

friTap: Decrypting TLS Traffic on the Fly



Daniel Baier

Francois Egner

Max J. Ufer

daniel.baier@fkie.fraunhofer.de

francois.egner@fkie.fraunhofer.de

max.jens.ufer@fkie.fraunhofer.de

Introduction

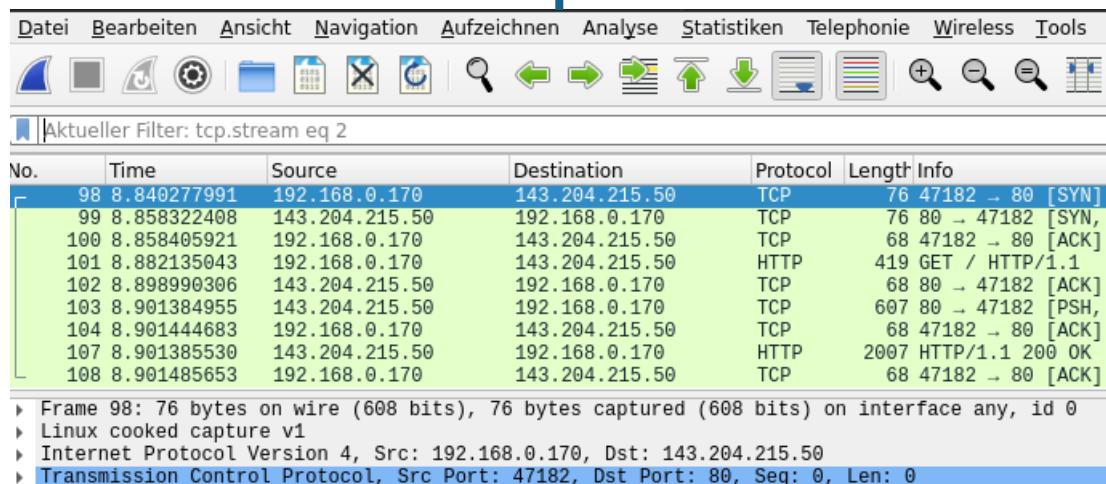


Introduction

Client Client



Server



Datei Bearbeiten Ansicht Navigation Aufzeichnen Analyse Statistiken Telefonie Wireless Tools

Aktueller Filter: tcp.stream eq 2

No.	Time	Source	Destination	Protocol	Length	Info
98	8.840277991	192.168.0.170	143.204.215.50	TCP	76	47182 → 80 [SYN]
99	8.858322408	143.204.215.50	192.168.0.170	TCP	76	80 → 47182 [SYN, ACK]
100	8.858405921	192.168.0.170	143.204.215.50	TCP	68	47182 → 80 [ACK]
101	8.882135043	192.168.0.170	143.204.215.50	HTTP	419	GET / HTTP/1.1
102	8.898990306	143.204.215.50	192.168.0.170	TCP	68	80 → 47182 [ACK]
103	8.901384955	143.204.215.50	192.168.0.170	TCP	607	80 → 47182 [PSH, ACK]
104	8.901444683	192.168.0.170	143.204.215.50	TCP	68	47182 → 80 [ACK]
107	8.901385530	143.204.215.50	192.168.0.170	HTTP	2007	HTTP/1.1 200 OK
108	8.901485653	192.168.0.170	143.204.215.50	TCP	68	47182 → 80 [ACK]

Frame 98: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.168.0.170, Dst: 143.204.215.50
Transmission Control Protocol, Src Port: 47182, Dst Port: 80, Seq: 0, Len: 0

Introduction

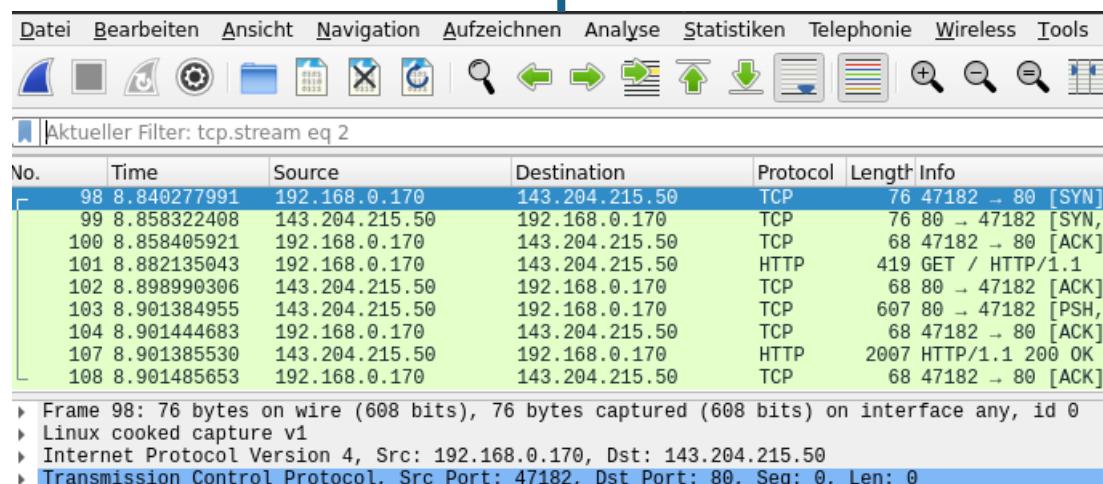
Client



Client



Server



No.	Time	Source	Destination	Protocol	Length	Info
98	8.840277991	192.168.0.170	143.204.215.50	TCP	76	47182 → 80 [SYN]
99	8.858322408	143.204.215.50	192.168.0.170	TCP	76	80 → 47182 [SYN, ACK]
100	8.858405921	192.168.0.170	143.204.215.50	TCP	68	47182 → 80 [ACK]
101	8.882135043	192.168.0.170	143.204.215.50	HTTP	419	GET / HTTP/1.1
102	8.898990306	143.204.215.50	192.168.0.170	TCP	68	80 → 47182 [ACK]
103	8.901384955	143.204.215.50	192.168.0.170	TCP	607	80 → 47182 [PSH, ACK]
104	8.901444683	192.168.0.170	143.204.215.50	TCP	68	47182 → 80 [ACK]
107	8.901385530	143.204.215.50	192.168.0.170	HTTP	2007	HTTP/1.1 200 OK
108	8.901485653	192.168.0.170	143.204.215.50	TCP	68	47182 → 80 [ACK]

Frame 98: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.168.0.170, Dst: 143.204.215.50
Transmission Control Protocol, Src Port: 47182, Dst Port: 80, Seq: 0, Len: 0



NeverSSL

What?

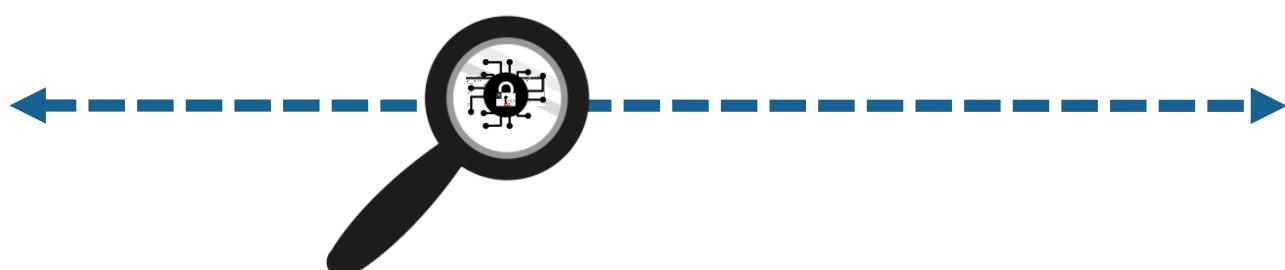
This website is for when you try to open Facebook, Google, Amazon, etc on a wifi network, and nothing happens. Type "http://neverssl.com" into your browser's url bar, and you'll be able to log on.

How?

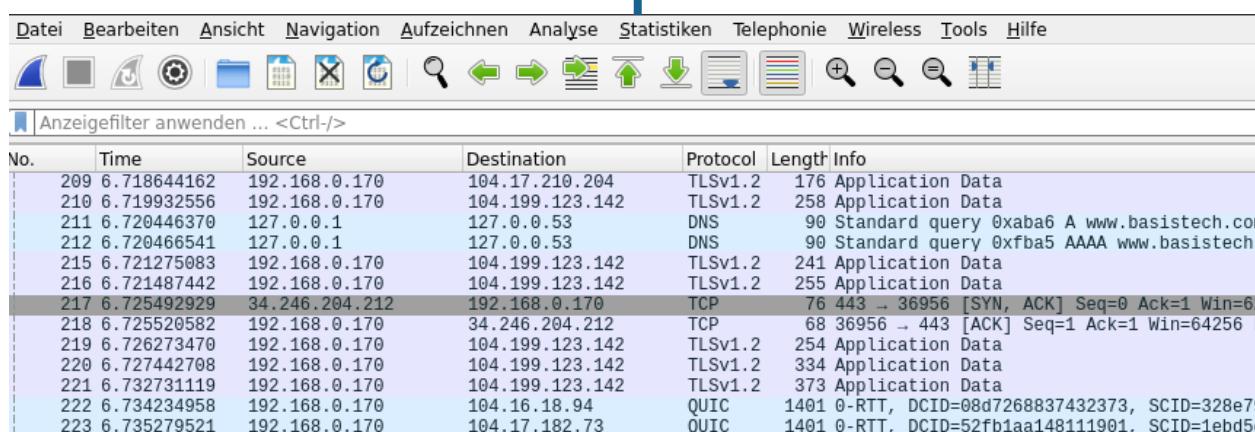
neverssl.com will never use SSL (also known as TLS). No encryption, no strong authentication, no HSTS, no HTTP/2.0, just plain old unencrypted HTTP and forever stuck in the dark ages of internet security.

Challenge: SSL/TLS

Client Client



Server



No.	Time	Source	Destination	Protocol	Length	Info
209	6.718644162	192.168.0.170	104.17.210.204	TLSv1.2	176	Application Data
210	6.719932556	192.168.0.170	104.199.123.142	TLSv1.2	258	Application Data
211	6.720446370	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xaba6 A www.basistech.com
212	6.720466541	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xfbfa5 AAAA www.basistech.com
215	6.721275083	192.168.0.170	104.199.123.142	TLSv1.2	241	Application Data
216	6.721487442	192.168.0.170	104.199.123.142	TLSv1.2	255	Application Data
217	6.725492929	34.246.204.212	192.168.0.170	TCP	76	443 → 36956 [SYN, ACK] Seq=0 Ack=1 Win=62
218	6.725520582	192.168.0.170	34.246.204.212	TCP	68	36956 → 443 [ACK] Seq=1 Ack=1 Win=64256 L
219	6.726273470	192.168.0.170	104.199.123.142	TLSv1.2	254	Application Data
220	6.727442708	192.168.0.170	104.199.123.142	TLSv1.2	334	Application Data
221	6.732731119	192.168.0.170	104.199.123.142	TLSv1.2	373	Application Data
222	6.734234958	192.168.0.170	104.16.18.94	QUIC	1401	0-RTT, DCID=08d7268837432373, SCID=328e79
223	6.735279521	192.168.0.170	104.17.182.73	QUIC	1401	0-RTT, DCID=52fb1aa148111901, SCID=1ebd5c



Challenge: SSL/TLS

Client Client



Nowadays most network traffic is encrypted. Mostly utilizing
SSL/TLS 😞

Server



Datei Bearbeiten Ansicht Navigation Aufzeichnen Analyse Statistiken Telefonie Wireless Tools Hilfe

