Intricacies testing SSL:

sockets, schools, threa{t,d}s and sometimes no shake-hands

Dirk Wetter

@drwetter

0. $CWD

- **Independent security consultant**
  - pentests / defense+hardening / telling what's best / PM
  - historical strong unix-/networking background
- **Community 2: OWASP**
  - Chair + PC lead OWASP AppSec Research 2013
  - Charing German OWASP Day 2012, 2014
  - German Chapter Lead
- **Community 1: GUUG**
  - Program committee's
  - Former board

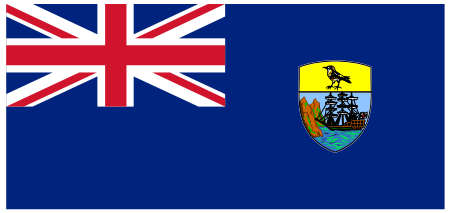
https://drwetter.eu/

1. Intro

- **testssl.sh: what is that?**
  - Blunt:
    - Check's any server's SSL/TLS encryption
  - Cool thing:
    - `/bin/bash`
  - compatible:
    - Linux
    - Mac OS X
    - (Free)BSD
    - Windows: MSYS2, Cygwin

1. Intro

- **testssl.sh**
  - 2005: inhouse testing tool (pentests)
  - Open sourced: ~ 2010
    - 2/2014: same name: domain
    - 4/2014: bitbucket
    - 10/2014: github
    - 3 releases in 2015
  - ~ 5400 LoC



Flag: https://upload.wikimedia.org/wikipedia/commons/0/00/Flag_of_Saint_Helena.svg from „Patricia Fidi“. Licence public domain

1. Intro

- **testssl.sh**
  - Distributions:
    - Pentesting:
      - BackTrack, BlackArch Linux, Weakerthan Linux
      - Hello Kali, anybody out there? ;-)
    - Regular:
      - ArchLinux
      - Debian testing
      - Ubuntu Xenial Xerus! (=16.04 LTS)

1. Intro

- **idea annum 2005**

  - EZ
    - SSL/TLS protocol versions
    - Cipher suites
    - CN / expiration date
      - certificate
  - Trust:
    - s.o. / -verify
    - Browser

→ Demo (1)

