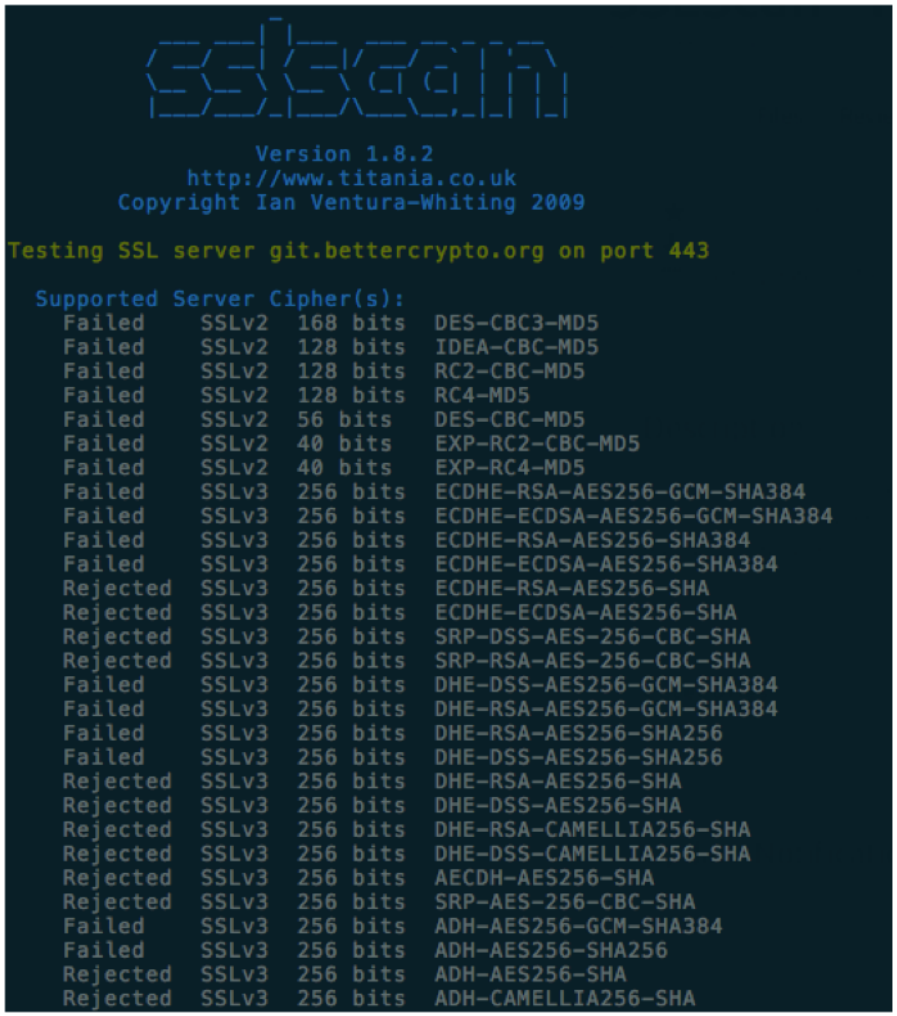
- openssl s_client (or gnutls-cli)
- **ssllabs.com**: checks for servers as well as clients
- xmpp.net
- sslscan: for internal scans
- SSLyze: for internal scans
- masscan: for internal scans
- nmap: for internal scans