Anzeigefilter anwenden ... <Ctrl-/>

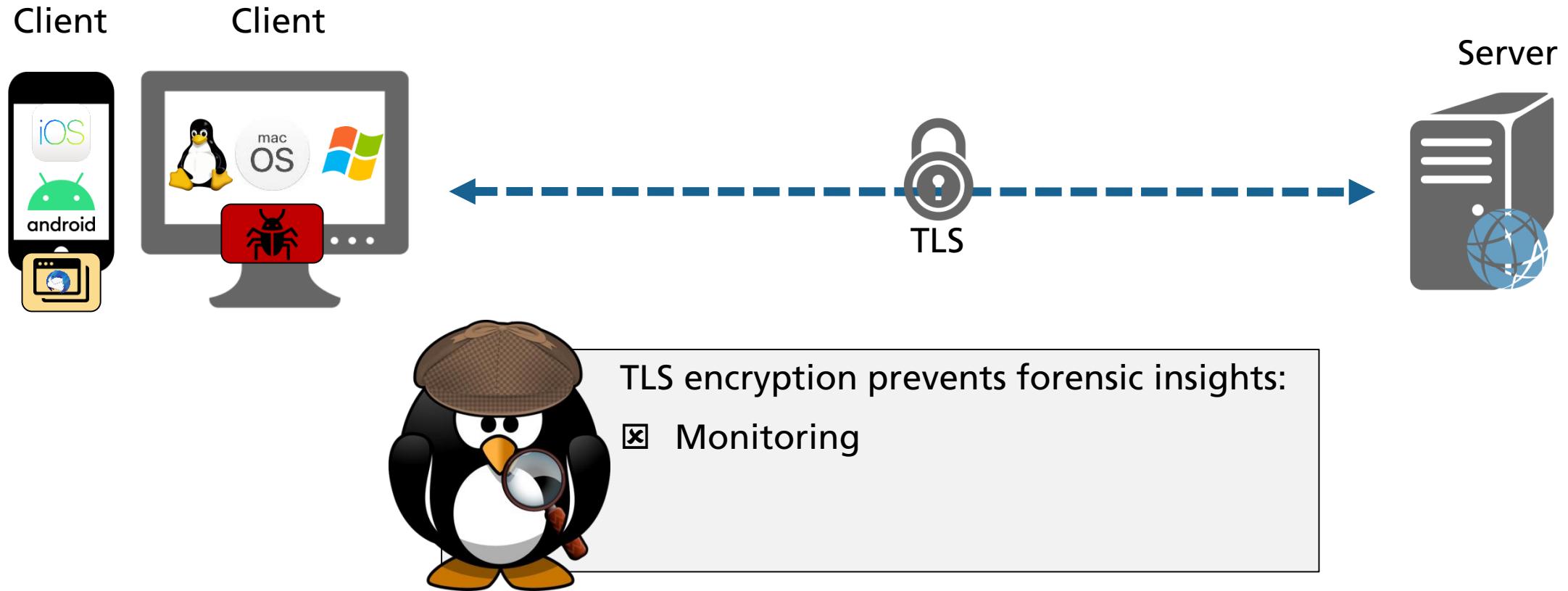
No.	Time	Source	Destination	Protocol	Length	Info
209	6.718644162	192.168.0.170	104.17.210.204	TLSv1.2	176	Application Data
210	6.719932556	192.168.0.170	104.199.123.142	TLSv1.2	258	Application Data
211	6.720446370	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xaba6 A www.basistech.com
212	6.720466541	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xfbfa5 AAAA www.basistech.com
215	6.721275083	192.168.0.170	104.199.123.142	TLSv1.2	241	Application Data
216	6.721487442	192.168.0.170	104.199.123.142	TLSv1.2	255	Application Data
217	6.725492929	34.246.204.212	192.168.0.170	TCP	76	443 → 36956 [SYN, ACK] Seq=0 Ack=1 Win=62
218	6.725520582	192.168.0.170	34.246.204.212	TCP	68	36956 → 443 [ACK] Seq=1 Ack=1 Win=64256 L
219	6.726273470	192.168.0.170	104.199.123.142	TLSv1.2	254	Application Data
220	6.727442708	192.168.0.170	104.199.123.142	TLSv1.2	334	Application Data
221	6.732731119	192.168.0.170	104.199.123.142	TLSv1.2	373	Application Data
222	6.734234958	192.168.0.170	104.16.18.94	QUIC	1401	0-RTT, DCID=08d7268837432373, SCID=328e79
223	6.735279521	192.168.0.170	104.17.182.73	QUIC	1401	0-RTT, DCID=52fb1aa148111901, SCID=1ebd5c



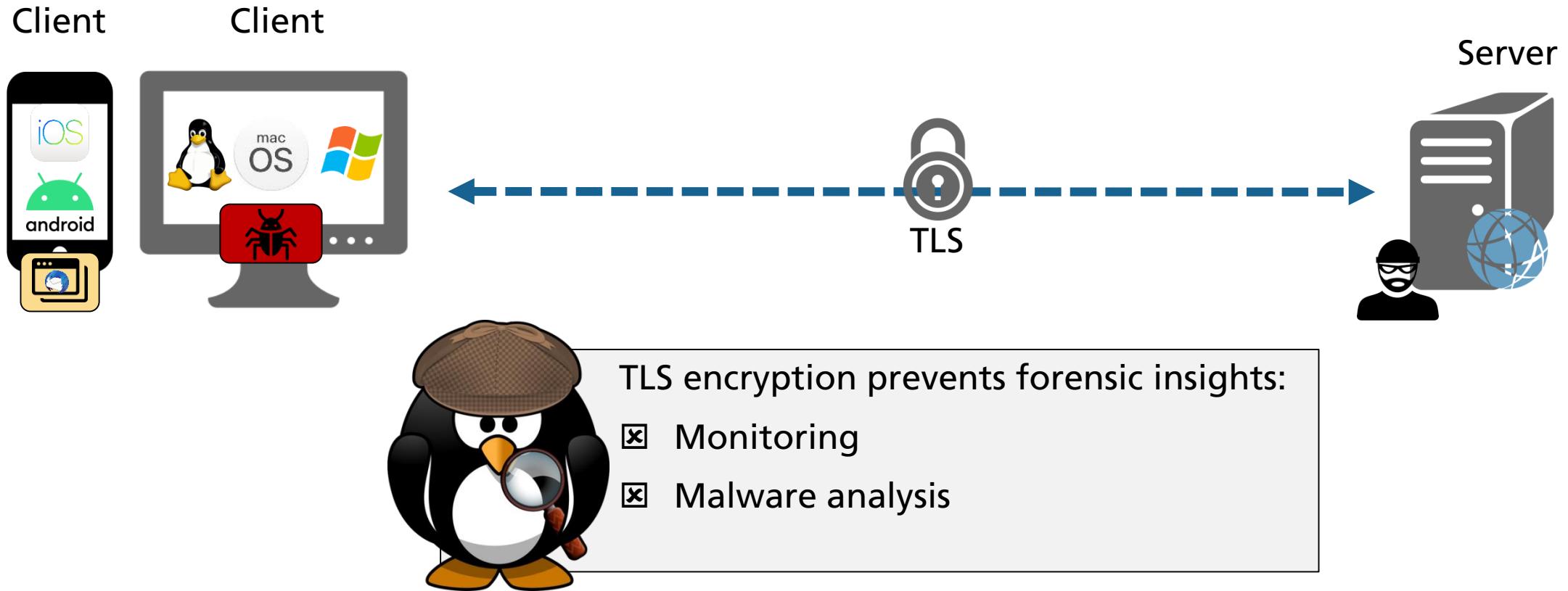
This is perfect for our privacy but...