2. Code

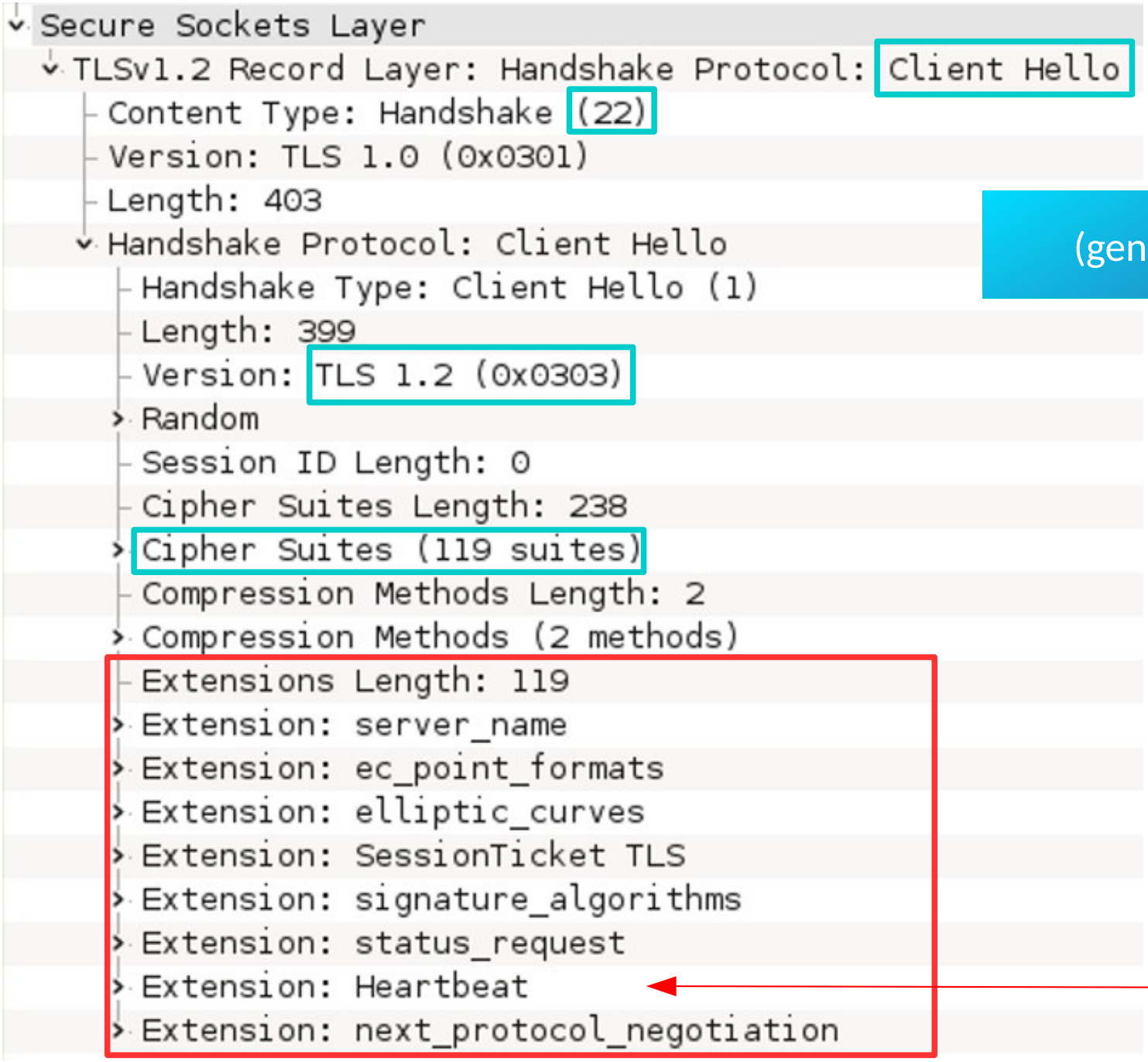
- **A decade later: 2015**
  - vulnerabilities

```
-U, --vulnerable                tests all vulnerabilities
-B, --heartbleed                tests for heartbleed vulnerability
-I, --ccs, --ccs-injection      tests for CCS injection vulnerability
-R, --renegotiation             tests for renegotiation vulnerabilities
-C, --compression, --crime      tests for CRIME vulnerability
-T, --breach                    tests for BREACH vulnerability
-O, --poodle                    tests for POODLE (SSL) vulnerability
-Z, --tls-fallback              checks TLS_FALLBACK_SCSV mitigation
-F, --freak                     tests for FREAK vulnerability
-A, --beast                     tests for BEAST vulnerability
-J, --logjam                    tests for LOGJAM vulnerability
-s, --pfs, --fs,--nsa           checks (perfect) forward secrecy settings
-4, --rc4, --appelbaum          which RC4 ciphers are being offered?
```

