Debugging TLS issues with Wireshark

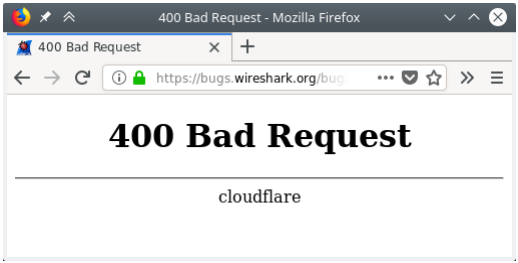
Tuesday June 11th, 2019

Peter Wu
Wireshark Core Developer
peter@lekensteyn.nl

About me

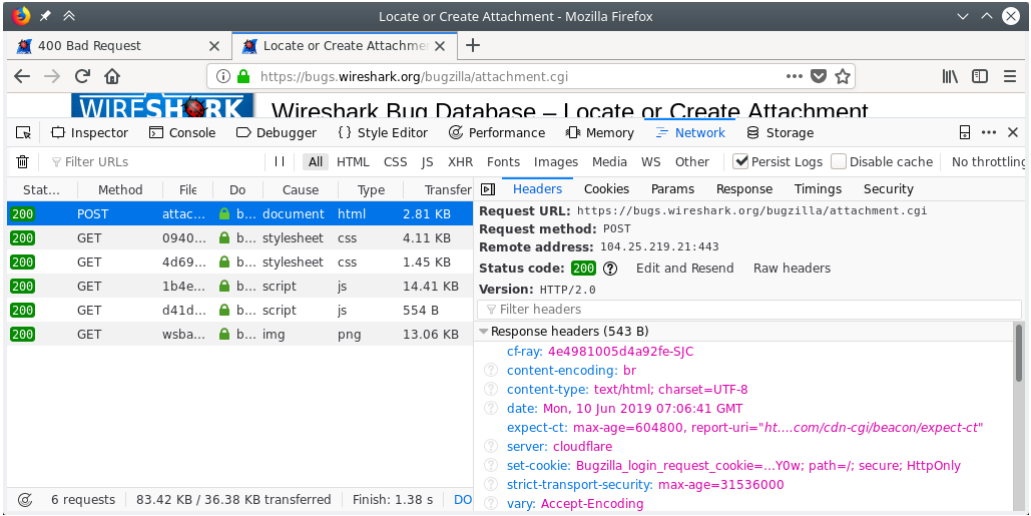
- Wireshark contributor since 2013, core developer since 2015.
- Areas of interest: TLS, Lua, security, ...
- Cloudflare crypto team.

- Problem description: uploading a file to a website failed with a 400 Bad Request.
- Environment: Firefox 61 on Linux.[1]
- Steps to reproduce:
  1. Select file in upload form.
  2. Modify file contents.
  3. Hit the Submit button.
- Expected result: ...
- Actual result: ...