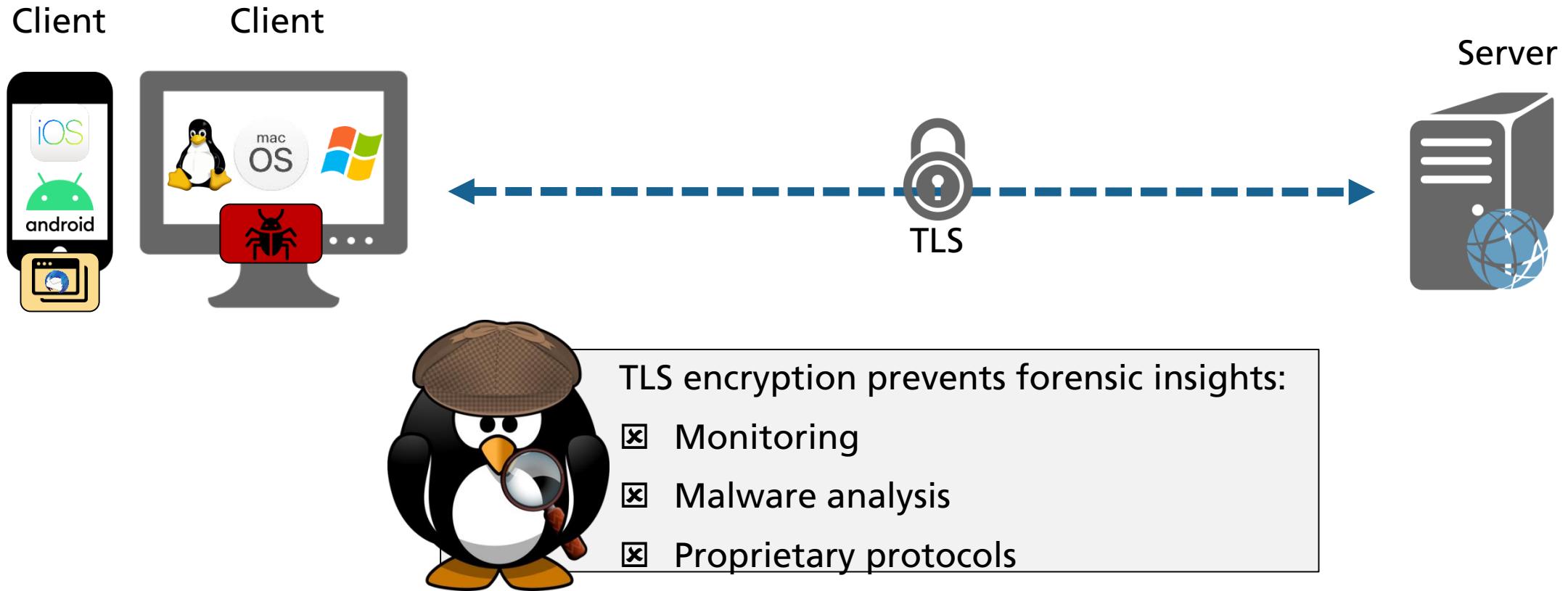
Challenge: Traffic analysis



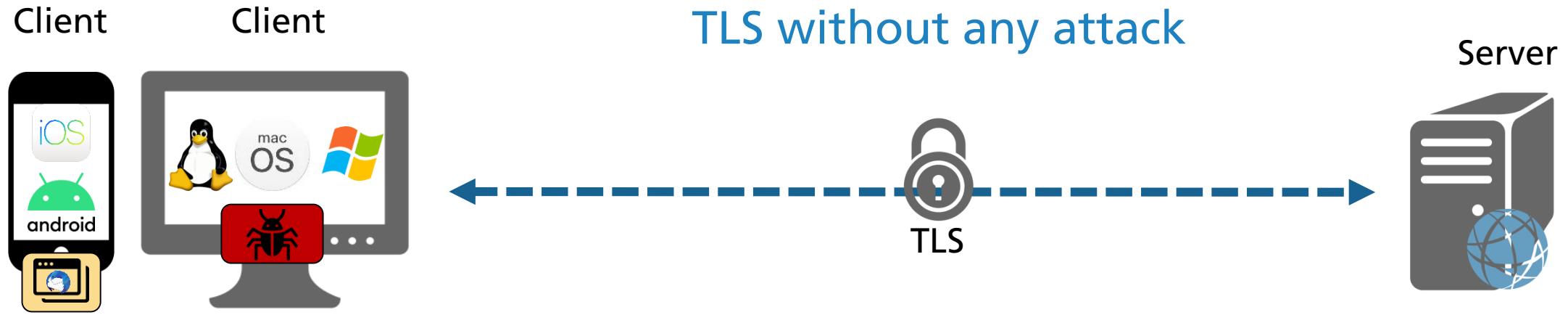
Challenge: Traffic analysis



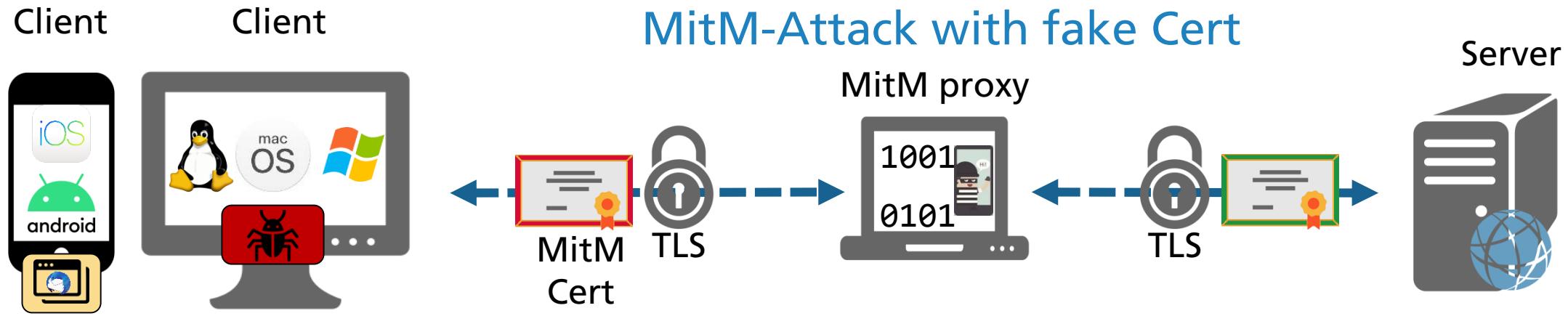
Challenge: Traffic analysis



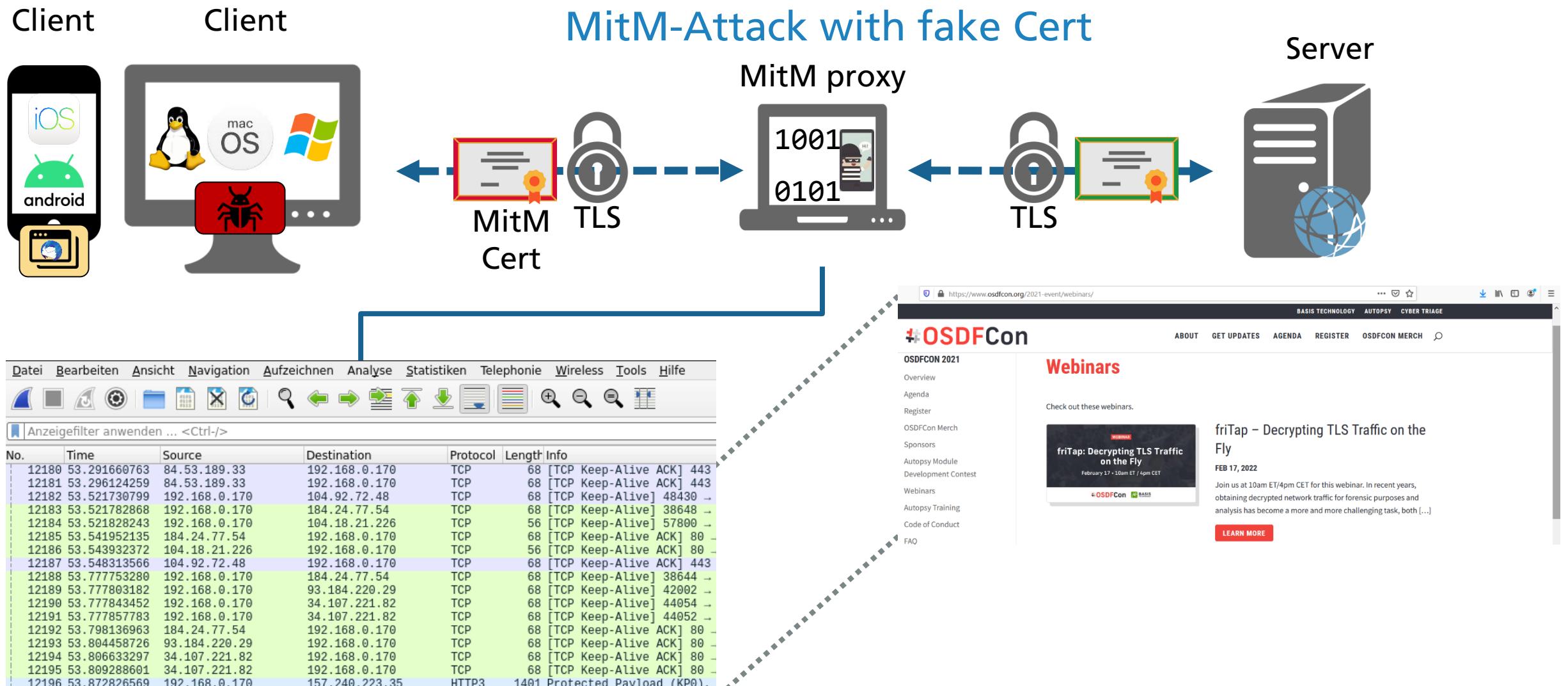
Current state: MitM proxy



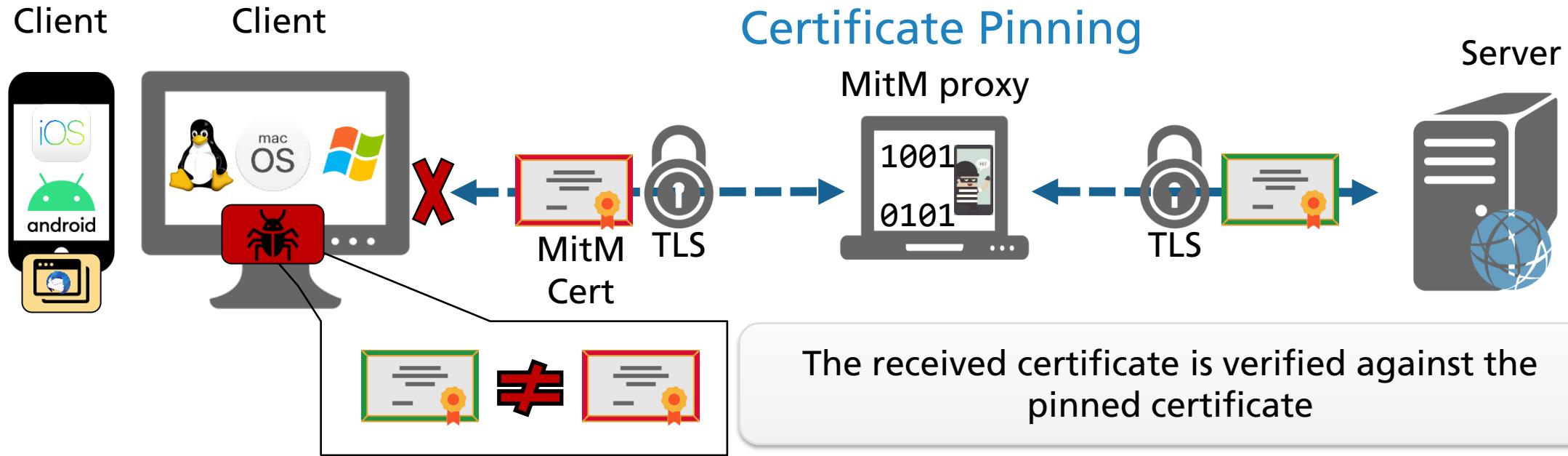
Current state: MitM proxy



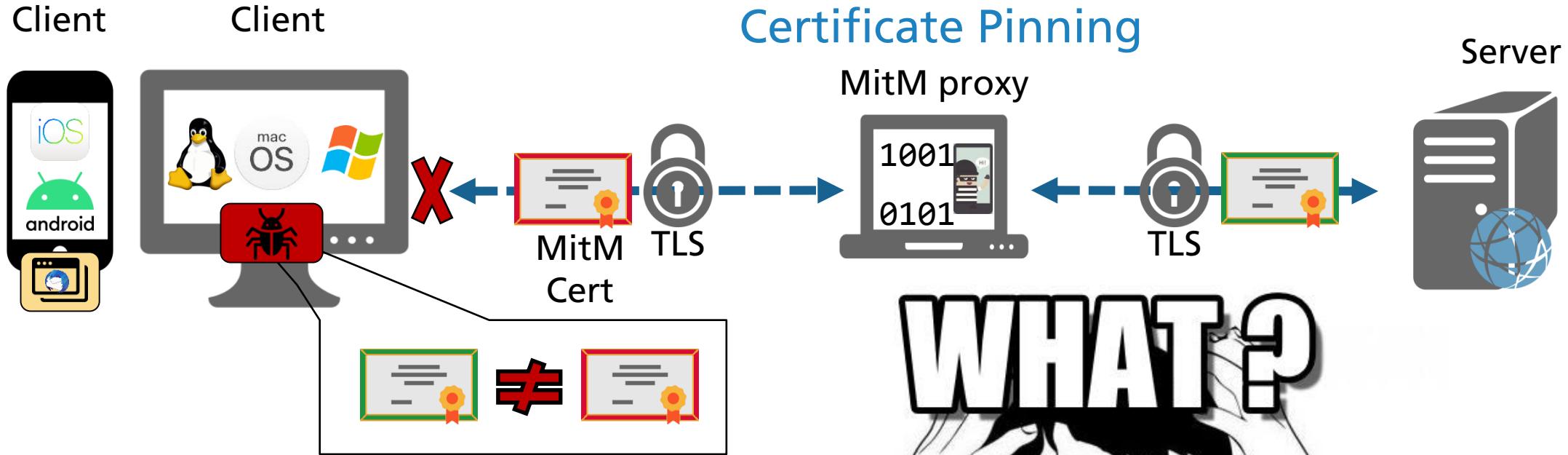
Current state: MitM proxy



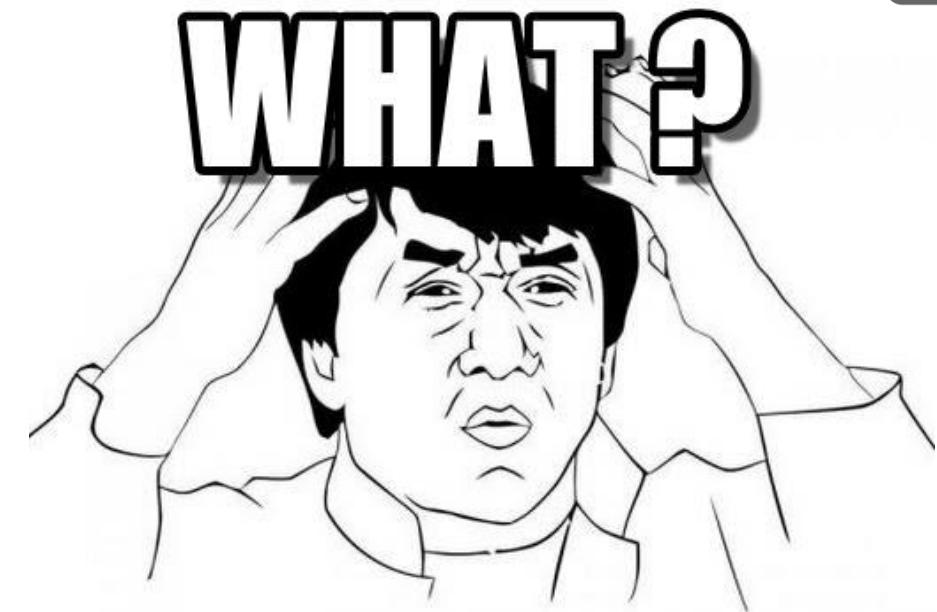