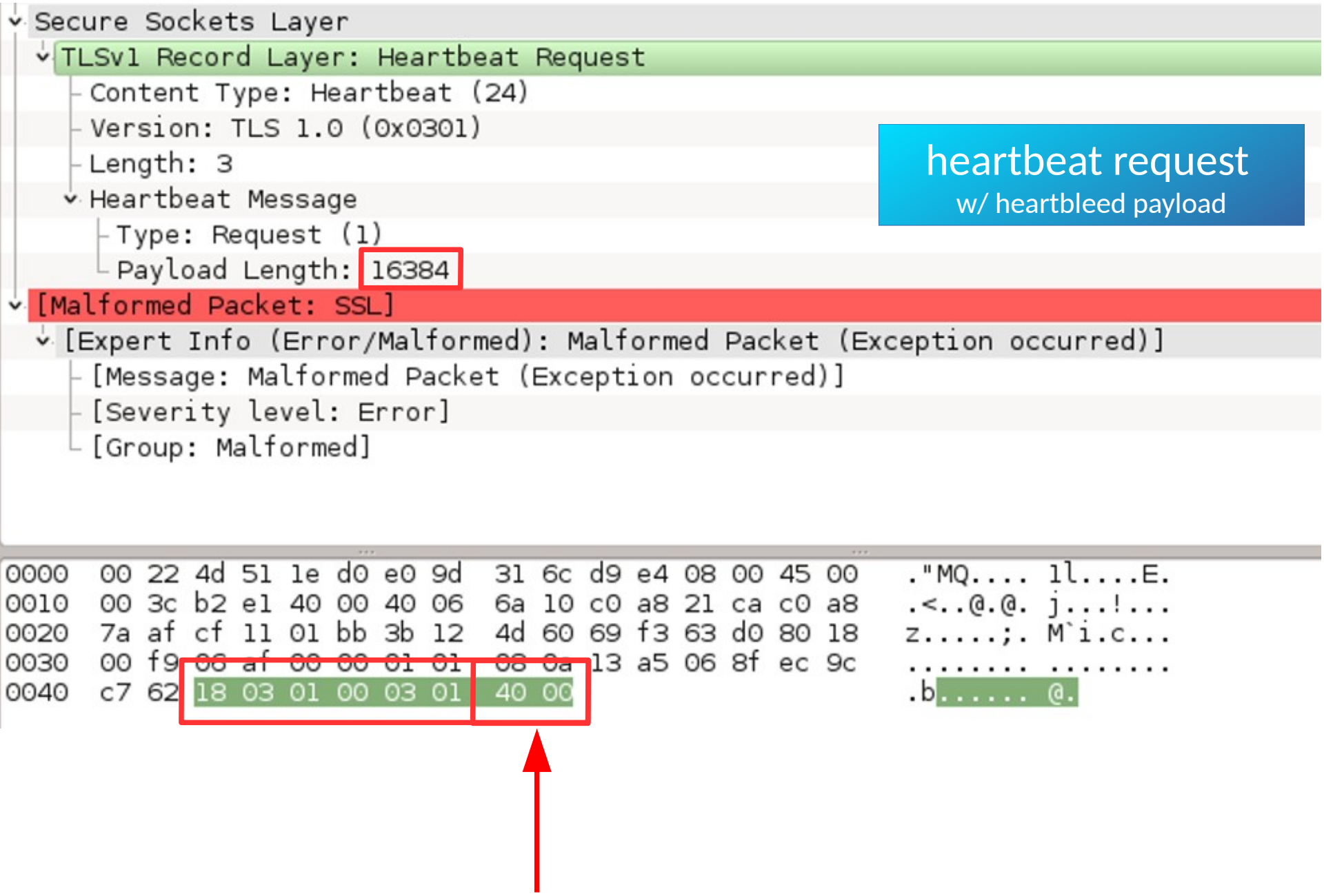
2. Code

- **But how the heck works**
  - Heat**beat**: TLS extension
    - **useless** for 99,99% usage

2. Code



(general) request


Secure Sockets Layer
TLSv1 Record Layer: Heartbeat Request
Content Type: Heartbeat (24)
Version: TLS 1.0 (0x0301)
Length: 3
Heartbeat Message
Type: Request (1)
Payload Length: 16384
[Malformed Packet: SSL]
[Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
[Message: Malformed Packet (Exception occurred)]
[Severity level: Error]
[Group: Malformed]
0000 00 22 4d 51 1e d0 e0 9d 31 6c d9 e4 08 00 45 00
0010 00 3c b2 e1 40 00 40 06 6a 10 c0 a8 21 ca c0 a8
0020 7a af cf 11 01 bb 3b 12 4d 60 69 f3 63 d0 80 18
0030 00 f9 08 af 00 00 01 01 08 0a 13 a5 06 8f ec 9c
0040 c7 62 18 03 01 00 03 01 40 00
heartbeat request
w/ heartbleed payload

```
Transmission Control Protocol, Src Port: https (443), Dst Port: 53009 (53
    Source port: https (443)
    Destination port: 53009 (53009)
    [Stream index: 0]
    Sequence number: 1292    (relative sequence number)
    [Next sequence number: 2740    (relative sequence number)]
    Acknowledgment number: 234    (relative ack number)
    Header length: 32 bytes
  > Flags: 0x010 (ACK)
    Window size value: 1877
    [Calculated window size: 30032]
    [Window size scaling factor: 16]
```

```
0040  06 8f 18 03 01 40 00 02  40 00 d8 03 01 53 43 5b   .....@.. @....SC[
0050  90 9d 9b 72 0b bc 0c bc  2b 92 a8 48 97 cf bd 39   ...r.... +..H...9
0060  04 cc 16 0a 85 03 90 9f  77 04 33 d4 de 00 00 66   ........ w.3....f
0070  c0 14 c0 0a c0 22 c0 21  00 39 00 38 00 88 00 87   .....".! .9.8....
0080  c0 0f c0 05 00 35 00 84  c0 12 c0 08 c0 1c c0 1b   .....5.. ........
0090  00 16 00 13 c0 0d c0 03  00 0a c0 13 c0 09 c0 1f   ........ ........
00a0  c0 1e 00 33 00 32 00 9a  00 99 00 45 00 44 c0 0e   ...3.2.. ...E.D..
00b0  c0 04 00 2f 00 96 00 41  c0 11 c0 07 c0 0c c0 02   .../...A ........
00c0  00 05 00 04 00 15 00 12  00 09 00 14 00 11 00 08   ........ ........
00d0  00 06 00 03 00 ff 01 00  00 49 00 0b 00 04 03 00   ........ .I......
00e0  01 02 00 0a 00 34 00 32  00 0e 00 0d 00 19 00 0b   .....4.2 ........
00f0  00 0c 00 18 00 09 00 0a  00 16 00 17 00 08 00 06   ........ ........
0100  00 07 00 14 00 15 00 04  00 05 00 12 00 13 00 01   ........ ........
0110  00 02 00 03 00 0f 00 10  00 11 00 23 00 00 00 0f   ........ ...#....
0120  00 01 01 4a d6 ce 56 0e  d8 86 87 2e 61 6d b1 7e   ...J..V. ....am.~
0130  7a 4d 5a 12 ee eb 13 27  cd 8a 61 10 69 b0 4d cf   zMZ....' ..a.i.M.
0140  2c 2a 39 78 99 04 b3 54  0f 9d 6a c1 36 03 00 0a   ,*9x...T ..j.6...
0150  00 93 00 15 00 12 00 0f  00 0c 00 09 00 ff 02 01   ........ ........
0160  00 00 64 00 00 00 0b 00  09 00 00 06 62 6f 72 6b   ..d..... ....bork
0170  65 6e 00 0b 00 04 03 00  01 02 00 0a 00 1c 00 1a   en...... ........
0180  00 17 00 19 00 1c 00 1b  00 18 00 1a 00 16 00 0e   ........ ........
0190  00 0d 00 0b 00 0c 00 09  00 0a 00 23 00 00 00 0d   ........ ...#....
01a0  00 20 00 1e 06 01 06 02  06 03 05 01 05 02 05 03   . ...... ........
01b0  04 01 04 02 04 03 03 01  03 02 03 03 02 01 02 02   ........ ........
```