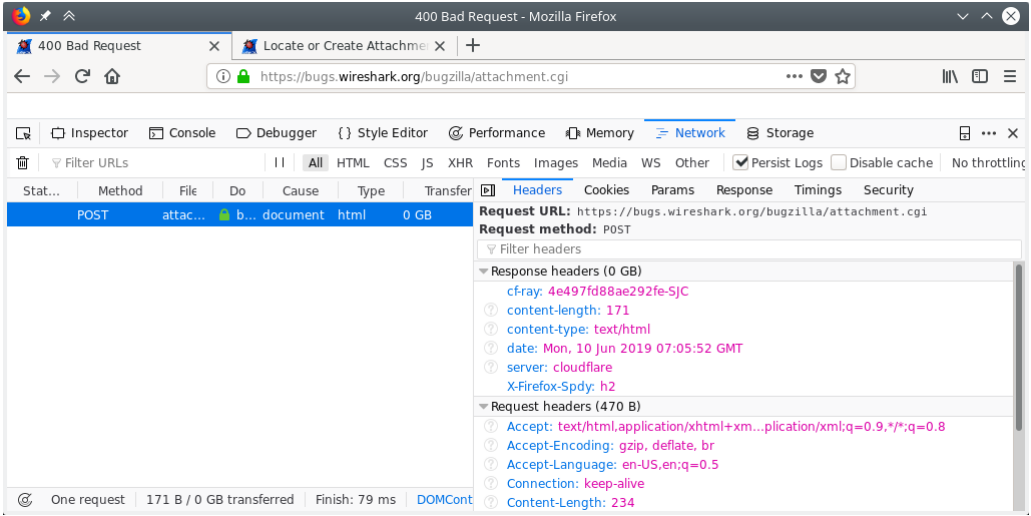


[1] Fixed in Firefox 67, `https://bugzil.la/1459999`

Debug attempt #1: Firefox Developer Tools – expected result

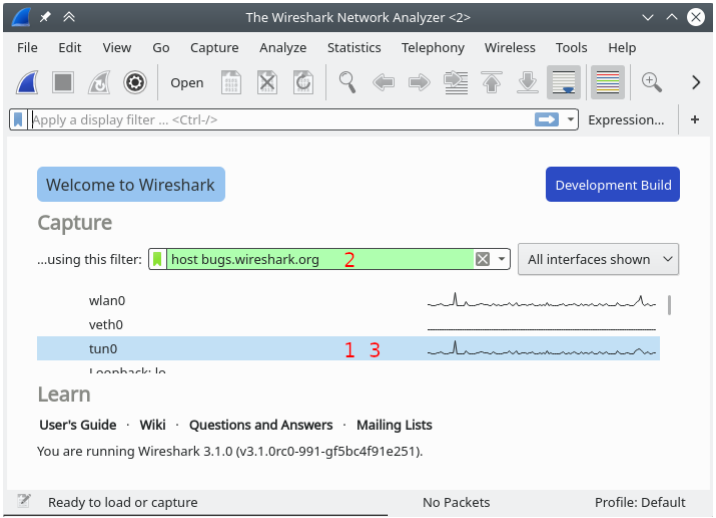


Debug attempt #1: Firefox Developer Tools – actual result



- Application layer protocol: HTTP/2 over TLS (HTTPS).
- To access the decrypted HTTP request, we have to:
  - Capture packets *including the initial TLS handshake*.
  - Capture TLS session secrets to enable decryption.

Wireshark – Capture setup



1. Select network interface.
2. Limit capture file size with a *capture filter*[2].
3. Start capture with Ctrl + E (⌘ + E) or by double-clicking.

[2] `https://www.tcpdump.org/manpages/pcap-filter.7.html`

Capture TLS session secrets into a key log file

- Set environment variable `SSLKEYLOGFILE` **before** starting Firefox or Chrome. Programs will append secrets to a file at this location.
- Firefox on Windows, create `start-fx.cmd` file, without quotes in the `set` line:

```
set SSLKEYLOGFILE=C:\Users\User\Desktop\keys.txt
start firefox
```

- Chrome on Windows, create a shortcut with:

```
chrome --ssl-key-log-file="C:\Users\User\Desktop\keys.txt"
```

- One-liner for Linux and macOS, start Firefox or Chromium with a new profile:

```
SSLKEYLOGFILE="$PWD/keys.txt" firefox -no-remote -profile /tmp/ff
SSLKEYLOGFILE="$PWD/keys.txt" chromium --user-data-dir=/tmp/cr
```

- curl 7.58.0 (Ubuntu 18.04, Fedora 28, Arch Linux):

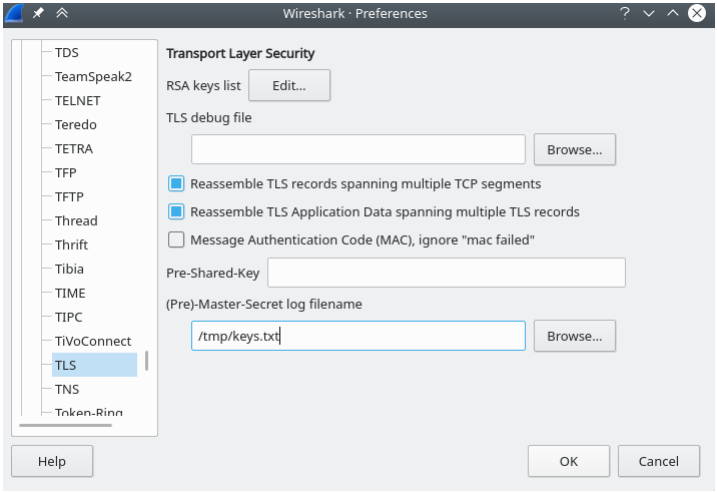
```
export SSLKEYLOGFILE="$PWD/keys.txt"
curl https://example.com
```