Current state: MitM proxy



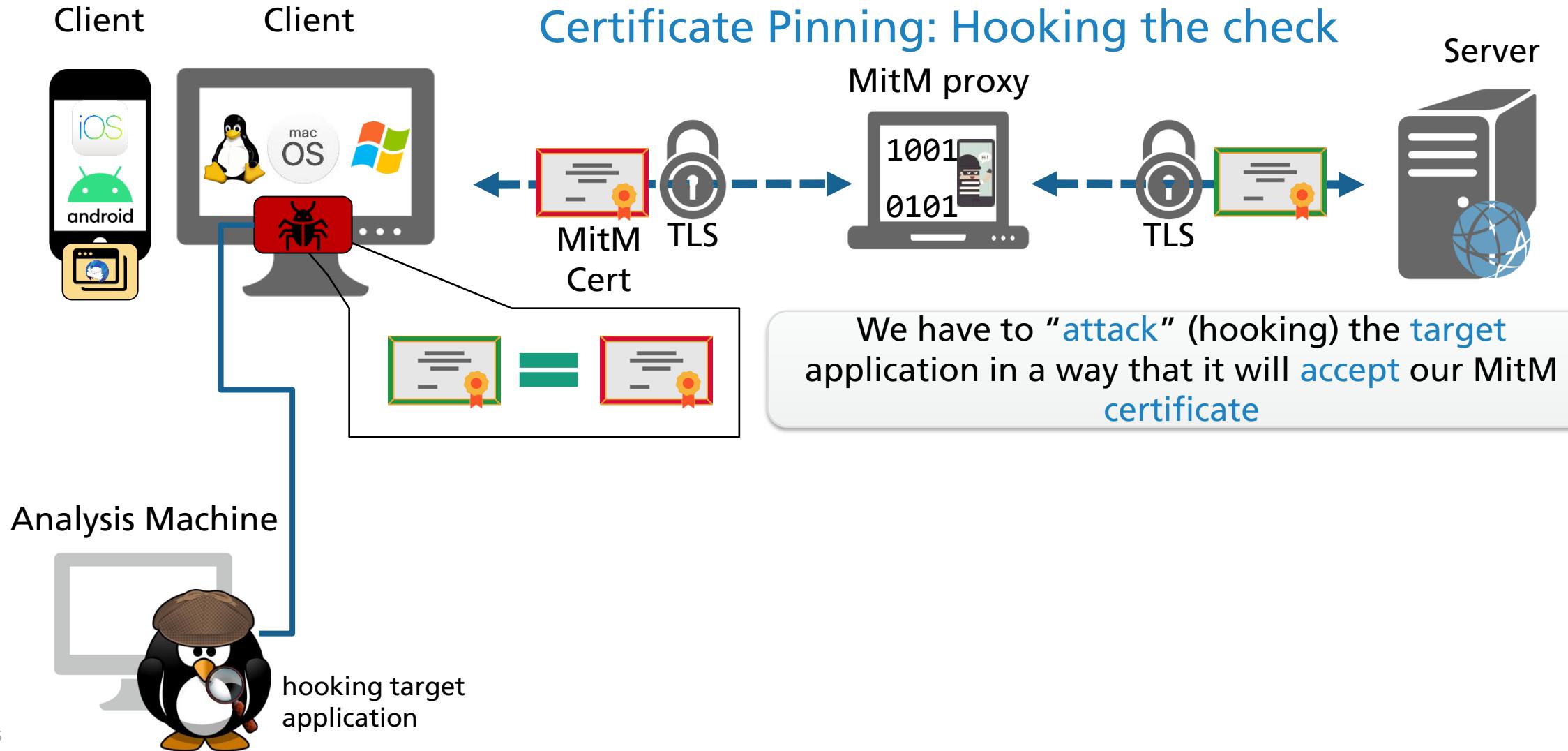
Current state: MitM proxy



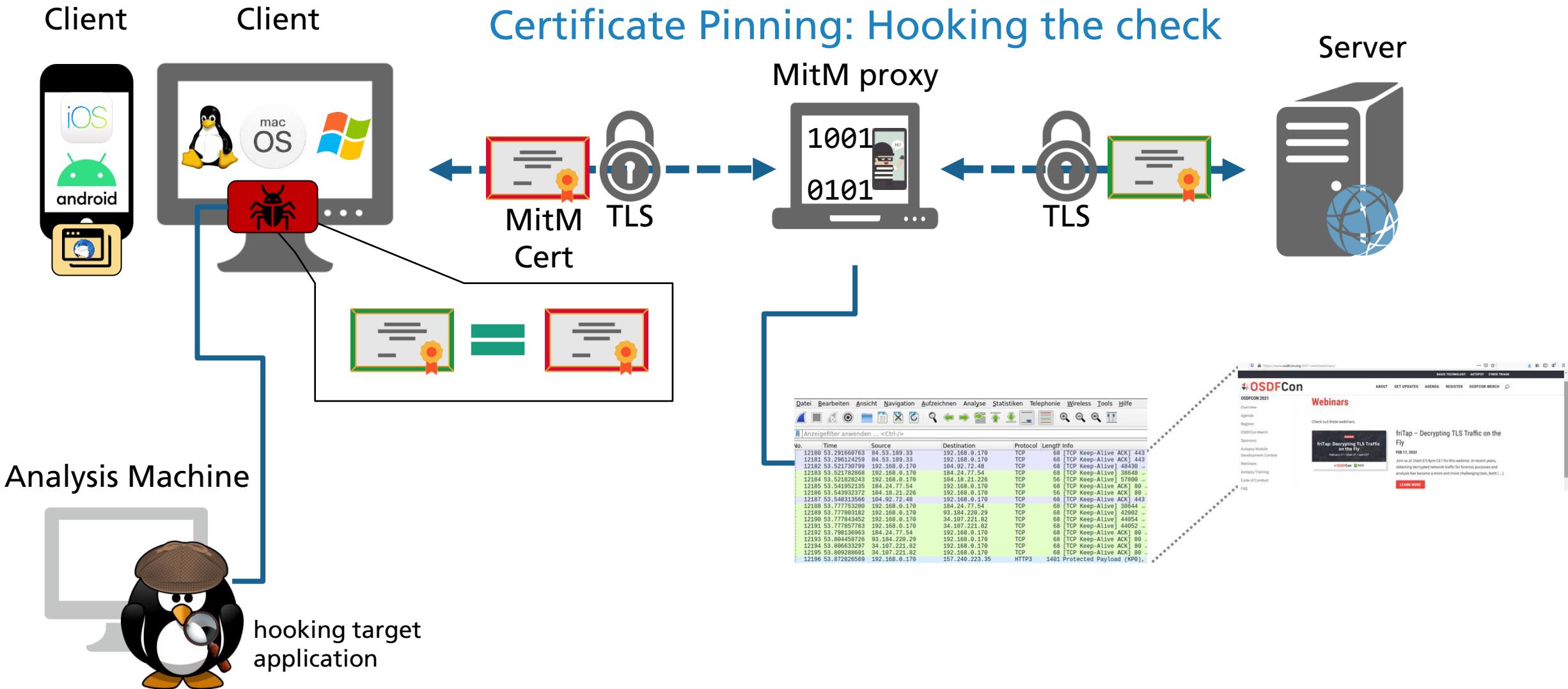
Without **disabling** this **check** in the **application** we are not able to redirect the traffic into our MitM proxy



Current state: MitM proxy

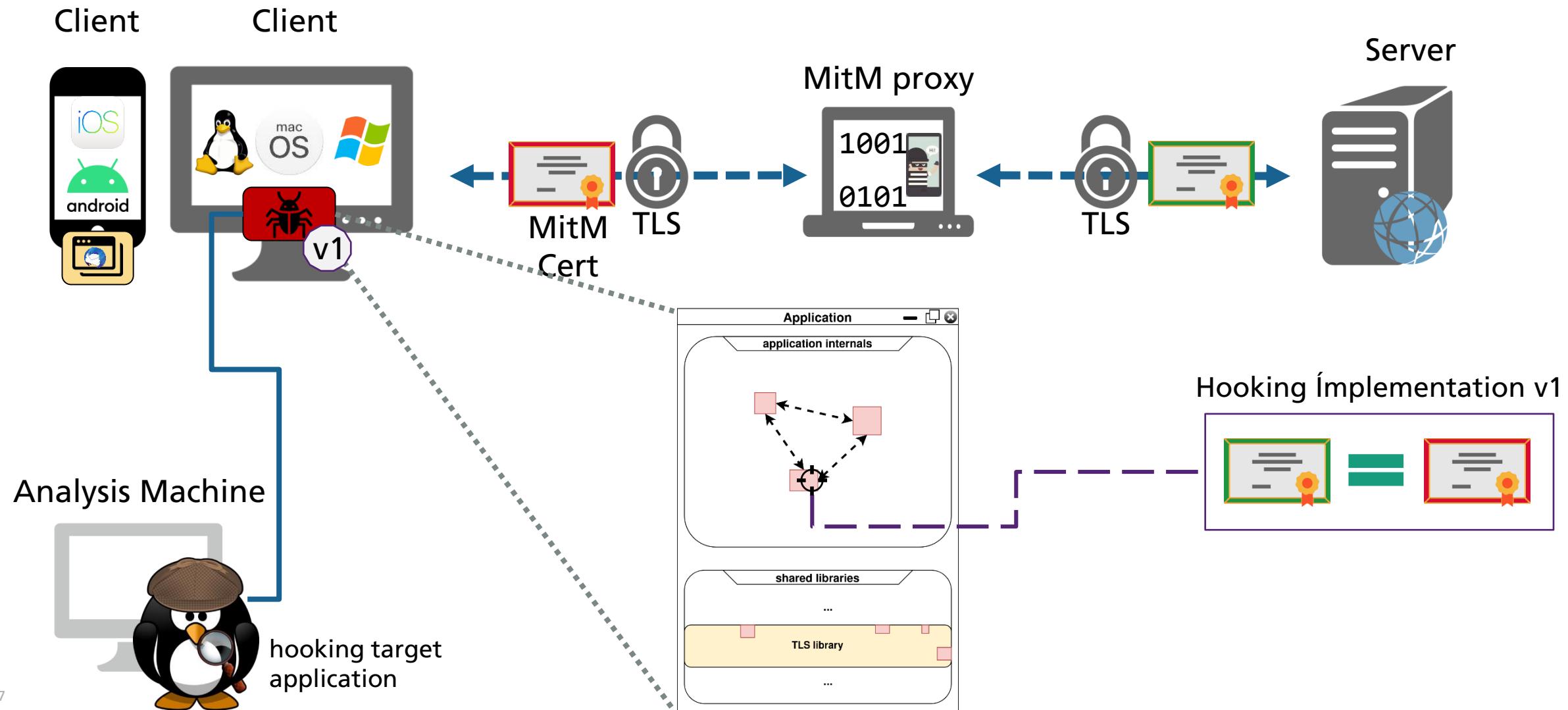


Current state: MitM proxy



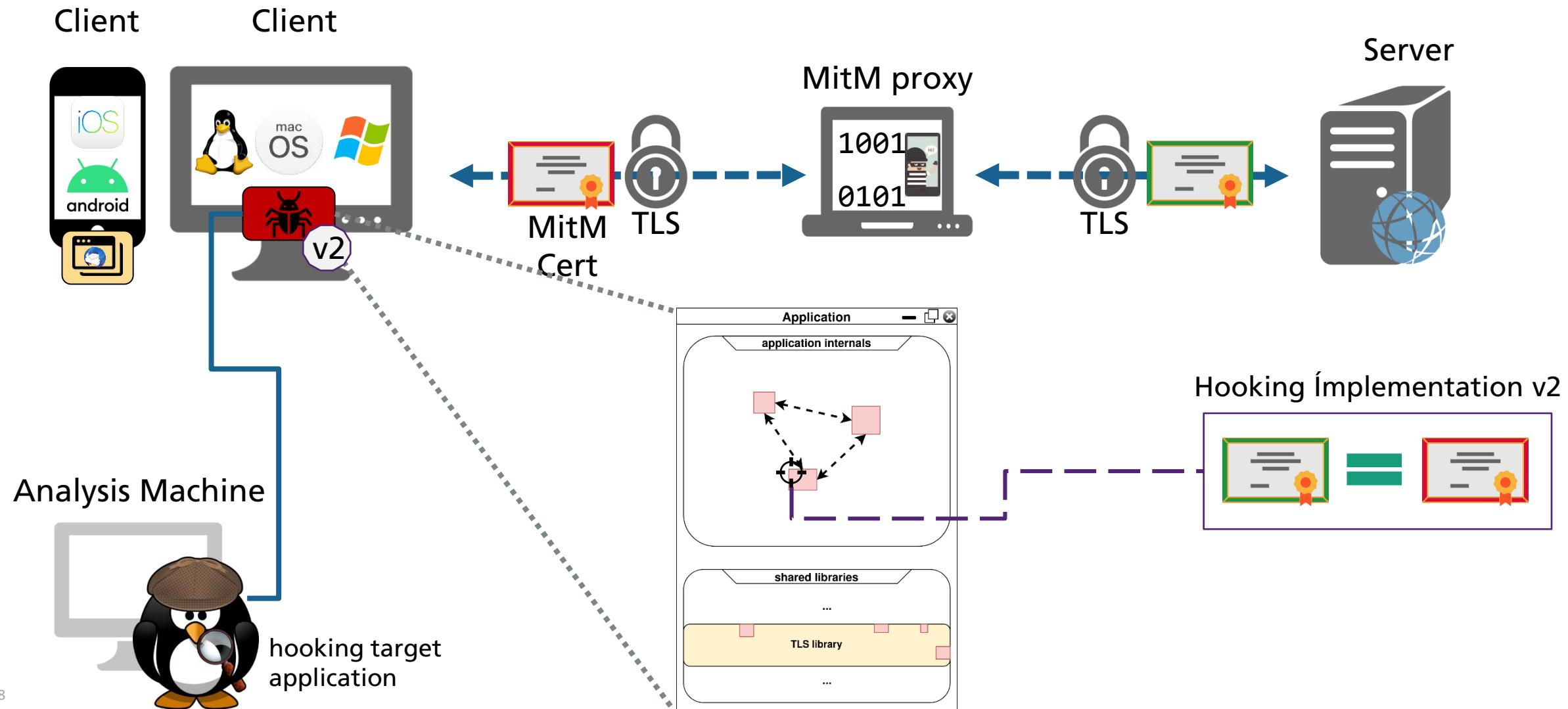
Problem: Certificate Pinning implementations