heart**bleed** response

2. Code

- But how the heck works
  - Heat**beat**: TLS extension
    - **useless** for 99,99% usage

  - Buffer Overflow, mem access
    - Doesn't work w/ OpenSSL!
    - PoC in bash sockets

→ Demo (5)

2. Code

- **Sockets**
  - Heartbleed
    - TLS Extension
  - CCS Injection (CVE-2014-0224)
    - Not extension-based
    - Similar check = socket based
      - First PoC in bash sockets

2. Code

- **Sockets**
  - TLS Handshakes (+SSLv2)
    - ClientHello
    - Parser for ServerHello
  - atm: for protocol checks only
  - Cool:
    - Proxy
    - STARTTLS
  - nice by-product:
    - TLS TIME – depends on server
    - always works: HTTP Time Stamp

    → fingerprinting!

→ Code (5)

```
Testing now (2015-05-16 22:44) --->[redacted]:443 ([redacted]) <---

 rDNS ([redacted]):  --
 Service detected:       HTTP


--> Testing server defaults (Server Hello)

 TLS clock skew:              +159 sec from localtime
 HTTP clock skew:             +20 sec from localtime
 TLS server extensions        server name, renegotiation info, session ticket
 Session Tickets RFC 5077     (none)
 Server key size              2048 bit
 Signature Algorithm          SHA256withRSA
 Fingerprint / Serial         SHA1 [redacted]DB1D92056B628EE26345E4AB4[redacted] / [redacted]9988
                              SHA256 [redacted]49CE2D445F9C8C4892D8018B948E0F9F74859F[redacted]
 Common Name (CN)             [redacted] (works w/o SNI)
 subjectAltName (SAN)         [redacted]
 Issuer                       thawte SSL CA - G2 (thawte, Inc. from US)
 Certificate Expiration       >= 60 days  (2015-01-26 01:00 --> 2018-04-27 01:59 +0200)
 # of certificates provided   2
 Certificate Revocation List  http://tj.symcb.com/tj.crl
 OCSP URI                     http://tj.symcd.com
 OCSP stapling                not offered


Done now (2015-05-16 22:44) --->[redacted]:443 ([redacted]) <---
```

2. Code

- **Problem using OpenSSL**
  - new Linux/BSD distributions, Mac OS X: „Fixes“
    - Null, Anonymous Ciphers
    - SSLv2
      - wrt SSL Poodle: SSLv3 coming
    - export ciphers (FREAK)
    - weak DH ciphers (Logjam)

2. Code

- **Until full socket implementation**
  - Distribution of hand crafted binaries
    - based on Peter Mosmans fork
    - Linux, BSD, Darwin, ARM7
  - Broken features + ciphers
  - Advanced features + ciphers
    - 3x Chacha20/Poly1305 cipher
    - `-proxy`, `-xmpphost <host>`, ...
    - @ horizon: OpenSSL 1.1.0: CCM Cipher
  - ugly: github

