`testssl.sh`:
Über die Herausforderungen, SSL-Testing in /bin/bash zu implementieren

Dirk Wetter

@drwetter

1. :~> whoami

https://drwetter.eu/

- **Unabhängiger Sicherheitsberater (selbständig)**
  - Pentests / Absicherung / Beratung + Projektmanagement
  - Historisch starker Unix-/Networking-Hintergrund
    - Hochsprachenentwicklung: long time ago
- **Ehrenamt GUUG**
  - PK-Leiter FFGs, Linux-Kongress
  - Ex-Vorstand
- **Ehrenamt OWASP**
  - OWASP AppSec Research 2013
  - German OWASP Day 2012, 2014
  - German Chapter Lead

2. Einleitung

- **Was ist testssl.sh**
  - Besonderheit: Kommandozeile!
    - `/bin/bash`
  - Kompatibel:
    - Linux
    - Mac OS X
    - (Free)BSD
    - Windows: MSYS2, Cygwin

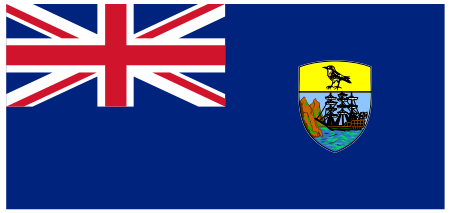
2. Einleitung

- **Anno 2005**
  - OpenSSL als Schweizer Messer
  - Mit paar Befehlen überprüft
    - CN / expiration date
      - Zertifikate
    - Protokollversionen
    - Cipher
  - Trust:
    - s.o. / -verify
    - Browser

→ Demo (1)

2. Einleitung

- **testssl.sh**
  - Gestartet 2005 als Inhouse-Tool (Pentests)
  - Open sourced: <= 2010
    - 2/2014: gleichnamige Domain
    - 4/2014: bitbucket
    - 10/2014: github
  - Distros:
    - ArchLinux, BackTrack, BlackArch Linux, Weakerthan Linux
    - Debian : testing



2. Einleitung

- **Anno 2005**
  - Mehr?
  - Brauchte es (fast) nicht
    - Ok ok …
    - es gab da so ein paar Bugs ;-)
      - Debian weak keys (2006, CVE-2008-0166)
      - Sonst: Version/Banner Fingerprinting
      - Sonst: NSE Plugin ggf.

3. Heute

- **Anno 2015**
  - Tierisch gewachsen
    - Gut 5000 Zeilen Code
    - Relativ „reif“
    - Viele Features
  - Drei Releases

→ Demo (2)