There are many changes between different versions



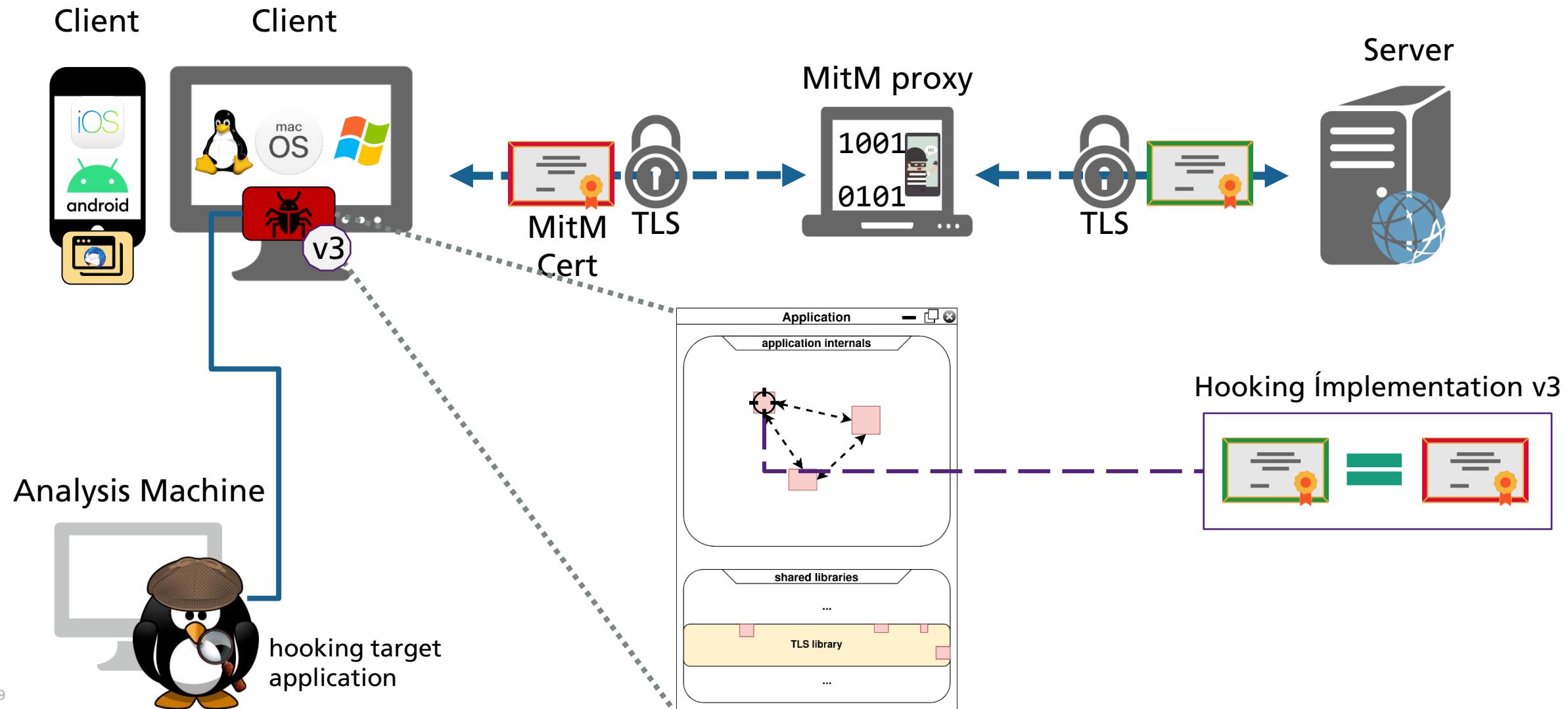
Problem: Certificate Pinning implementations

There are many changes between different versions



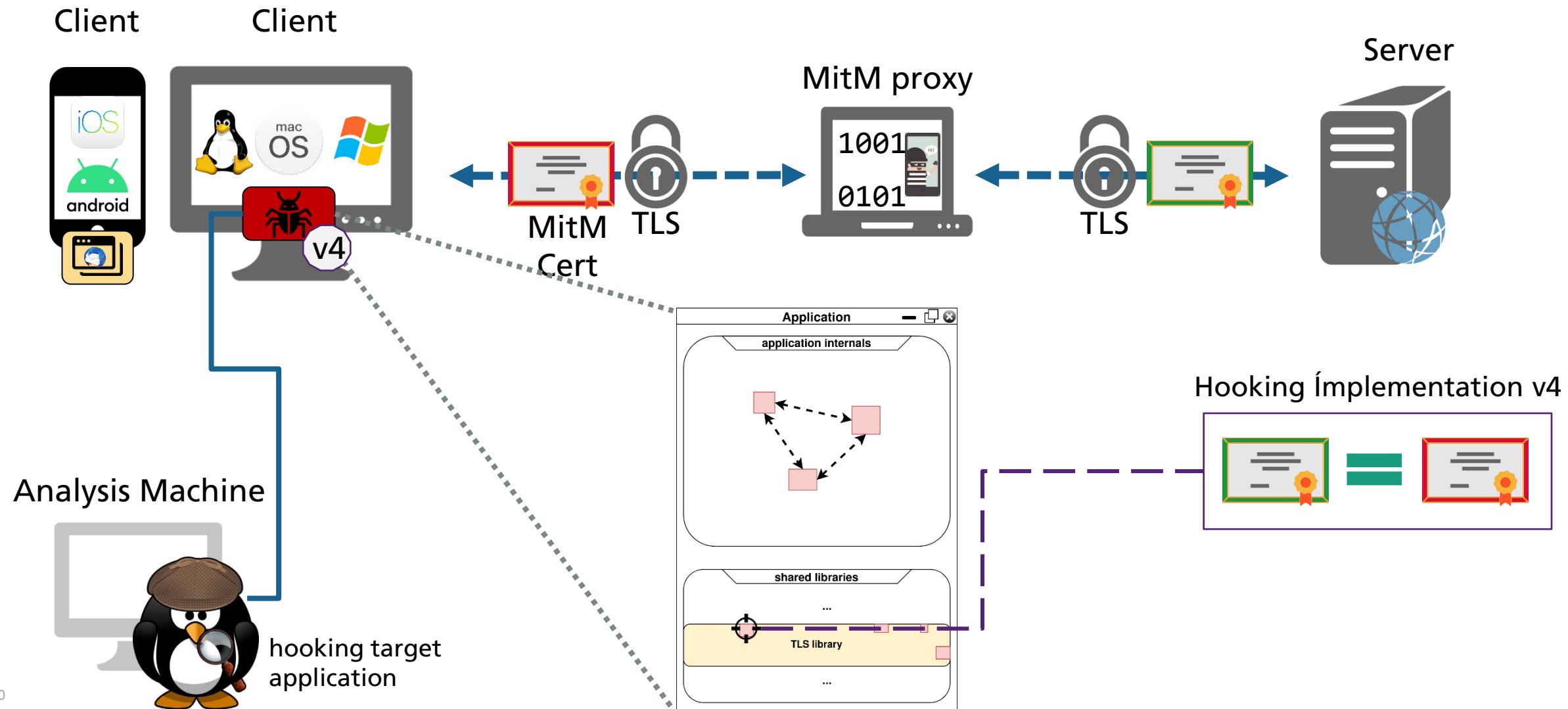
Problem: Certificate Pinning implementations

There are many changes between different versions



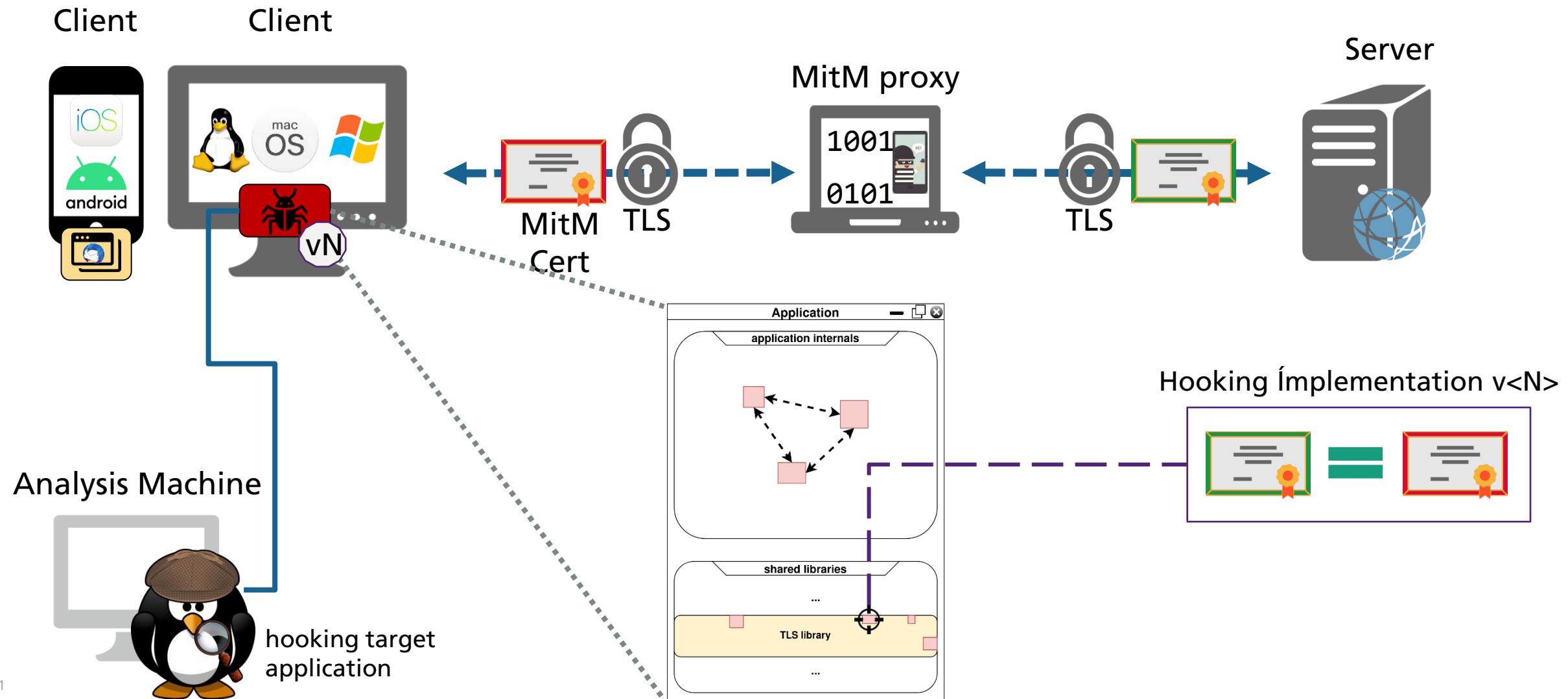
Problem: Certificate Pinning implementations

There are many changes between different versions



Problem: Certificate Pinning implementations

There are many changes between different versions



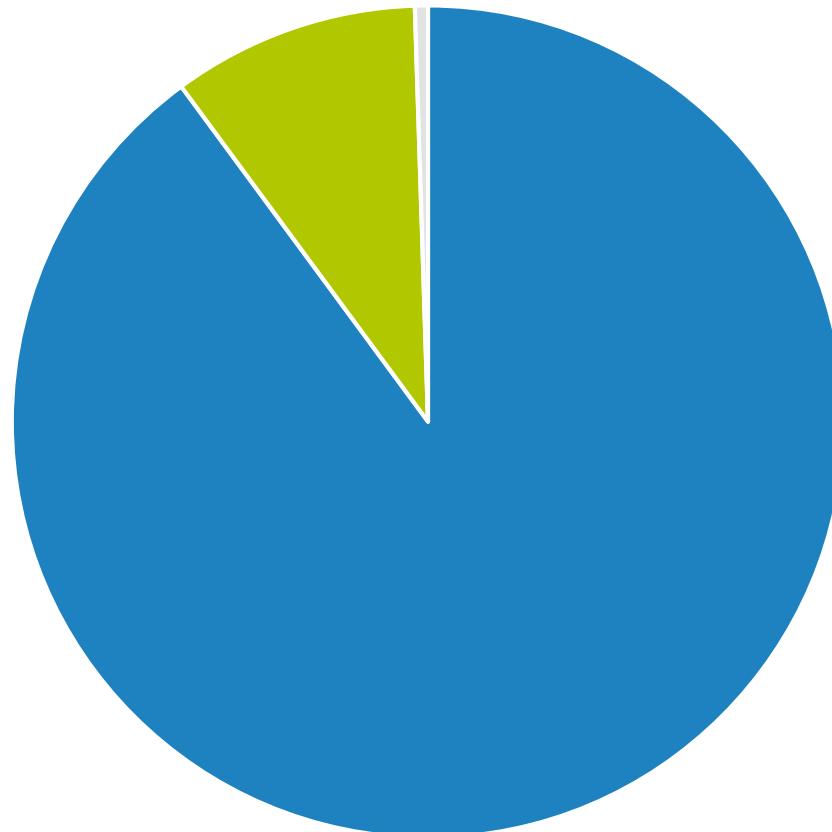


When we have to „attack“ the App why not directly extract the payload of the TLS stream?

