



SharkFest '19 US



Debugging TLS issues with Wireshark

Tuesday June 11th, 2019

Peter Wu

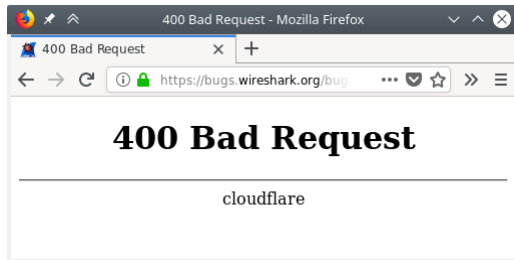
Wireshark Core Developer
peter@lekensteyn.nl



- ▶ 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: ...



¹Fixed in Firefox 67, <https://bugzil.la/1459999>



Debug attempt #1: Firefox Developer Tools – expected result



400 Bad Request

Locate or Create Attachment

https://bugs.wireshark.org/bugzilla/attachment.cgi

WIRESHARK

Wireshark Bug Database – Locate or Create Attachment

Inspector

Console

Debugger

Style Editor

Performance

Memory

Network

Storage

Filter URLs

All

HTML

CSS

JS

XHR

Fonts

Images

Media

WS

Other

☒ Persist Logs

☐ Disable cache

No throttling

Stat...	Method	File	Do	Cause	Type	Transfer
200	POST	attac...	b...	document	html	2.81 KB
200	GET	0940...	b...	stylesheet	css	4.11 KB
200	GET	4d69...	b...	stylesheet	css	1.45 KB
200	GET	1b4e...	b...	script	js	14.41 KB
200	GET	d41d...	b...	script	js	554 B
200	GET	wsba...	b...	img	png	13.06 KB

Headers

Cookies

Params

Response

Timings

Security

Request URL: https://bugs.wireshark.org/bugzilla/attachment.cgi

Request method: POST

Remote address: 104.25.219.21:443

Status code: 200 ? Edit and Resend Raw headers

Version: HTTP/2.0

Filter headers

Response headers (543 B)

cf-ray: 4e4981005d4a92fe-SJC

content-encoding: br

content-type: text/html; charset=UTF-8

date: Mon, 10 Jun 2019 07:06:41 GMT

expect-ct: max-age=604800, report-uri="https://bugs.wireshark.org/bugzilla/attachment.cgi/beacon/expect-ct"

server: cloudflare

set-cookie: Bugzilla_login_request_cookie=...Y0w; path=/; secure; HttpOnly

strict-transport-security: max-age=31536000

vary: Accept-Encoding

6 requests

83.42 KB / 36.38 KB transferred

Finish: 1.38 s

DO



Debug attempt #1: Firefox Developer Tools – actual result



400 Bad Request - Mozilla Firefox

400 Bad Request x Locate or Create Attachment x +

https://bugs.wireshark.org/bugzilla/attachment.cgi

Inspector Console Debugger Style Editor Performance Memory Network Storage

Filter URLs All HTML CSS JS XHR Fonts Images Media WS Other Persist Logs Disable cache No throttling

Stat...	Method	File	Do	Cause	Type	Transfer
	POST	attach...	b...	document	html	0 GB

Request URL: https://bugs.wireshark.org/bugzilla/attachment.cgi
Request method: POST

Filter headers

Response headers (0 GB)

- cf-ray: 4e497fd88ae292fe-SJC
- content-length: 171
- content-type: text/html
- date: Mon, 10 Jun 2019 07:05:52 GMT
- server: cloudflare
- X-Firefox-Spdy: h2

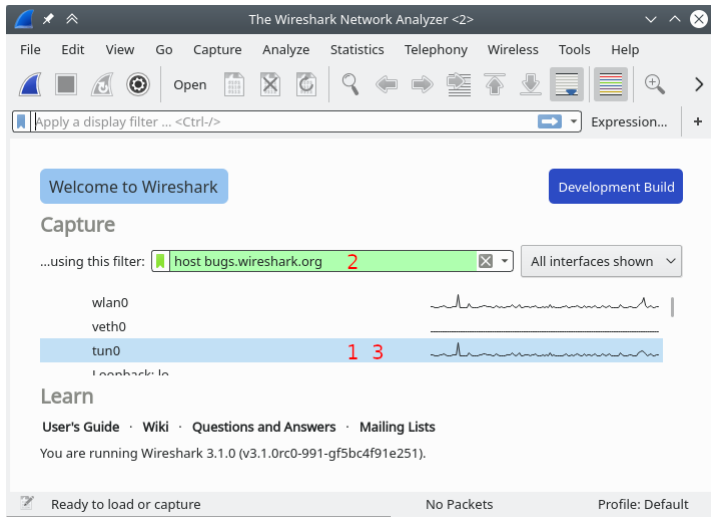
Request headers (470 B)

- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
- Accept-Encoding: gzip, deflate, br
- Accept-Language: en-US,en;q=0.5
- Connection: keep-alive
- Content-Length: 234

One request | 171 B / 0 GB transferred | Finish: 79 ms | DOMCont



- ▶ 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.



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>



- ▶ 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
```



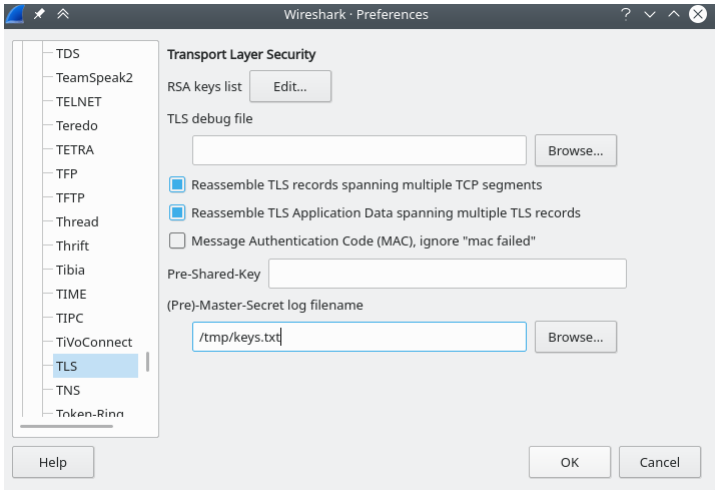

- ▶ Text file with unique per-session secrets³.
- ▶ TLS 1.2 format: CLIENT_RANDOM <Client Hello Random> <master secret>.
- ▶ TLS 1.3 requires four different secrets (handshake and traffic secrets).
- ▶ Check that the file is created and updated, it looks like:

```
CLIENT_RANDOM F8566FD1E091C4CD1583313B04BB2834C817D917FC3BEDC351529BD8CC6A5FD1 9BC6A9D65B89835DB86BD857D08A8D87847F0BE08B88618BCB25A1A
CLIENT_RANDOM CC5A30A4606104A670D0A82B27A112E9BCD05E1A498F7C8445027334157DFDD3 CFCE47C71B69D198BCF63FC4206D16BB9A524C0CB0ACCEA36DC6DD
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
```

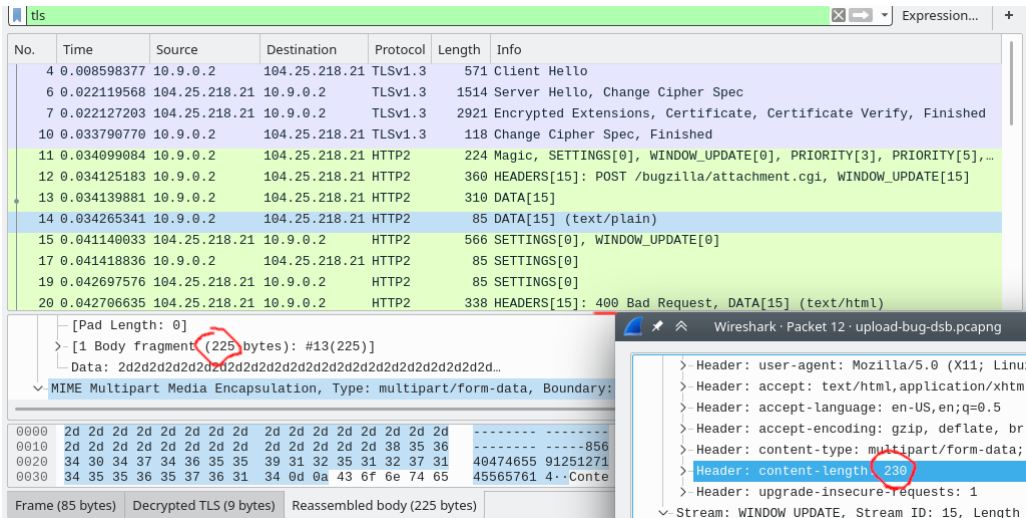
³File format at https://developer.mozilla.org/NSS_Key_Log_Format



Configure Key Log File in Wireshark



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



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

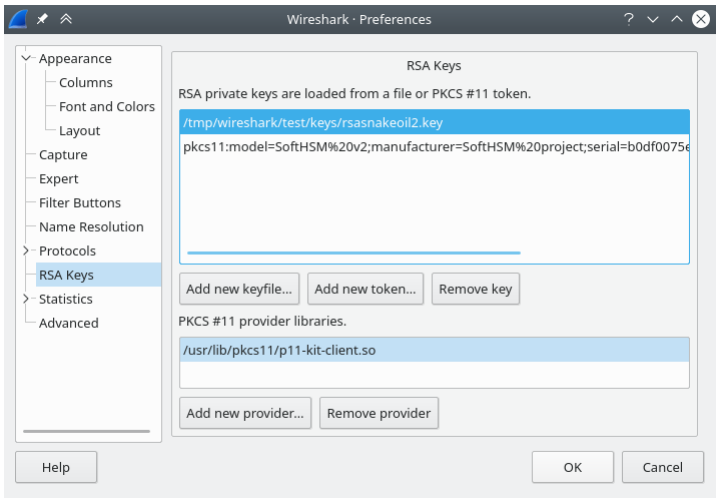


- ▶ TLS decryption requires pairing capture files with key log files. This makes switching between different files and file distribution more difficult.
- ▶ Solution in Wireshark 3.0: embed key log file in a **pcapng** file.
- ▶ `editcap --inject-secrets tls,keys.txt in.pcap out-dsb.pcapng`
- ▶ Replace 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>



- ▶ 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_ECDHE_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*).



- ▶ 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



No.	Time	Source	Destination	Protocol	Seq	NextSeq	Length	Info
1579	16.736765680	172.217.17.46	10.9.0.2	TLSv1.2	34613	41368	6821	[TLS segment of a reassembled PDU]
1582	16.739032059	172.217.17.46	10.9.0.2	TLSv1.2	41368	52176	10874	[TLS segment of a reassembled PDU][TLS segment of a ...
1583	16.739056036	172.217.17.46	10.9.0.2	TLSv1.2	58931	64335	5470	[TCP Previous segment not captured] , Ignored Unknown...
1584	16.739064218	172.217.17.46	10.9.0.2	TCP	52176	54878	2768	[TCP Out-Of-Order] 443 → 45766 [ACK] Seq=52176 Ack=1...
1588	16.739289424	172.217.17.46	10.9.0.2	TCP	54878	58931	4119	[TCP Out-Of-Order] 443 → 45766 [ACK] Seq=54878 Ack=1...
1590	16.744371306	172.217.17.46	10.9.0.2	TLSv1.2	64335	67037	2768	Ignored Unknown Record
1592	16.747520430	172.217.17.46	10.9.0.2	TLSv1.2	67037	77845	10874	Ignored Unknown Record

- ▶ 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/>

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

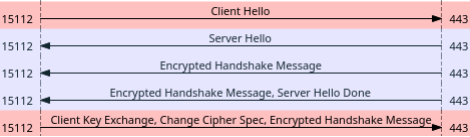


Caveat: large certificates are not properly displayed

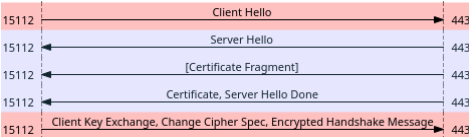


- ▶ Handshake fragmentation may break dissection and TLS 1.3 decryption.
- ▶ Capture sample: bug3303.cap⁶

Bug in Wireshark 3.0:



Fixed in Wireshark 3.1 (dev):

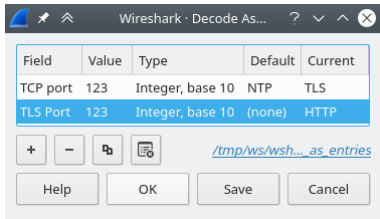


```
> [3 Reassembled TCP Segments (3355 bytes): #24(1460), #25(1460), #27(435)]
└─ Transport Layer Security
   └─ TLSv1 Record Layer: Handshake Protocol: Certificate
      └─ Content Type: Handshake (22)
      └─ Version: TLS 1.0 (0x0301)
      └─ Length: 3350
      └─ Handshake Protocol: Certificate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

⁶https://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