Tools: openssl s_client

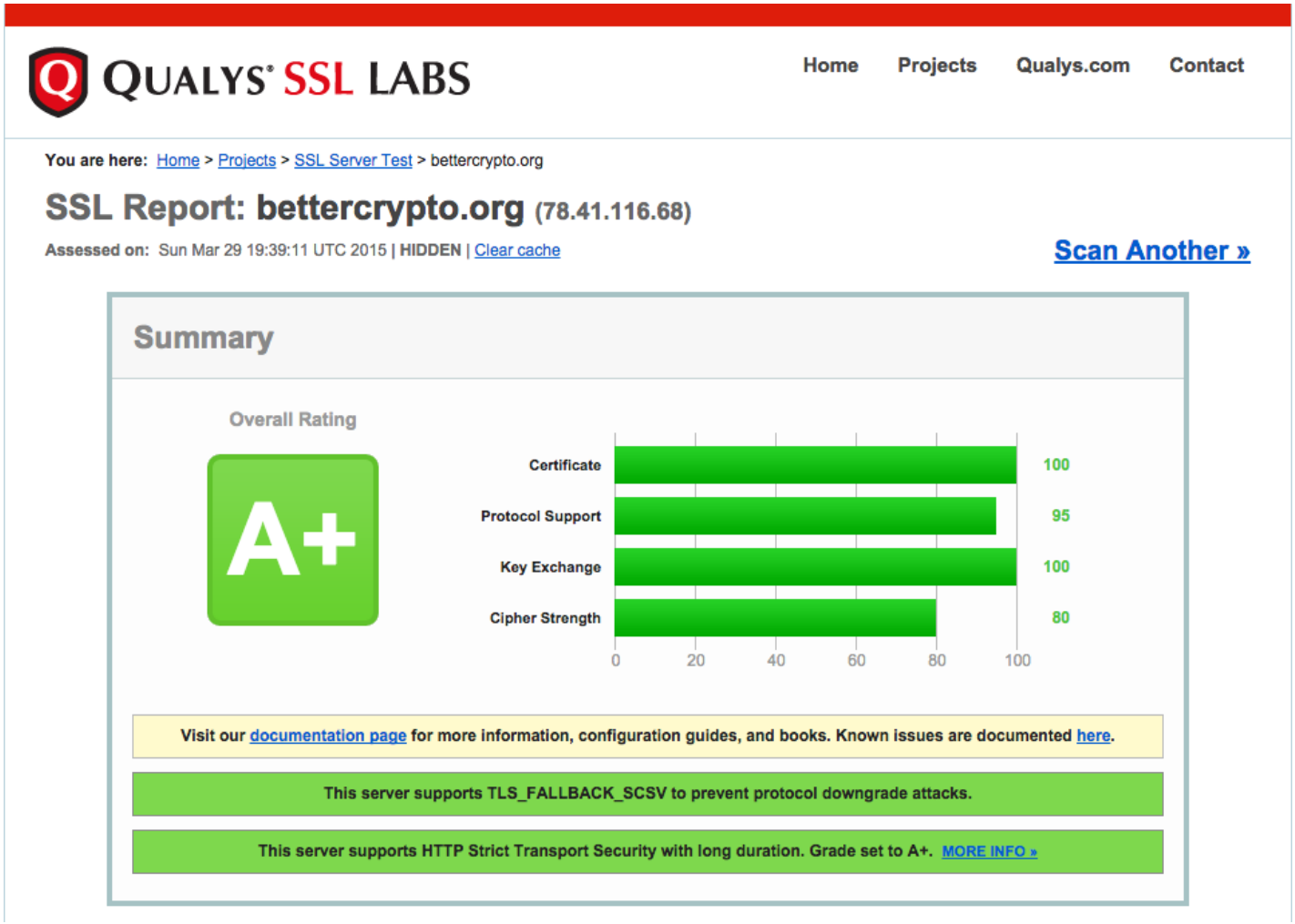
openssl s_client -showcerts –connect git.bettercrypto.org:443

```
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 4096 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: 53D90B7D9D1FFC7EA98C105A2FC27F752B9CE9026CDAB57F4A7D4491C3C5ECC6
    Session-ID-ctx:
    Master-Key: 8F06DE9669BD6BF9628A38DF4F92C2CEBA6B7EA91F465164440CF31F7E8F55F2A67E7320B388D6E7AC4BC141C2FF3F68
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - fe 5b 93 84 a8 c6 ab 4a-74 b8 59 81 dc 3e 52 40   .[.....Jt.Y..>R@
    0010 - 0e dd f6 59 b4 a1 d2 54-65 df 9a 1b c9 fb 0d 2e   ...Y...Te.......
    0020 - 64 9c 65 cf 1c 0d d9 19-57 a6 cd 50 a5 d9 16 a4   d.e.....W..P....
    0030 - 17 b6 e8 38 ac e5 76 15-a4 9d d5 62 ee 51 55 09   ...8..v....b.QU.
    0040 - 52 36 58 84 04 0f 93 94-7b a9 dc e3 6f 8e 2f 7a   R6X.....{...o./z
    0050 - 9f bf 3d 4f a1 e1 bb 83-21 0f 7d f2 bd 02 48 a6   ..=O....!.}...H.
    0060 - 5a 96 82 fd dc a6 5a 55-77 b3 9f fb 60 0d 86 66   Z.....ZUw...`..f
    0070 - f1 68 42 e2 90 93 8b f6-25 aa 85 cf 08 07 c6 76   .hB.....%......v
    0080 - 06 62 37 32 09 4f ac 23-28 9c db b9 29 c0 23 1b   .b72.O.#(...).#.
    0090 - e4 c3 d2 a3 a4 b4 87 b5-0e 5c 68 16 73 07 96 90   .........\h.s...

    Start Time: 1385118946
    Timeout   : 300 (sec)
    Verify return code: 21 (unable to verify the first certificate)
---
```