Solution: friTap



SSL Implementations on android



- 1. BoringSSL
- 2. GmsCore_OpenSSL
- 3. Other (Spongycastle, Conscrypt, ...)

Solution: friTap



<https://github.com/fkie-cad/friTap>



- TLS keys
- Decypted TLS payload as PCAP

```
FKIE ➤ ~/DEF/research/friTap ➤ ./friTap.py -m -k TLS_keys.log <target_app>
```

```
Usage: friTap.py [-m] [-d] [-k <path>] [-l] [-p <path>] [-s] [-env <env.json>] [-v] [--enable_spawn_gating] <executable/app name/pid>
```

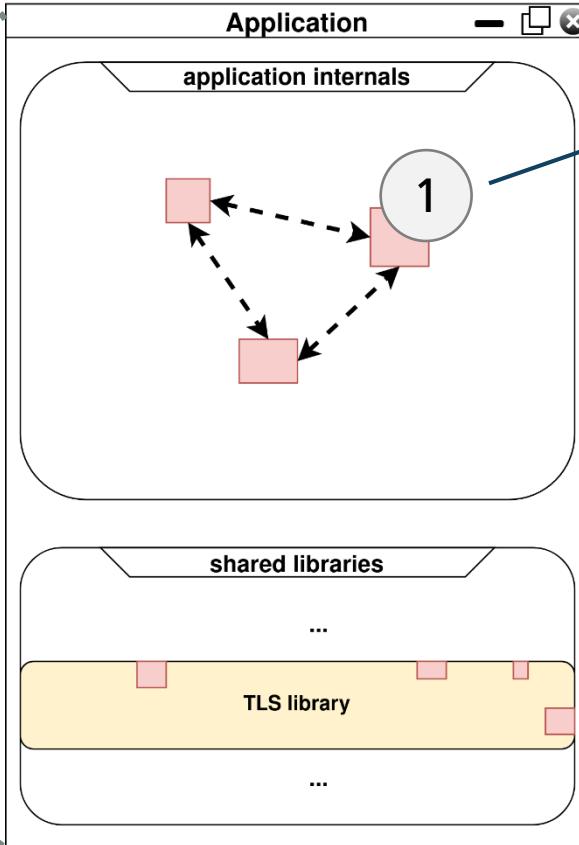
Decrypts and logs an executables or mobile applications SSL/TLS traffic.

Arguments:

- m, --mobile Attach to a process on android or ios
- d, --debug Set the debug output of friTap
- k <path>, --keylog <path>
Log the keys used for tls traffic
- l, --live Creates a named pipe /tmp/sharkfin which can be read by Wireshark during the capturing process
- p <path>, --pcap <path>

friTap: Approach

Client

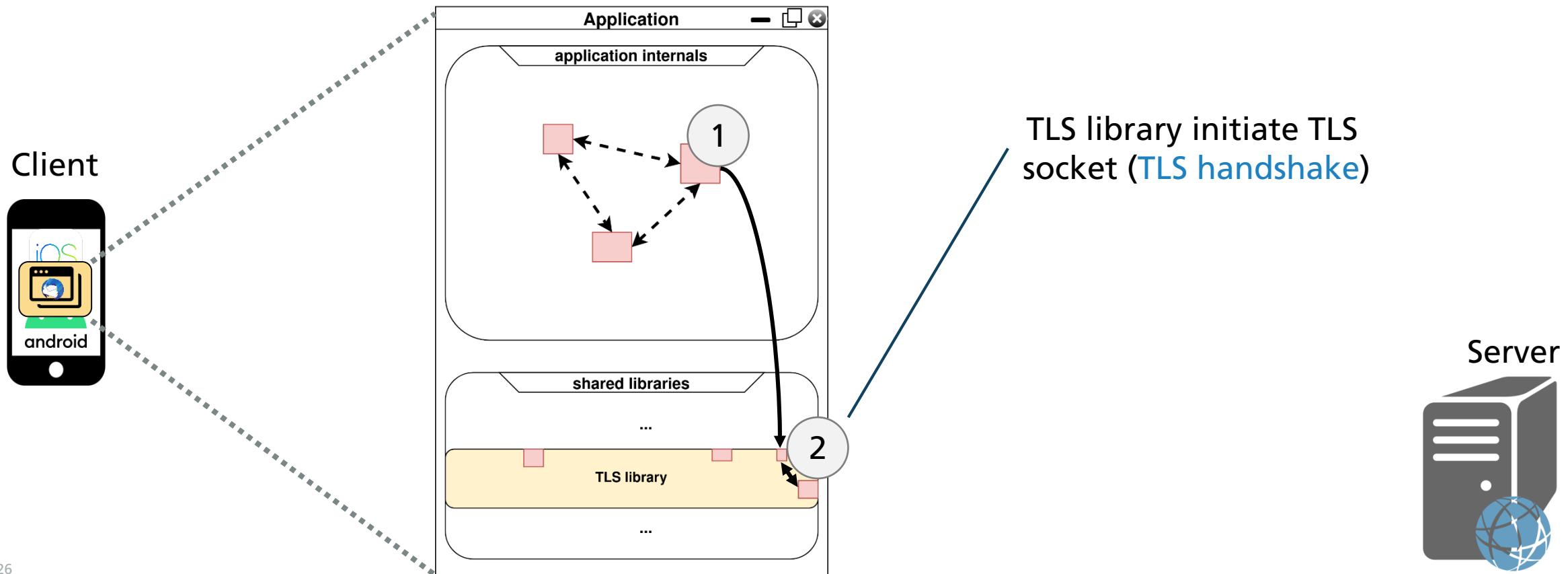


Application decides to
create TLS connection.