- Text file with unique per-session secrets[3].
- TLS 1.2 format: `CLIENT_RANDOM` *<Client Hello Random> <master secret>*.
- TLS 1.3 requires four different secrets (handshake and traffic secrets).
- Check that the file is created and updated, it looks like:

```
CLIENT_RANDOM F8566FD1E091C4CD1583313B04BB2834C817D917FC3BEDC351529BD8CC6A5FD1 9BC6A9D65B89835DB86BD857D08A8D87847F0BE08B88618BCB25A1A
CLIENT_RANDOM CC5A30A4606104A670D0A82B27A112E9BCD05E1A498F7C8445027334157FDFD3 CFCE47C71B69D198BCF63FC4206D16BB9A524C0CB0CACCEA36DC6DD
CLIENT_RANDOM 607AAA3D657D8A08F1073AE75B62CD284C87BB5504D275631CA86533707FB080 B27567070A3832CA2C072D1D0905647EF364C1E017A33001ED0BB2E
CLIENT_HANDSHAKE_TRAFFIC_SECRET e27a03ae85ae8035b331a1af6089dd1e2f300cce131b03fdb9f07a25f1a10876 8ac2e7e210e30e8f660048e20d45209935d6a
SERVER_HANDSHAKE_TRAFFIC_SECRET e27a03ae85ae8035b331a1af6089dd1e2f300cce131b03fdb9f07a25f1a10876 21c21f13865944c2c411ed1a7271809834dbe
CLIENT_TRAFFIC_SECRET_0 e27a03ae85ae8035b331a1af6089dd1e2f300cce131b03fdb9f07a25f1a10876 0de57183beff9a8c43994f517fba1d79ca374bff53b2a
SERVER_TRAFFIC_SECRET_0 e27a03ae85ae8035b331a1af6089dd1e2f300cce131b03fdb9f07a25f1a10876 f26e64d69b8095bbcdcbd04d48f2f9d96aedc1abc6463
EXPORTER_SECRET e27a03ae85ae8035b331a1af6089dd1e2f300cce131b03fdb9f07a25f1a10876 3ab0346dcf11212792839c1f89c9e05aed7b159e680b7a5057189
```

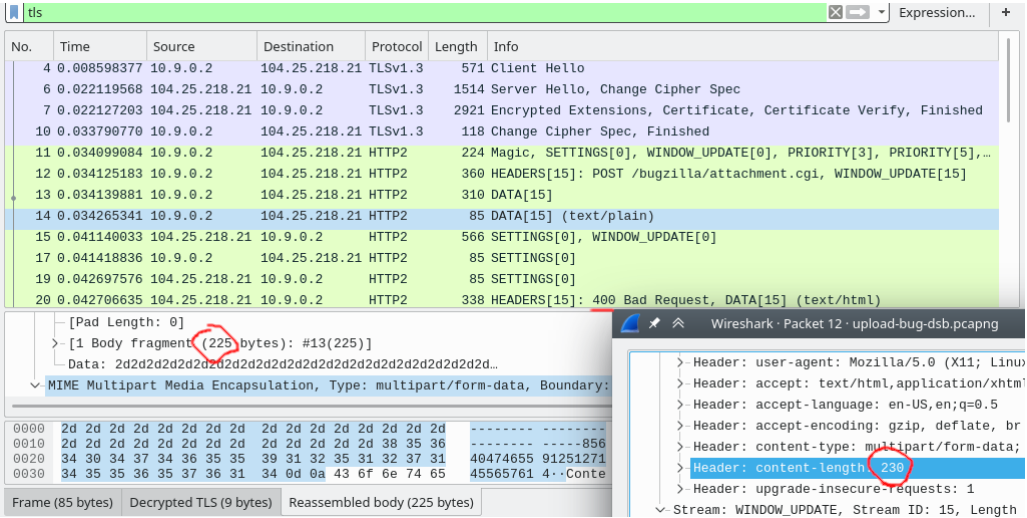
[3] File format at `https://developer.mozilla.org/NSS_Key_Log_Format`