Tools: sslscan



Tools: ssllabs.com



Tools: sslllabs.com (2)

Configuration

Protocols

TLS 1.2	Yes
TLS 1.1	Yes
TLS 1.0	Yes
SSL 3	No
SSL 2	No

Cipher Suites (SSL 3+ suites in server-preferred order, then SSL 2 suites where used)

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) ECDH 256 bits (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) ECDH 256 bits (eq. 3072 bits RSA) FS	256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) DH 4096 bits (p: 512, g: 1, Ys: 512) FS	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x6b) DH 4096 bits (p: 512, g: 1, Ys: 512) FS	256
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) DH 4096 bits (p: 512, g: 1, Ys: 512) FS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) ECDH 256 bits (eq. 3072 bits RSA) FS	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) DH 4096 bits (p: 512, g: 1, Ys: 512) FS	256
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	256

Tools: sslllabs.com (3)

Handshake Simulation

Bing Oct 2013	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) FS	256
Chrome 31 / Win 7	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
Firefox 10.0.12 ESR / Win 7	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256
Firefox 17.0.7 ESR / Win 7	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256
Firefox 21 / Fedora 19	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256
Firefox 24 / Win 7	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256
Googlebot Oct 2013	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
IE 6 / XP No FS [1] No SNI [2]			Fail[3]
IE 7 / Vista	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
IE 8 / XP No FS [1] No SNI [2]			Fail[3]
IE 8-10 / Win 7	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
IE 11 / Win 7	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
IE 11 / Win 8.1	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
Java 6u45 No SNI [2]			Fail[3]
Java 7u25			Fail[3]
OpenSSL 0.9.8y	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) FS	256
OpenSSL 1.0.1e	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) FS	256
Opera 17 / Win 7	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x6b) FS	256
Safari 5.1.9 / OS X 10.6.8	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
Safari 6 / iOS 6.0.1	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) FS	256
Safari 6.0.4 / OS X 10.8.4	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) FS	256
Safari 7 / OS X 10.9	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) FS	256
Tor 17.0.9 / Win 7	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256
Yahoo Slurp Oct 2013	TLS 1.0	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88) FS	256

Tools: masscan

masscan

"TCP port scanner, spews SYN packets asynchronously, scanning entire Internet in under 5 minutes."

- Idea: if you have a very large network range, use masscan (***be sure to rate-limit!***) to discover all open ports 443, 993, 143, 25
- Now you have an inventory of SSL-speaking ports
- Use SSLyze to test these internally

Tools: SSLyze

SSLyze is a "Fast and full-featured SSL scanner"

A tool to test internally which cipher strings are supported.
The tool offers these features (amongst others):

- get a list of targets (ip:port) from a file
- XML output
- heartbleed test
- OCSP stapling test
- SSLv2-TLS1.2 testing
- finding preferred and supported cipher strings
- STARTTLS testing (IMAP, pop, . . .)
- XMPP testing
- SNI support
- HSTS testing

Tools: SSLyze (1)

```
aaron@homeostasis-old-2:~/Desktop/work/nic.at/projects/ACH/ach-master/tools/sslyze % python sslyze.py test.bettercrypto.org --regular --hsts


REGISTERING AVAILABLE PLUGINS
-----------------------------

 PluginOpenSSLCipherSuites
 PluginHSTS
 PluginCertInfo
 PluginSessionRenegotiation
 PluginCompression
 PluginSessionResumption
 PluginChromeSha1Deprecation
 PluginHeartbleed


CHECKING HOST(S) AVAILABILITY
-----------------------------

  test.bettercrypto.org:443          => 78.41.116.67:443


SCAN RESULTS FOR TEST.BETTERCRYPTO.ORG:443 - 78.41.116.67:443
---------------------------------------------------------

 * Deflate Compression:
     OK - Compression disabled

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

 * Certificate - Content:
     SHA1 Fingerprint:                  f706f915c6bd0ce2fcee9a27cc1b4d6f4152fc8b
     Common Name:                       test.bettercrypto.org
     Issuer:                            StartCom Class 1 Primary Intermediate Server CA
     Serial Number:                     057241752D85F3
     Not Before:                        Mar 28 17:36:44 2015 GMT
     Not After:                         Mar 29 08:47:53 2016 GMT
     Signature Algorithm:               sha256WithRSAEncryption
     Key Size:                          4096 bit
```

Tools: SSLyze (2)