2. Code

- **Summary: Sockets vs. OpenSSL**
  - Using both!
    - Sockets where possible/needed
      - protocol checks SSLv2 - TLS 1.1
      - (TLS 1.2)
    - TLS time
    - HB+CCS
  - More to come

3. Shell rulz

- **but sometimes not as easy... ;-)**
  - Platform compatible cmd line parser
    - `/bin/bash: getopts`
      - Always works (BSDs / Linux)
      - Catch: ~~`--long option`~~
    - `/usr/bin/getopt`
      - Linux (util-linux): GNU (long und short options)
      - BSD/Mac OSX: just different
  - Logic conclusion:
    - own parser
    - Peter Mosmans

3. Shell rulz

- **but sometimes not as easy... ;-)**
  - Mac OS X:
    - Bash3 !!1! 7/2004

3. Shell rulz

- **Static Code Analysis**
  - Shellcheck (github.com/koalaman/shellcheck)
  - **Demo:** shellcheck.net
  - security:
    - Randomly
  - **Demo:** testssl.sh

→ Demo (7)

3. Shell rulz

- ~~sed & awk~~ /bin/bash!
  - `cat "$x" | sed 's/pattern/replace/g'`

  → `"${x//pattern/replace}"`

  - `echo -n "$line" | wc -c`

  → `echo "${#line}"`

```
var="0x00,0xA5 - DH-DSS-AES256-GCM-SHA384 TLSv1.2
Kx=DH/DSS   Au=DH   Enc=AESGCM(256) Mac=AEAD”
```

  - `echo "$var" | awk '{ print $NF }'`

  → `echo "${var##0x* }"`

nixCraft
@nixcraft

Following

HowTo: Use Bash Parameter Substitution Like A Pro cyberciti.biz/tips/bash-shel … #unix #linux #sysadmin #devops #bsd

${parameter:-defaultValue}	Get default shell variables value
${parameter:=defaultValue}	Set default shell variables value
${parameter:?"Error Message"}	Display an error message if parameter is not set
${#var}	Find the length of the string
${var%pattern}	Remove from shortest rear (end) pattern
${var%%pattern}	Remove from longest rear (end) pattern
${var:num1:num2}	Substring
${var#pattern}	Remove from shortest front pattern
${var##pattern}	Remove from longest front pattern
${var/pattern/string}	Find and replace (only replace first occurrence)
${var//pattern/string}	Find and replace all occurrences

3. Shell rulz

- **limits**
  - ~~threads / events~~
    - important for broken servers/ load balancers etc.
    - Background, asynchronous & polling:

```
printf "$GET_REQ11" |
      $OPENSSL s_client -quiet -connect $NODEIP:$PORT \
           $PROXY $SNI 1>$HEADERFILE 2>$ERRFILE &

wait_kill $! $MAXSLEEP  # wait_kill() waits for PID (= $!) in
                        # the background for $HEADER_MAXSLEEP
                        # seconds.  !=0: killed
```

→ results in `$HEADERFILE, $ERRFILE`

4. Threats

```
###########################################################
     testssl.sh       2.7dev from https://testssl.sh/dev/
     (1.393 2015/09/26 20:44:32)

      This program is free software. Distribution and
             modification under GPLv2 permitted.
      USAGE w/o ANY WARRANTY. USE IT AT YOUR OWN RISK!

       Please file bugs @ https://testssl.sh/bugs/

###########################################################

 Using "OpenSSL 1.0.2-chacha (1.0.2e-dev)" [~199 ciphers] on
 [redacted]:/[redacted]/openssl64
 (built: "Jul 20 18:52:44 2015", platform: "linux-elf")
```

4. Threats

- **Wait… what, risk???**
  - Threat Modeling…
    - Web

4. Threats

The only thing left is to make it nice and simple so the service desk can run it. That's where aha comes in. Aha (or the ANSI HTML Adapter) takes terminal output with ANSI colour and formatting codes, and turns it into nice neat HTML for your browser. It's directly compatible with testssl.th,

```Shell
./testssl.sh --warnings batch wsus:8531 | aha > wsus-SSL.html
```

but that doesn't quite handle everything, I've still got to start the scan manually. PHP to the rescue! we can use shell_exec() to run the script. But hold on, that's rather dangerous, I hear you say. Well, you're right.