3. Heute

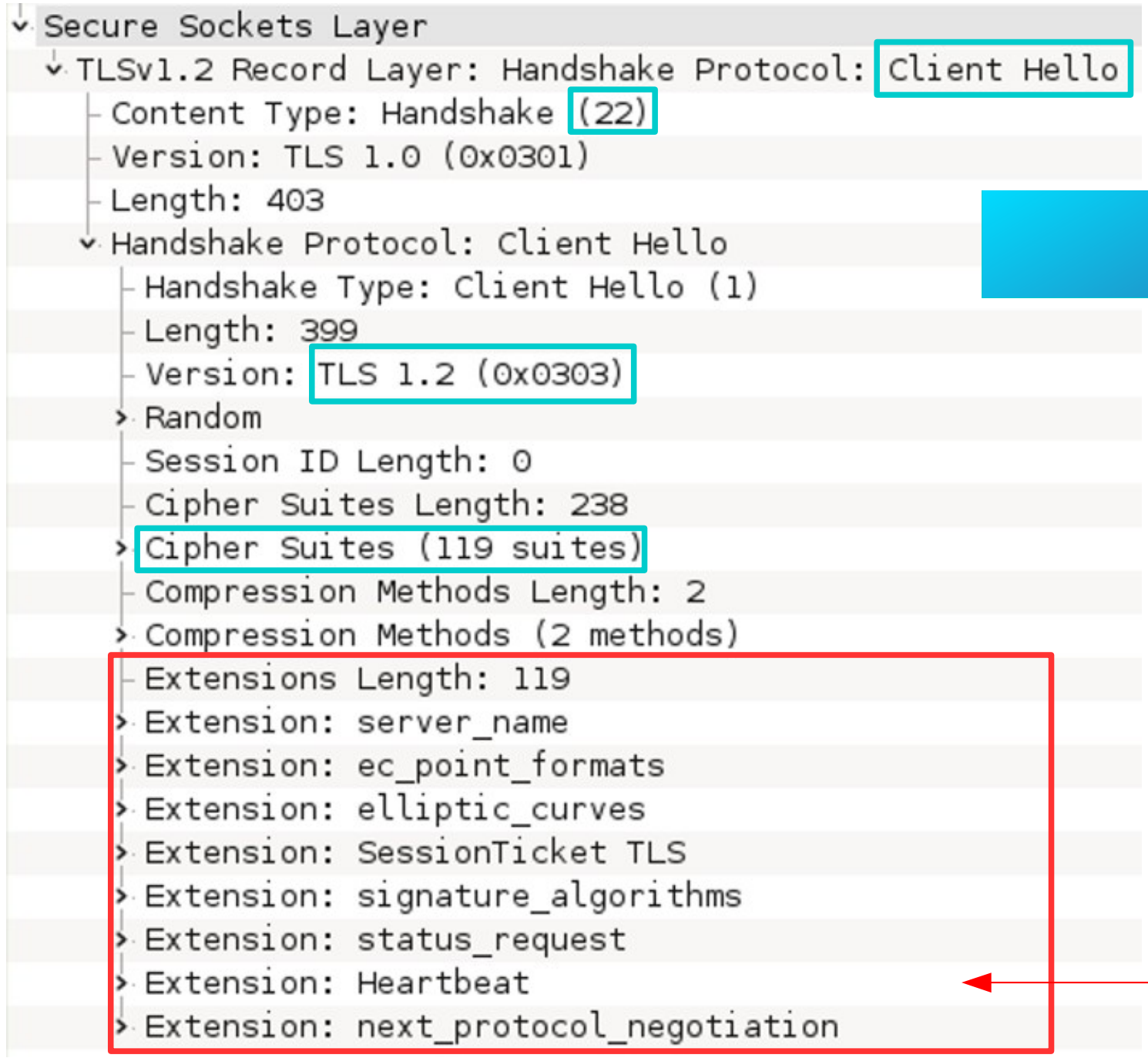
- **Anno 2015**
  - Verwundbarkeiten

→ Demo (3)