Configure Key Log File in Wireshark




```
tshark -otls.keylog_file:/tmp/keys.txt -r some.pcapng
```

Debug attempt #2: analyze packets with Wireshark – results



https://lekensteyn.nl/files/captures/upload-bug-dsb.pcapng

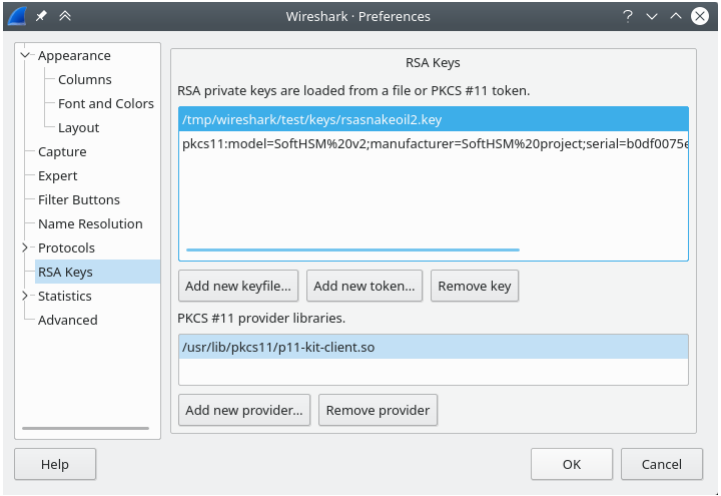
- TLS decryption requires pairing capture files with key log files. This makes switching between different files and file distribution more difficult.
- Solution in Wireshark 3.0: embed key log file in a **pcapng** file.
- `editcap --inject-secrets tls,keys.txt in.pcap out-dsb.pcapng`
- Replace secrets: `editcap --discard-all-secrets --inject-secrets ...`
- inject-tls-secrets.sh: script to embed a subset of TLS secrets in a pcapng file.[4]
  Example: given keys.txt and some.pcap, create some-dsb.pcapng:
  `./inject-tls-secrets.sh keys.txt some.pcap`

[4] `https://gist.github.com/Lekensteyn/f64ba6d6d2c6229d6ec444647979ea24`

RSA decryption keys

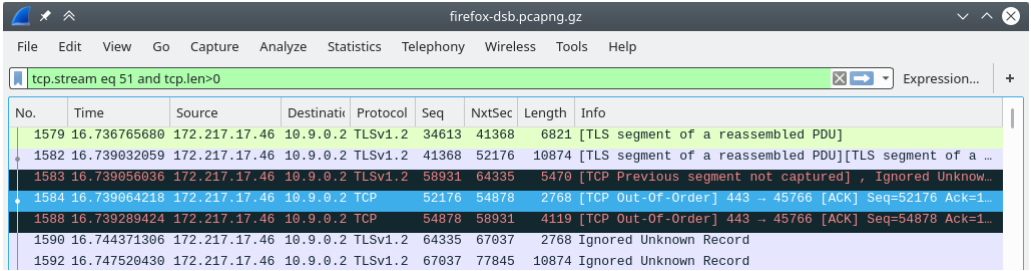
- What if TLS key log file is not supported, for example on Windows applications?
- Solution: decryption through RSA private keys.
- Advantage over key log: decrypt all traffic after configuring the private key once.
- Limitations:
  - Requires server admin to provide the key file.
  - Does not work with ciphers like TLS_EC**DHE**_RSA_WITH_AES_128_GCM_SHA256.
  - Does not work with session resumption.
  - Does not work with TLS 1.3.
- Danger: Leaking the private key compromises all previous and future traffic (RSA ciphers are not *forward secret*).

RSA Keys configuration



- New in Wireshark 3.0.
- Replaces RSA keys list in TLS preferences.
- Simplified interface.
- PKCS#11 token and HSM support.
- Accepts passwordless PEM-encoded or PKCS#12 key file.