That could allow any command to be run on my server. So, let's fix that. Luckily, PHP can take care of that for us as well, that' what escapeshellcmd() is for, it gets rid of all the nasty stuff. That leaves us with a rather nice one-liner:

```PHP
echo shell_exec(
    escapeshellcmd(
        "/root/testssl.sh/testssl.sh --warnings batch ".$_GET['server']." | aha"
    )
);
```

4. Threats

- Wait … what, risk???
  - Threat Modeling…
    - Web
    - XSS through DNS

4. Threats

```
$ dig jamiehankins.co.uk txt
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18242
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;jamiehankins.co.uk.    IN TXT

;; ANSWER SECTION:
jamiehankins.co.uk.     300 IN TXT "google-site-verification=nZUP4BagJAjQZO6AImXyzJZBXBf9s1FbDZr8pz
NLTCI"
jamiehankins.co.uk.     300 IN TXT "v=spf1 include:spf.mandrillapp.com ?all"
jamiehankins.co.uk.     300 IN TXT "<script src='//peniscorp.com/topkek.js'></script>"
jamiehankins.co.uk.     300 IN TXT "<iframe width='420' height='315' src='//www.youtube.com/embed/d
Qw4w9WgXcQ?autoplay=0' frameborder='0' allowfullscreen></iframe>"
```

```
300     IN      TXT     "X5O!P%@AP[4PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*"
```

4. Threats

- Wait ... what, risk???
  - Threat Modeling…
    - Web
    - XSS through DNS
    - Tavis Ormandy (AV)

4. Threats

Project Member Reported by tav...@google.com, Sep 25, 2015

```
Avast will render the commonName of X.509 certificates into an HTMLLayout frame when your
MITM proxy detects a bad signature. Unbelievably, this means CN="<h1>really?!?!?</h1>"
actually works, and is pretty simple to convert into remote code execution.

To verify this bug, I've attached a demo certificate for you. Please find attached key.pem,
cert.pem and cert.der. Run this command to serve it from a machine with openssl:

$ sudo openssl s_server -key key.pem -cert cert.pem -accept 443

Then visit that https server from a machine with Avast installed. Click the message that
appears to demonstrate launching calc.exe.

Thanks, Tavis.
```

This bug is subject to a 90 day disclosure deadline. If 90 days elapse without a broadly available patch, then the bug report will automatically become visible to the public.

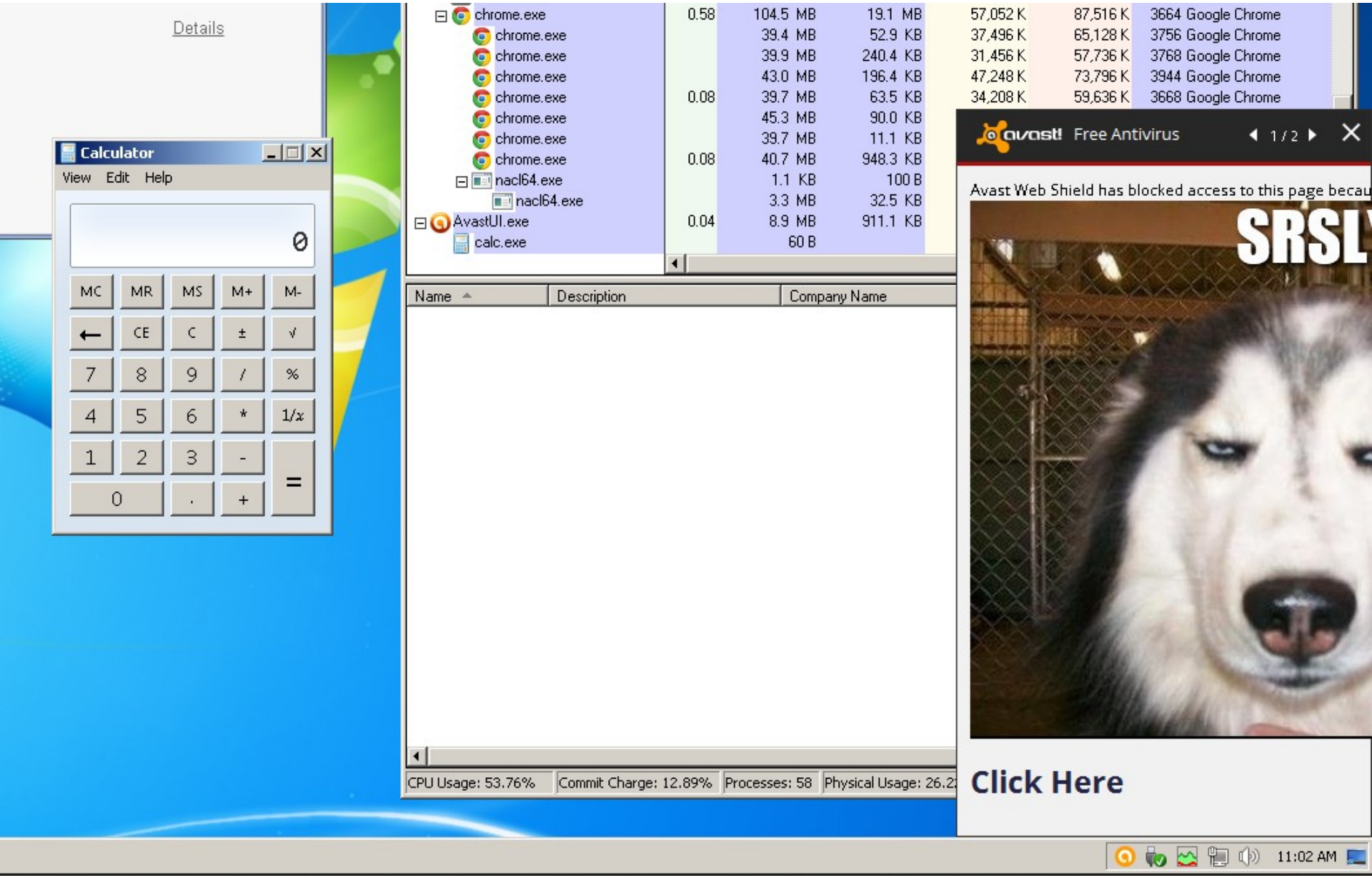
Project Member #1 tav...@google.com Sep 25, 2015