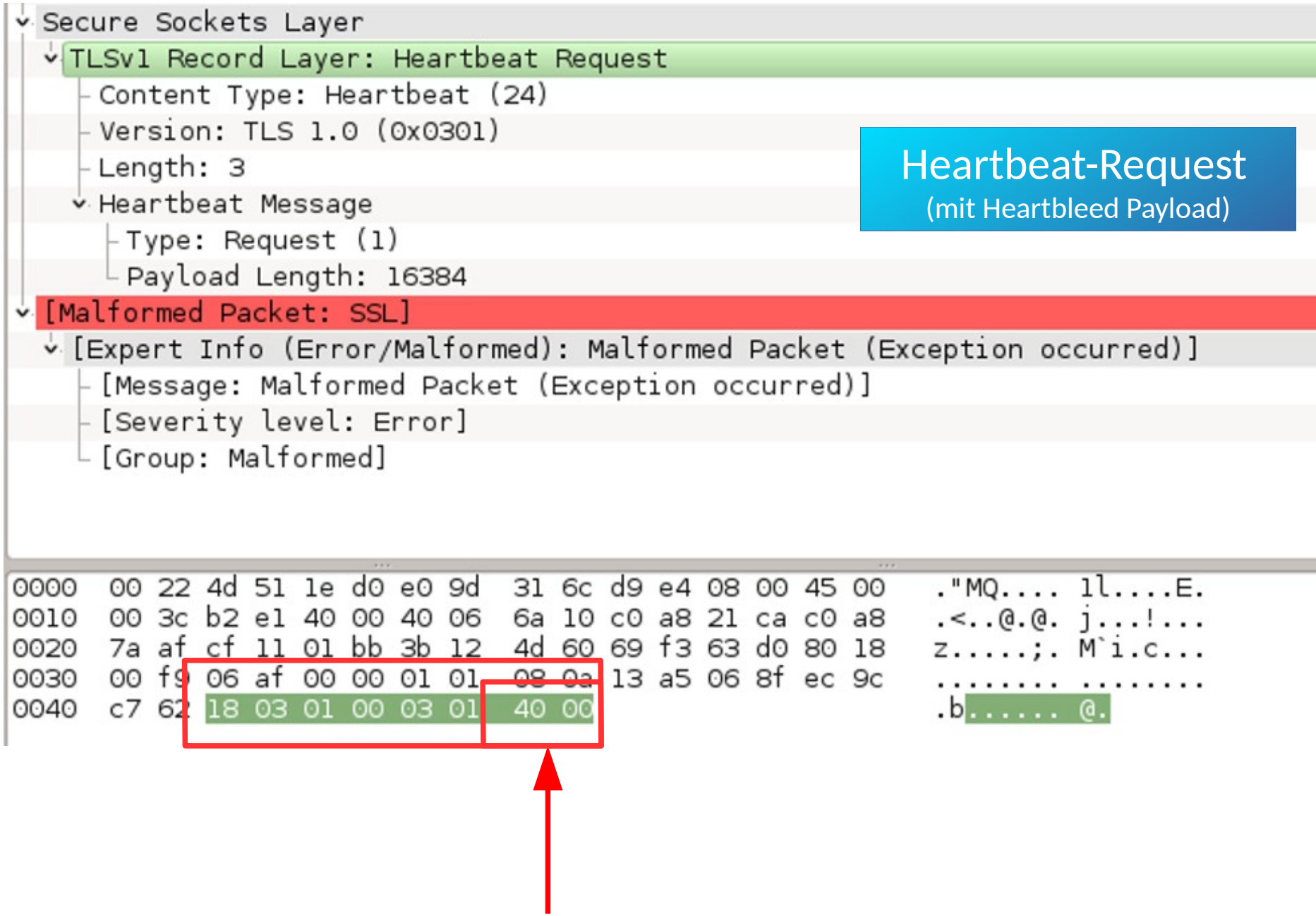
3. Code

- **Aber wie macht der das mit**
  - Heartbleed
    - TLS Extension
    - Heartbeat: **sinnlose** Extension
      - (für die meisten)

→ Demo (4)


Secure Sockets Layer
TLSv1.2 Record Layer: Handshake Protocol: Client Hello
Content Type: Handshake (22)
Version: TLS 1.0 (0x0301)
Length: 403
Handshake Protocol: Client Hello
Handshake Type: Client Hello (1)
Length: 399
Version: TLS 1.2 (0x0303)
Random
Session ID Length: 0
Cipher Suites Length: 238
Cipher Suites (119 suites)
Compression Methods Length: 2
Compression Methods (2 methods)
Extensions Length: 119
Extension: server_name
Extension: ec_point_formats
Extension: elliptic_curves
Extension: SessionTicket TLS
Extension: signature_algorithms
Extension: status_request
Extension: Heartbeat
Extension: next_protocol_negotiation