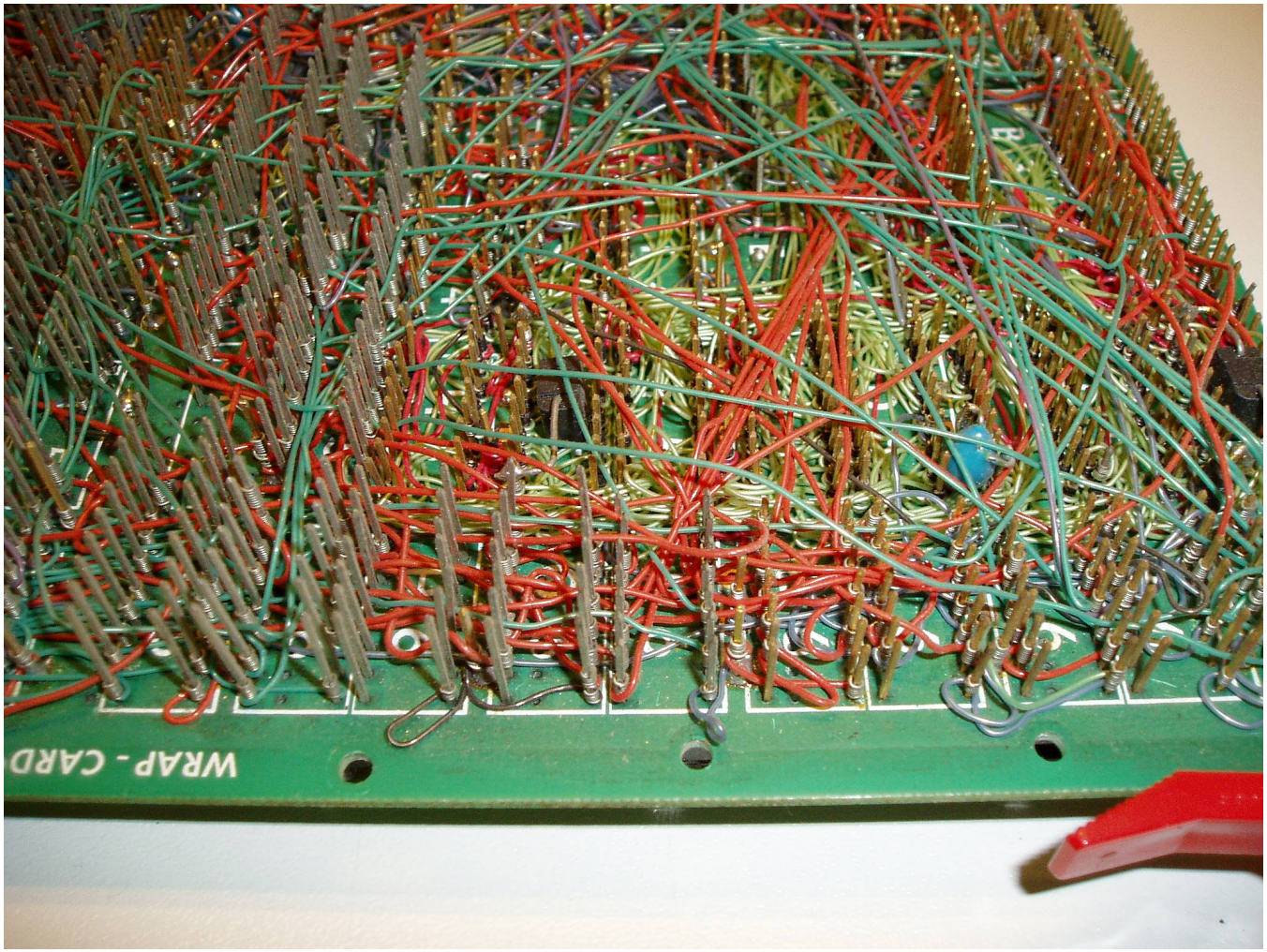
```
    Signature Algorithm:                sha256WithRSAEncryption
    Key Size:                           4096 bit
    Exponent:                           65537 (0x10001)
    X509v3 Subject Alternative Name:    {'DNS': ['test.bettercrypto.org', 'bettercrypto.org']}

* Certificate - Trust:
    Hostname Validation:                OK - Subject Alternative Name matches
    "Mozilla NSS - 08/2014" CA Store:   FAILED - Certificate is NOT Trusted: unable to get local issuer certificate
    "Microsoft - 08/2014" CA Store:     FAILED - Certificate is NOT Trusted: unable to get local issuer certificate
    "Apple - OS X 10.9.4" CA Store:     FAILED - Certificate is NOT Trusted: unable to get local issuer certificate
    "Java 6 - Update 65" CA Store:      FAILED - Certificate is NOT Trusted: unable to get local issuer certificate
    Certificate Chain Received:         ['test.bettercrypto.org']

* Certificate - OCSP Stapling:
    NOT SUPPORTED - Server did not send back an OCSP response.

* HTTP Strict Transport Security:
    NOT SUPPORTED - Server did not send an HSTS header.

* OpenSSL Heartbleed:
    OK - Not vulnerable to Heartbleed

* TLSV1_2 Cipher Suites:
    Preferred:
                ECDHE-RSA-AES256-GCM-SHA384   ECDH-256 bits  256 bits      HTTP 200 OK
    Accepted:
                ECDHE-RSA-AES256-SHA384       ECDH-256 bits  256 bits      HTTP 200 OK
                ECDHE-RSA-AES256-SHA          ECDH-256 bits  256 bits      HTTP 200 OK
                ECDHE-RSA-AES256-GCM-SHA384   ECDH-256 bits  256 bits      HTTP 200 OK
                DHE-RSA-CAMELLIA256-SHA       DH-1024 bits   256 bits      HTTP 200 OK
                DHE-RSA-AES256-SHA256         DH-1024 bits   256 bits      HTTP 200 OK
                DHE-RSA-AES256-SHA            DH-1024 bits   256 bits      HTTP 200 OK
                DHE-RSA-AES256-GCM-SHA384     DH-1024 bits   256 bits      HTTP 200 OK
                CAMELLIA256-SHA               -              256 bits      HTTP 200 OK
                AES256-SHA256                 -              256 bits      HTTP 200 OK
                AES256-SHA                    -              256 bits      HTTP 200 OK
                AES256-GCM-SHA384             -              256 bits      HTTP 200 OK
                ECDHE-RSA-RC4-SHA             ECDH-256 bits  128 bits      HTTP 200 OK
                ECDHE-RSA-AES128-SHA256       ECDH-256 bits  128 bits      HTTP 200 OK