Attaching testcases.

cert.der
884 bytes Download

4. Threats

```
dirks@laptop:/tmp|0% openssl x509 -in cert.der -inform der -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 17797272642042972612 (0xf6fc9d2c87bfc9c4)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=CA, L=Mountain View, O=Google, OU=Project Zero, CN=testing
        Validity
            Not Before: Sep 25 14:29:11 2015 GMT
            Not After : Oct 25 14:29:11 2015 GMT
        Subject: C=US, ST=CA, L=Mountain View, O=Google, OU=Project Zero, CN=<a href="file://
/c:/Windows/System32/calc.exe">\x0D\x0A<img src="http://www.troll.me/images/are-you-fucking-k
idding-me/srsly.jpg" />\x0D\x0A</a>\x0D\x0A<h1><a href="file:///c:/Windows/System32/calc.exe"
>Click Here</a></h1>
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
```


Details
Calculator
View Edit Help
0
MC MR MS M+ M-
← CE C ± √
7 8 9 / %
4 5 6 * 1/x
1 2 3 - =
0 . +
chrome.exe 0.58 104.5 MB 19.1 MB 57,052 K 87,516 K 3664 Google Chrome
chrome.exe 39.4 MB 52.9 KB 37,496 K 65,128 K 3756 Google Chrome
chrome.exe 39.9 MB 240.4 KB 31,456 K 57,736 K 3768 Google Chrome
chrome.exe 43.0 MB 196.4 KB 47,248 K 73,796 K 3944 Google Chrome
chrome.exe 0.08 39.7 MB 63.5 KB 34,208 K 59,636 K 3668 Google Chrome
chrome.exe 45.3 MB 90.0 KB
chrome.exe 39.7 MB 11.1 KB
chrome.exe 0.08 40.7 MB 948.3 KB
nacl64.exe 1.1 KB 100 B
nacl64.exe 3.3 MB 32.5 KB
AvastUI.exe 0.04 8.9 MB 911.1 KB
calc.exe 60 B
Name Description Company Name
CPU Usage: 53.76% Commit Charge: 12.89% Processes: 58 Physical Usage: 26.2
avast! Free Antivirus 1 / 2
Avast Web Shield has blocked access to this page becau
SRSL
Click Here
11:02 AM

4. Threats

- **Wait… what, risk???**
  - Threat Modeling…
    - Web
    - XSS through DNS
    - Tavis Ormandy (A/V)

    - Command Injection

→ Idea (8)