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 ."MQ.... 1l....E.
0010 00 3c b2 e1 40 00 40 06 6a 10 c0 a8 21 ca c0 a8 .<..@.@. j...!...
0020 7a af cf 11 01 bb 3b 12 4d 60 69 f3 63 d0 80 18 z.....;. M`i.c...
0030 00 f9 06 af 00 00 01 01 08 0a 13 a5 06 8f ec 9c ....
0040 c7 62 18 03 01 00 03 01 40 00 .b...... @.
Heartbeat-Request
(mit Heartbleed Payload)

Heartbleed Response

```
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   ........ ........
```

3. Code

- **Aber wie macht der das mit**
  - Heartbleed
    - TLS Extenstion
    - Heartbeat: **sinnlose** Extension
      - für die meisten
  - Buffer Overflow, mem access
    - Trivialer Zugriff
    - Geht nicht mit OpenSSL!
    - PoC in bash sockets

→ Demo (5)

3. Code

- **Sockets** vs. OpenSSL
  - Heartbleed
    - TLS Extension
  - CCS Injection
    - braucht auch Sockets
  - SSLv2 Handshake
  - TLS Handshake

3. Code

- **Sockets** vs. OpenSSL
  - TLS Handshakes
    - Client hello
      - Generell
      - Proxy
      - STARTTLS
    - Senden
    - Parser für Server Hello

→ Demo=Code (6)

3. Code

- **Sockets** vs. OpenSSL
  - TLS Handshakes
    - Client hello
      - Generell
      - Proxy
      - STARTTLS
    - Senden
    - Parser für Server Hello
    - Schönes Abfallprodukt:
      - TLS TIME
        - (je nach OpenSSL Version)
    - Geht immer: HTTP Time Stamp

} Version & Architektur

```
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]) <---
```

- **Sockets vs. OpenSSL**
  - Neue Distributionen / Mac OS X: „Fixes“
    - Null, Anonymous Ciphers
    - SSLv2
      - Wg. SSL-Poodle: SSLv3 maybe coming?
    - export ciphers (FREAK)
    - weak DH ciphers (Logjam)

3. Code

- Sockets vs. **OpenSSL**
  - OTOH..
    - Verteilung von Binaries
      - Basierend auf Peter Mosmans fork
      - Linux, BSD, Darwin, ARM
    - Borken features / ciphers
    - Advanced features / ciphers
      - 3x Chacha20/Poly1305 cipher
      - `-proxy`, `-xmpphost <host>`, ...
      - Horizont: OpenSSL 1.1.0: CCM Cipher
  - Häßlich: github
  - Docker images

- **Sockets vs. OpenSSL**
  - Beides!

    Sockets, ggw. wo nötig
    - Protokoll check SSLv2 - TLS 1.1
    - TLS time
    - s.o. HB+CCS

3. Shell rulz

- **but is also difficult...**
  - Cmd line parser
    - /bin/bash: `getopts`
      - geht immer (BSDs / Linux)
      - Haken: keine --long Option
    - `/usr/bin/getopt`
      - Linux (util-linux): GNU (long und short options)
      - BSD: halt anders
  - ==> eigener Parser
    - Peter Mosmans

3. Shell rulz

- **Statische Code Analyse**
  - Shellcheck (github.com/koalaman/shellcheck)
  - **Demo:** shellcheck.net
  - **Demo** an testssl.sh

  - Sicherheit:
    - eher zufällig

→ Demo (7)

3. Shell rulz

So old school!

- ~~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

~~sed & awk~~ /bin/bash!

- **In testssl.sh:**
  - **Vieles, noch nicht 100%ig**

- **Tipps**
  - **http://www.cyberciti.biz/tips/bash-shell-parameter-substitution-2.html**
  - **http://tldp.org/LDP/abs/html/string-manipulation.html**
  - **http://wiki.bash-hackers.org/syntax/pe**

3. Shell rulz

- **Script-Farbe**
  - N00b + „erster Blick“ key feature!
  - „Problem“
    - Linux / BSD
    - ANSI + termcap / ncurses
  - 256 Farben?