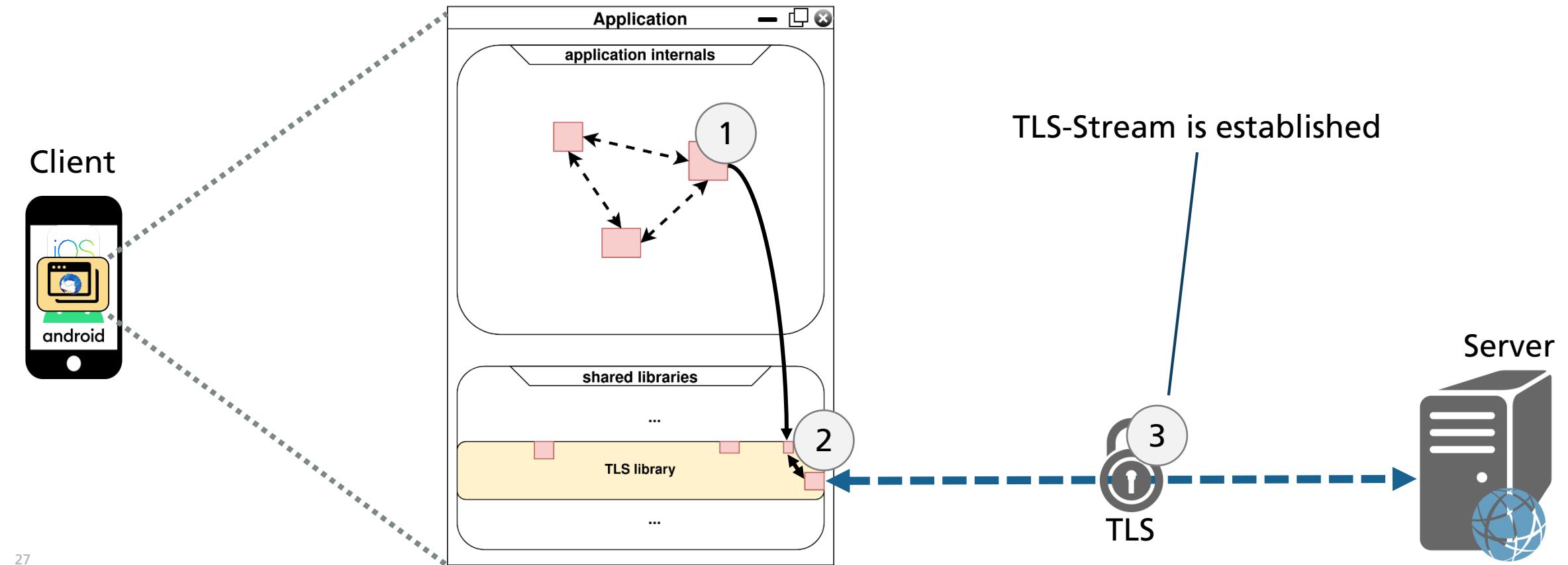
Server



friTap: Approach

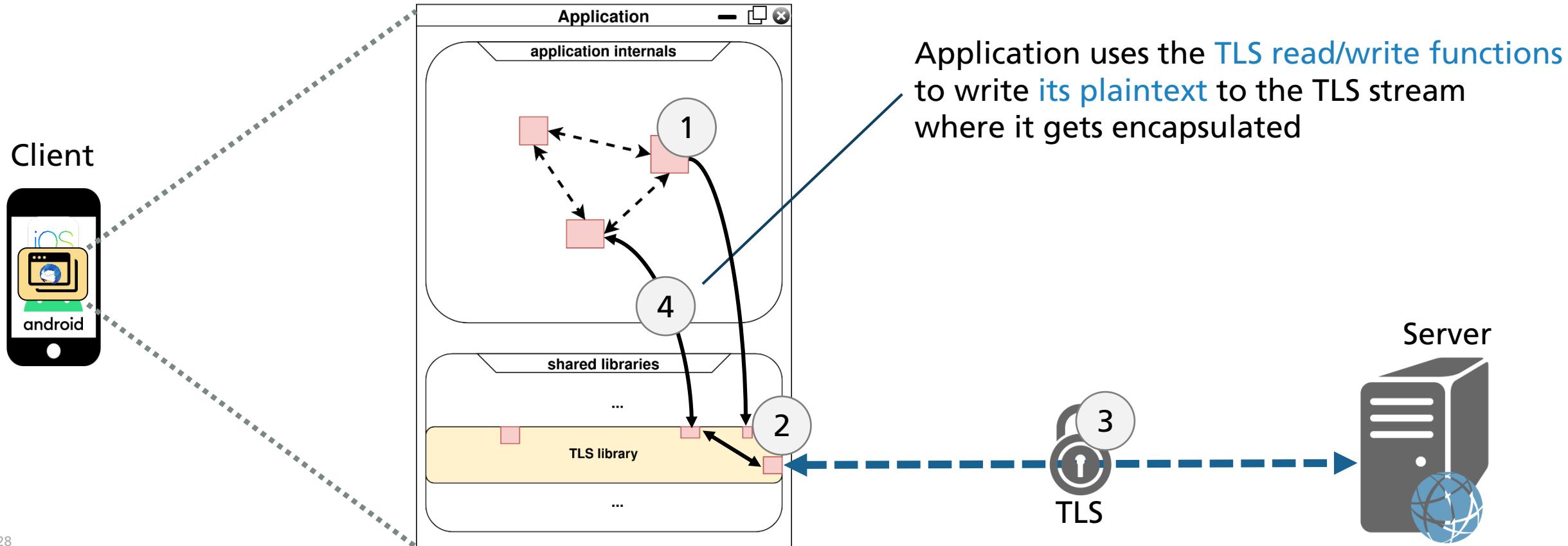


friTap: Approach



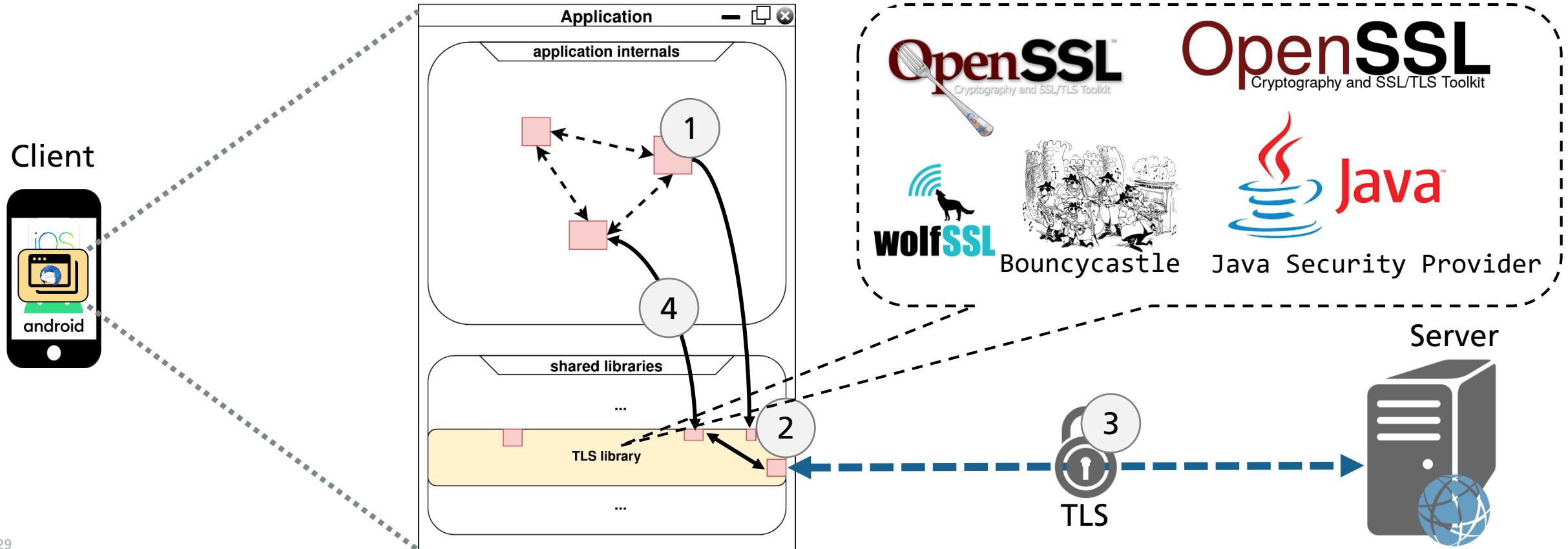
friTap: Approach

Application uses the **TLS read/write functions** to **read** and **write** the **plaintext** **from** and **to** the **TLS stream**

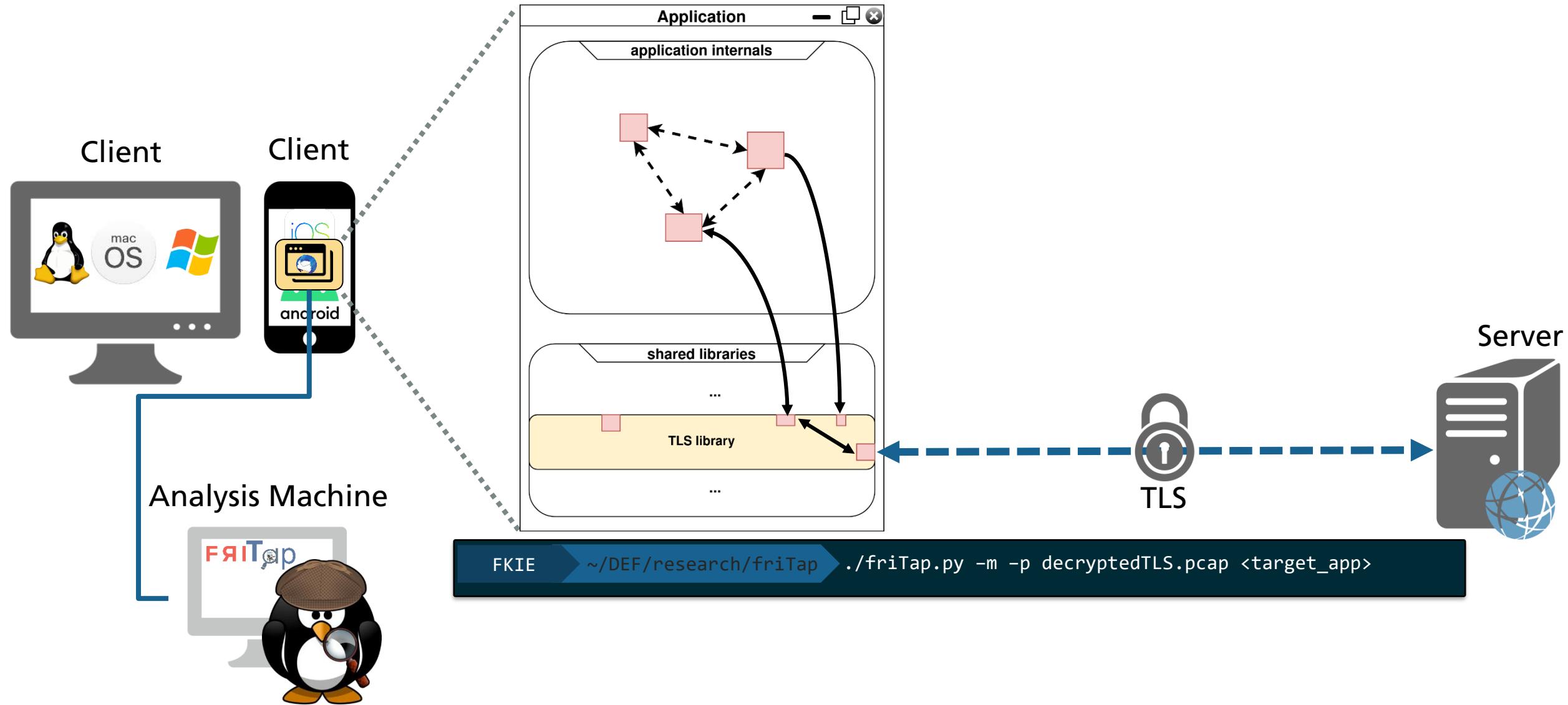


friTap: Approach

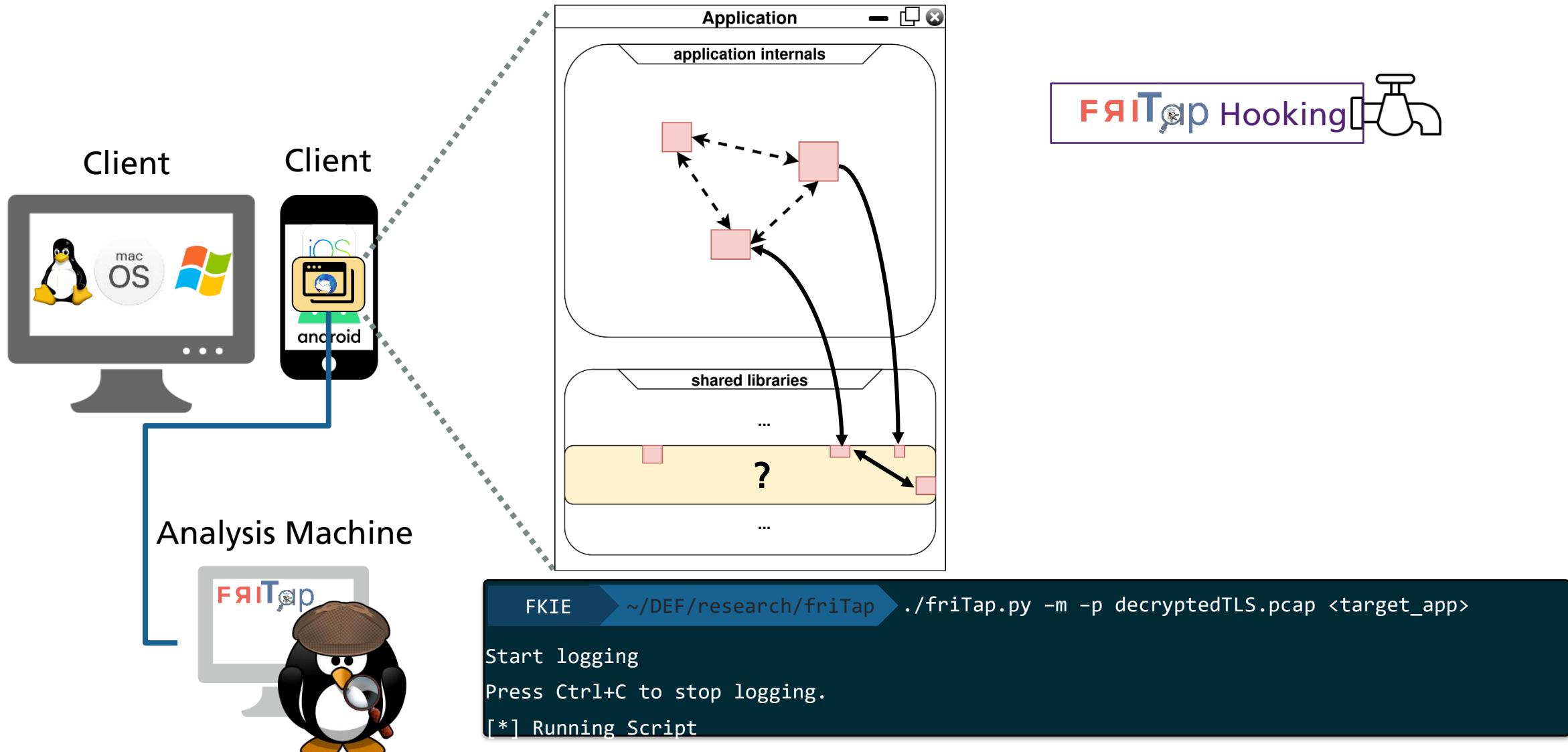
Application uses the **TLS read/write** functions to **read** and **write** the **plaintext from** and **to** the **TLS stream** and **friTap** is hooking them