Caveat: out-of-order TCP segments break decryption



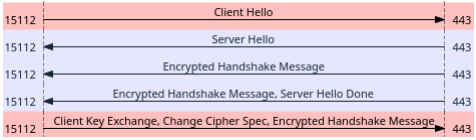
- Enable these **TCP** protocol preferences:
  - Allow subdissector to reassemble TCP streams.
  - Reassemble out-of-order segments (since Wireshark 3.0, disabled by default).[5]
- Sample capture: `https://lekensteyn.nl/files/firefox-google/`

[5] `https://www.wireshark.org/docs/wsug_html_chunked/ChAdvReassemblySection.html#ChAdvReassemblyTcp`

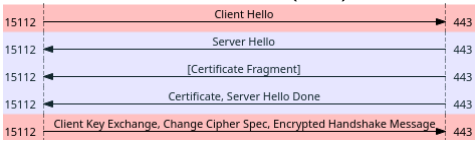
Caveat: large certificates are not properly displayed

- Handshake fragmentation may break dissection and TLS 1.3 decryption.
- Capture sample: bug3303.cap[6]

Bug in Wireshark 3.0:



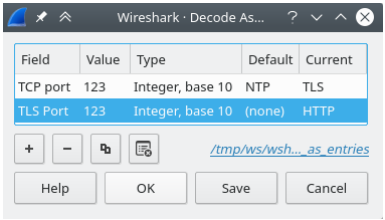
Fixed in Wireshark 3.1 (dev):




```
> [3 Reassembled TCP Segments (3355 bytes): #24(1460), #25(1460), #27(435)]
v Transport Layer Security
  v TLSv1 Record Layer: Handshake Protocol: Certificate
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 3350
      Handshake Protocol: Certificate (last fragment)
    > [2 Reassembled Handshake Fragments (19734 bytes): #22(16384), #27(3350)]
    > Handshake Protocol: Certificate
```

[6] https://bugs.wireshark.org/bugzilla/show_bug.cgi?id=3303#c8

- Wireshark detects TLS through heuristics, but standard port registrations take precedence. Use *Decode As* functionality to set an explicit protocol.
- Example: HTTPS on TCP server port 123.
  - Right-click TCP layer, *Decode As*. Change current protocol for **TCP Port** to *TLS*.
  - Press *OK* to apply just for now or *Save* to persist this port-to-protocol mapping.
  - Right-click SSL layer, *Decode As*. Change current protocol for **TLS Port** to *HTTP*.
- For STARTTLS protocols, select SMTP/IMAP/... instead of TLS for *TCP Port*.
- Tip: there are many protocols, just select the field, then use arrow keys or type the protocol name (typing *H* gives *HTTP*).



- Sample packet capture firefox-esni.pcap and key log file firefox-esni.keys: `https://bugs.wireshark.org/bugzilla/show_bug.cgi?id=14984`
- To enable in Firefox, open *about:config*.
  - Enable ESNI: set *network.security.esni.enabled* to *true*.
  - Enable DoH: set *network.trr.mode* to *2* (try trusted recursive resolver first).[7]
- A public key is retrieved using DNS Queries over HTTPS (DoH) – RFC 8484.
- The plain text server name extension is replaced by an Encrypted Server Name Indication (ESNI) extension – *draft*-ietf-tls-esni-01.
- DoH encrypts the server name. TLS 1.3 encrypts the server Certificate, ESNI additionally hides the server name.

[7] `https://daniel.haxx.se/blog/2018/06/03/inside-firefoxs-doh-engine/`

- Not to be confused with Google QUIC (gquic in Wireshark).
- The current QUIC draft (20) relies on TLS 1.3 for security.
- Almost everything is encrypted now (including Client Hello).
- QUIC is a transport protocol (compare it to TLS).
- HTTP/2 is based on TCP/TLS. HTTP/3 will use UDP/QUIC.
- Sample capture ngtcp2-19-dsb.pcapng:
  `https://bugs.wireshark.org/bugzilla/show_bug.cgi?id=13881#c209`
- Status of QUIC in Wireshark:
  `https://github.com/quicwg/base-drafts/wiki/Tools#wireshark`

- Use a key log file to enable TLS decryption in Wireshark.
- Embed these secrets in a pcapng file for easier distribution.
- Enable TCP reassembly preferences to enable decryption.
- Use the latest Wireshark version for the best results.
- For a more detailed background and key extraction from other applications, see `https://lekensteyn.nl/files/wireshark-ssl-tls-decryption-secrets-sharkfest18eu.pdf`

peter@lekensteyn.nl
lekensteyn.nl
@Lekensteyn