→ Demo

3. Shell rulz

- **Grenzen**
  - ~~Threads / Events~~
    - Wichtig für borken Server/ Load Balancer etc.
    - Gibt's nicht, Krücke:

```
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
```

→ Pollen, Resultate in `$HEADERFILE, $ERRFILE`

4. Gefahr, Gefahr

```
###########################################################
     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. Gefahr, Gefahr

- Mehr?
  - Threat Modeling:
    - Ins Web stellen, siehe z.B.
      - `testssl.sh -S $(dig +short testssl.sh)`
        - https://www.jacobmansfield.co.uk/p/2015/04/28/testing-for-ssl-vulnerabilities/
      - https://code.google.com/p/google-security-research/issues/detail?id=546
      - (unabhängig voneinander)
    - Command Injection

→ Idee (9)

5. Bugs ähm Features

- Es wird langsam kompliziert…

6. Zusammenfassung

- „Lustiger“ Debian/Ubuntu Bug

```
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%
```

- Sonst: Debian Wheezy, Ubuntu 15.04

5. Bugs ähm Features

- **OpenLiteSpeed**
  - SSLv2: disabled by default
  - Antwortet trotzdem
    - Problem1: Plaintext
    - Problem2: Es gibt keinen RFC-Handshake in SSLv2

→ Demo (10)

- **IIS 6.0**
  - Support ist ausgelaufen
    - (Für einige wohl egal)
    - OpenSSL 1.0.2: Handshake failure
      - handshake-size limit, OpenSSL 1.0.2 hat mehr Cipher

→ Demo (11)

5. Bugs ähm Features

- **Cisco ACE Loadbalancer**
  - Client Hello mit >128 Cipher
  - Again: Handshake Limit
- **Alte F5 BigIP Load Balancer**
  - SSL Request stall
  - Handshake < 256 kBytes
- **F5 SSL Offload Engine (Web Acc)**
  - Header Request stalls

6. Zusammenfassung

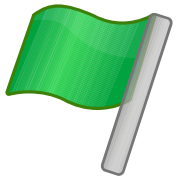
- **Projekt testssl.sh ist „kicking and alive“**
  - Es gibt Releases!
  - Letztes: 2.6, contributions++
    - Proxy
    - TLS_FALLBACK_SCSV
    - Peter Mosmans OpenSSL fork

6. Zusammenfassung

- **Herausforderungen**
  - Verwundbarkeiten: Am Ball bleiben
    - Erwartungen: wird weniger
  - Kaputte Handshakes
    - Drumrum programmieren
  - **Testplattformen!**
    - Plattform-Kompatibilität
    - Serverseite

6. Zusammenfassung

- **So what's new (2.7dev)**



  - Überprüfen der Trust Chain
    - Mozilla / Microsoft / JDK 1.8 / Linux ca-bundle.crt
  - IPv6 support
    - Wie,  erst jetzt ??
    - OpenSSL constraints, don't get me started...

6. Zusammenfassung

- **So what's new (2.7dev)**
  - Support for servers with X509 client authentication
  - SSL Session ID check
  - Smart logging
  - Some workarounds for buggy systems

7. Ausblick

- **Zukunft**
  - Features targeted for 2.8:
    github.com/drwetter/testssl.sh/milestones/2.7dev%20(2.8)%20
    - Mehr Sockets
      - TLS 1.2: extensions
      - Disabled ciphers
    - CN  Hostname Validierung
    - EC Kurven
    - Maschinenlesbarer Output
      - JSON
      - HTML: gibt's bereits über „aha“
    - Rating!

- **Zukunft, cont'd**
  - Interne Verbesserungen
    - Codequalität ;-)
      - Shellcheck!
    - Dokumentation

7. Schlusspointe

- **Bestellung auf Webseite:**
  - **Beste Verschlüsselung**

7. Schlusspointe

- **Bestätigungsmail**
  - Mit allen zuvor eingegebenen Daten

Danke!

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

  mail aet drwetter eu

@drwetter