                ECDHE-RSA-AES128-SHA          ECDH-256 bits  128 bits      HTTP 200 OK
                ECDHE-RSA-AES128-GCM-SHA256   ECDH-256 bits  128 bits      HTTP 200 OK
                DHE-RSA-SEED-SHA              DH-1024 bits   128 bits      HTTP 200 OK
                DHE-RSA-CAMELLIA128-SHA       DH-1024 bits   128 bits      HTTP 200 OK
                DHE-RSA-AES128-SHA256         DH-1024 bits   128 bits      HTTP 200 OK
                DHE-RSA-AES128-SHA            DH-1024 bits   128 bits      HTTP 200 OK
                DHE-RSA-AES128-GCM-SHA256     DH-1024 bits   128 bits      HTTP 200 OK
                SEED-SHA                      -              128 bits      HTTP 200 OK
                RC4-SHA                       -              128 bits      HTTP 200 OK
                CAMELLIA128-SHA               -              128 bits      HTTP 200 OK
                AES128-SHA256                 -              128 bits      HTTP 200 OK
                AES128-SHA                    -              128 bits      HTTP 200 OK
                AES128-GCM-SHA256             -              128 bits      HTTP 200 OK
                ECDHE-RSA-DES-CBC3-SHA        ECDH-256 bits  112 bits      HTTP 200 OK
```

Brainstorming: finding an internal testing strategy

Some questions to think about:

- Which tests are non-intrusive? Which should be avoided?
- Do we need a full inventory? Can we generate the inventory? Does the inventory match the scanned results (hosts & ports)?
- Should any mismatch raise alarms?
- Once we identified all IPs:ports - do we need to know hostnames? (SNI)?
- Defining a common base-line level which MUST be supported
- automatic testing against that base-line level? How?
- Integration into existing monitoring solutions?

Wrap-up



Current state as of 2015-03-29

- OK: Solid basis with Variant (A) and (B)
- Public draft was presented at the CCC Dec 2013. Well received. Good feedback (Dan Bernstein, . . .)
- more work needs to be done:
- certificate pinning
- DANE
- SPDY, HTTP/2
- etc.
- This is a process which should be done regularly

Links

- Website: `https://www.bettercrypto.org`
- Master (read-only) Git repo: `https://git.bettercrypto.org`
- Public github repo for PRs: `https://github.com/BetterCrypto/Applied-Crypto-Hardening`
- Mailing list: `http://lists.cert.at/cgi-bin/mailman/listinfo/ach`
- IRC: #bettercrypto on freenode
- Twitter: @bettercrypto

Thanks

Thanks