5. Bugs & not shake-hands

- So far, not so good

5. Bugs & not shake-hands

- **strange Debian/Ubuntu bug**
  - also: Debian Wheezy, Ubuntu 15.04

```
dirks@laptop:~|0% export OPENSSL_CONF=gost.conf
dirks@laptop:~|0% nslookup -query=a testssl.sh
GOST engine already loaded
08-Sep-2015 20:12:43.648 ENGINE_by_id failed (crypto failure)
08-Sep-2015 20:12:43.649 error:2606A074:engine routines:ENGINE_by_id:no such engine:eng_list.c:4
st
(null): dst_lib_init: crypto failure
dirks@laptop:~|10% host  testssl.sh
GOST engine already loaded
08-Sep-2015 20:12:56.324 ENGINE_by_id failed (crypto failure)
08-Sep-2015 20:12:56.325 error:2606A074:engine routines:ENGINE_by_id:no such engine:eng_list.c:4
st
host: dst_lib_init: crypto failure
dirks@laptop:~|1% dig +short testssl.sh
GOST engine already loaded
08-Sep-2015 20:13:06.326 ENGINE_by_id failed (crypto failure)
08-Sep-2015 20:13:06.327 error:2606A074:engine routines:ENGINE_by_id:no such engine:eng_list.c:4
st
dig: dst_lib_init: crypto failure
dirks@laptop:~|10% grep PRETTY_NAME /etc/os-release
PRETTY_NAME="Ubuntu 14.04.3 LTS"
dirks@laptop:~|0%
```

5. Bugs & not shake-hands

- **OpenLiteSpeed**
  - SSLv2: disabled by default
  - Plaintext answer!
    - There's no SSLv2 RFC
      - What to do with a drunken handshake ;-)

5. Bugs & not shake-hands

- **Class of servers with handshake-size limits**
  - OpenSSL 1.0.2+: handshake failure
    - handshake-size limit
    - More ciphers nowadays

5. Bugs & not shake-hands

- **IIS 6.0**
  - No support
    - Some folks don't care
      - Shodan

HTTP	987,345
HTTPS	314,890
HTTP (8080)	142,378
444	70,574
HTTP (81)	58,467

- **Cisco ACE Loadbalancer**
  - ClientHello w/ >128 ciphers

5. Bugs & not shake-hands

- **F5 BigIP Load Balancer + old firmware**
  - SSL request stalls
  - Weird limits for handshake

Project Member Reported by a...@chromium.org, Nov 6, 2013

```
F5 tell us that we need only ensure that a ClientHello record is not between 256 and
511 bytes long (inclusive). We can do that by adding an extension to pad the
ClientHello when it would otherwise fall into that range.
```

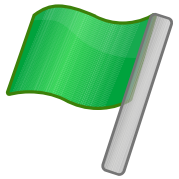
```
add `padding' extension when we might hit the F5 bug.
```

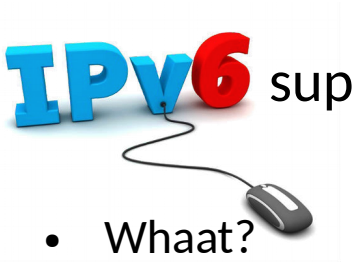
5. conclusion

- **Project testssl.sh is „alive and kicking“**
  - 3 releases this year
  - contributions++
    - Peter Mosmans OpenSSL fork
    - Proxy (John Newbigin)
    - TLS_FALLBACK_SCSV (Jonathan Roach)
    - …
      - you know who, see CREDITS.md ;-)

5. conclusion

- **what's new in 2.7dev?**
  - Check of trust chain
    - Mozilla / Microsoft / JDK 1.8 / Linux ca-bundle.crt



  -  support
    - Whaat? It's 2015!! C'mon...
    - OpenSSL constraints, don't get me started...

5. conclusion

- **What else in 2.7dev?**
  - Support for servers with X509 client authentication
    - SSL Session ID check
  - Smart logging
  - Some workarounds for buggy systems
  - Upcoming:
    - HTTP/2 a.k.a. ALPN (PR)
    - JSON output (Frank Breedijk)
    - Color blind (Thomas Martens)
    - POODLE TLS
  - External
    - Docker images
    - brew

6. outlook

- **future**
  - Features targeted for 2.8: github.com/drwetter/testssl.sh/milestones/2.7dev%20(2.8)%20
    - complete socket support
      - TLS 1.2: extensions + IIS
      - Disabled ciphers
    - Even more workarounds or buggy systems
    - CN  Hostname validation
    - EC curves: naming, check
    - Parallel scanning
    - …
  - Rating?
    - Management compatible
    - If fair: not as easy

6. outlook

- **challenges**
  - vulnerabilities: to be on the ball
    - expectation: getting less
  - broken handshakes
    - work around it
  - test plattforms
    - client: Plattform compatibility
    - server
    - Ideas for CI?

- **Thanks!**

  - https://testssl.sh/
  - dirk aet testssl.sh

    mail aet drwetter eu

@drwetter