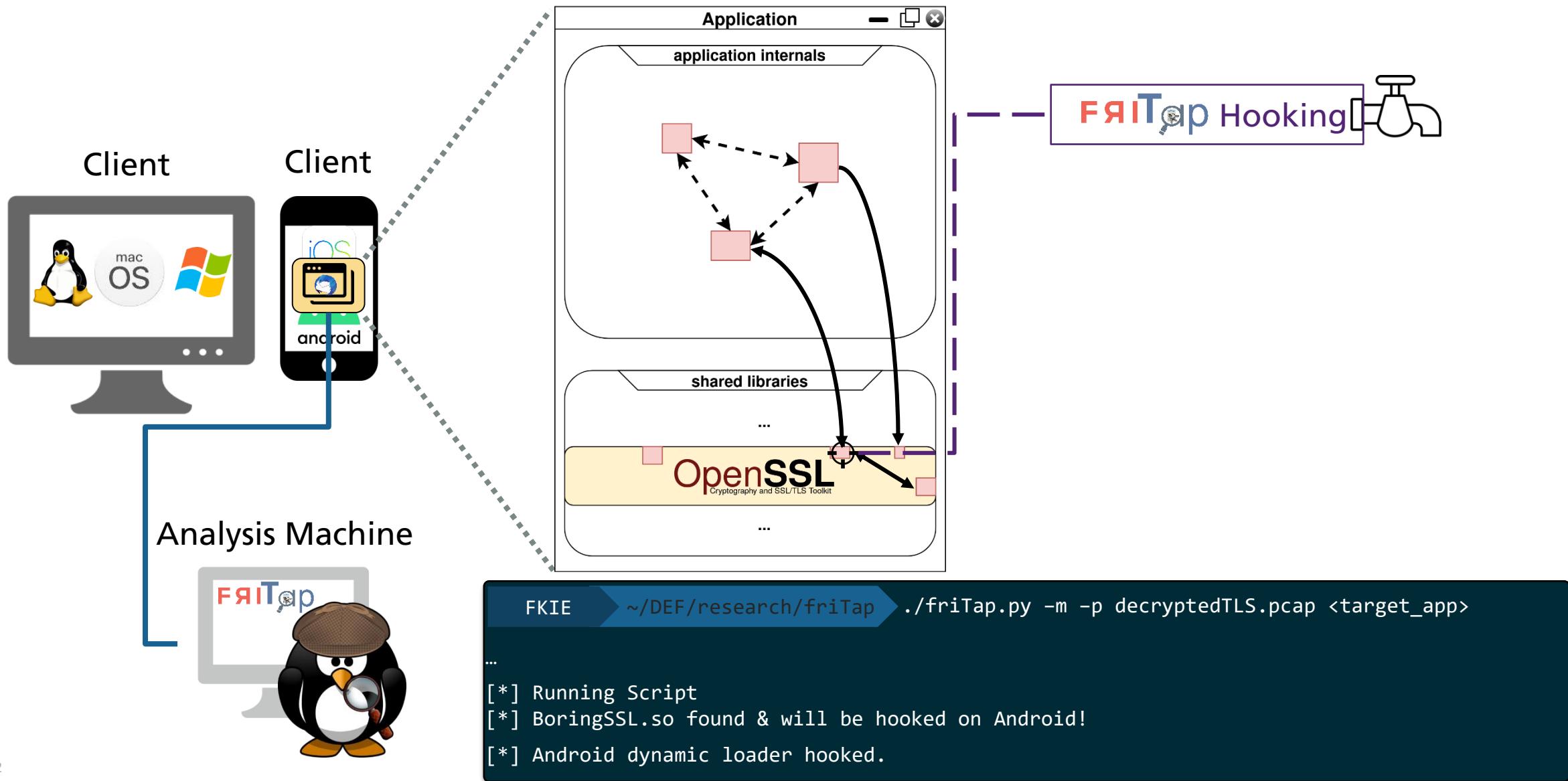
friTap: Decrypted TLS payload as PCAP



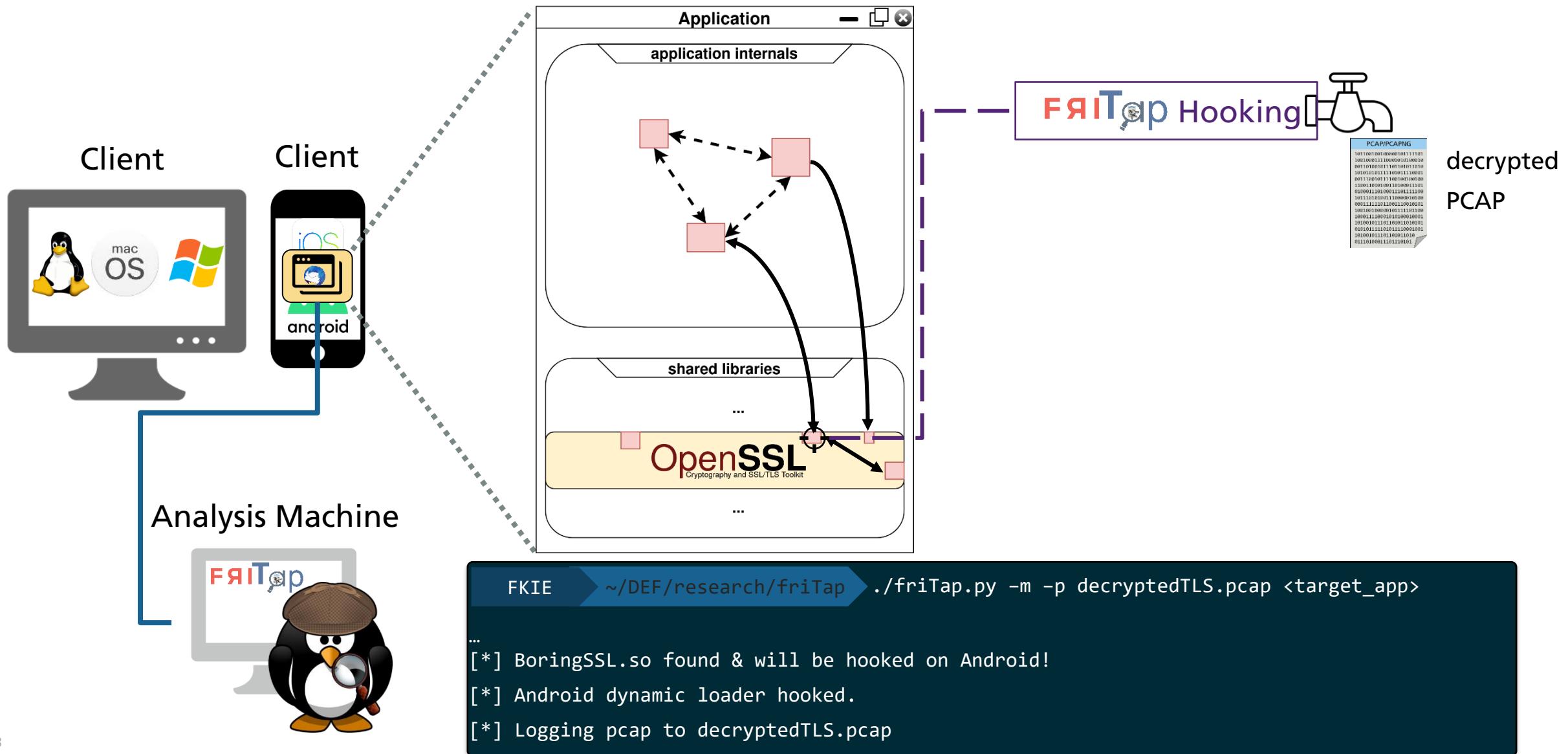
friTap: Decrypted TLS payload as PCAP



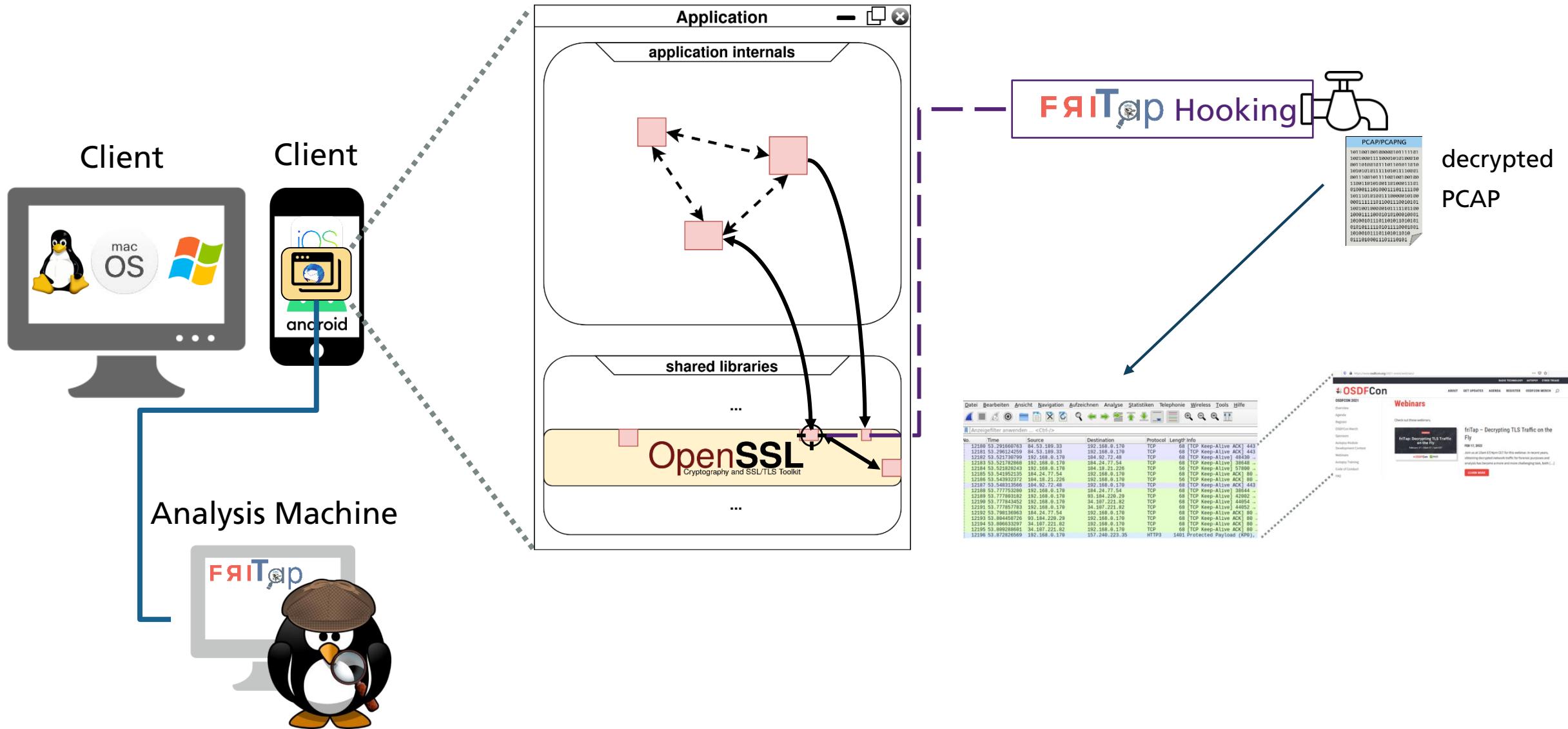
friTap: Decrypted TLS payload as PCAP



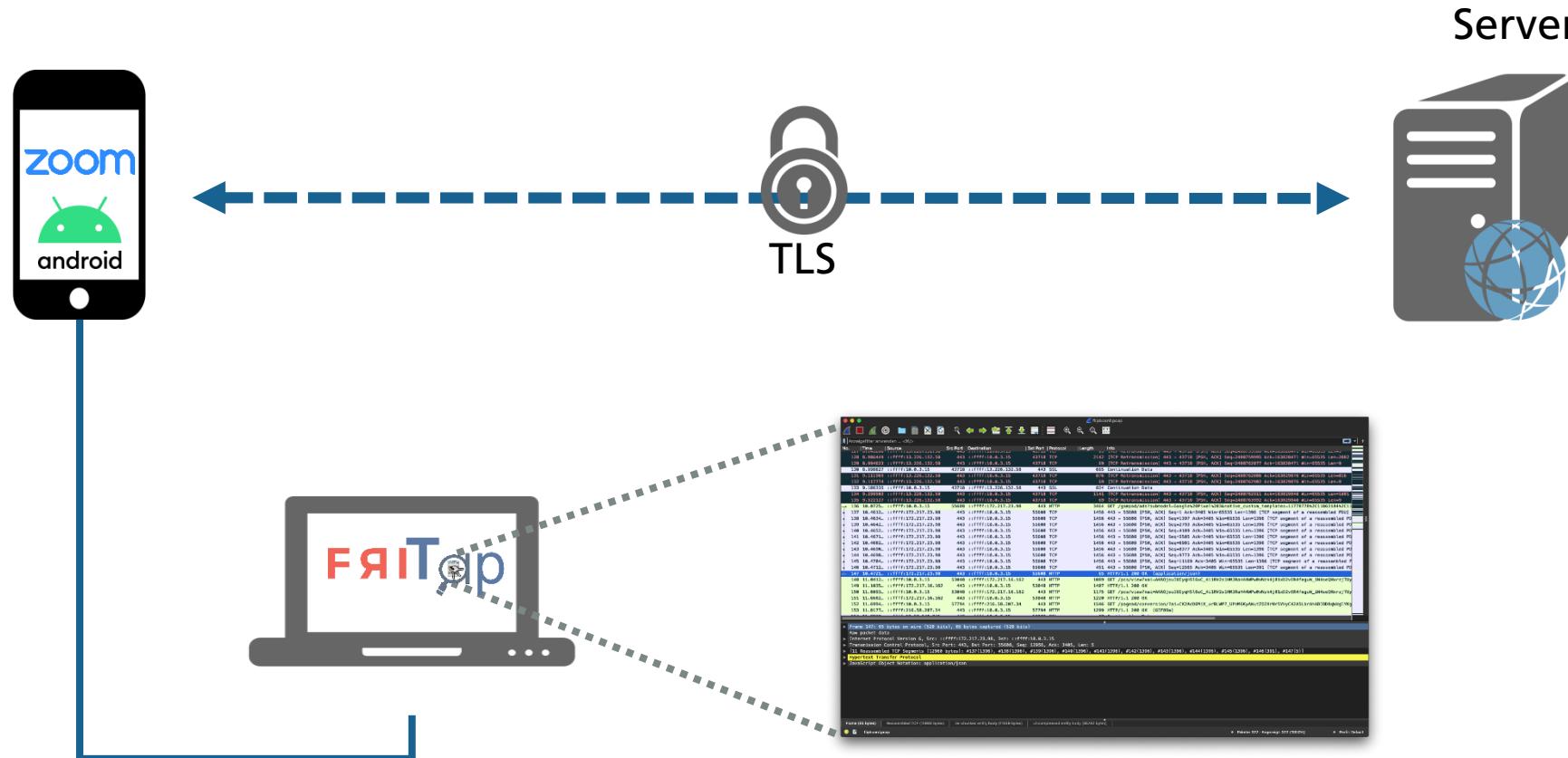
friTap: Decrypted TLS payload as PCAP



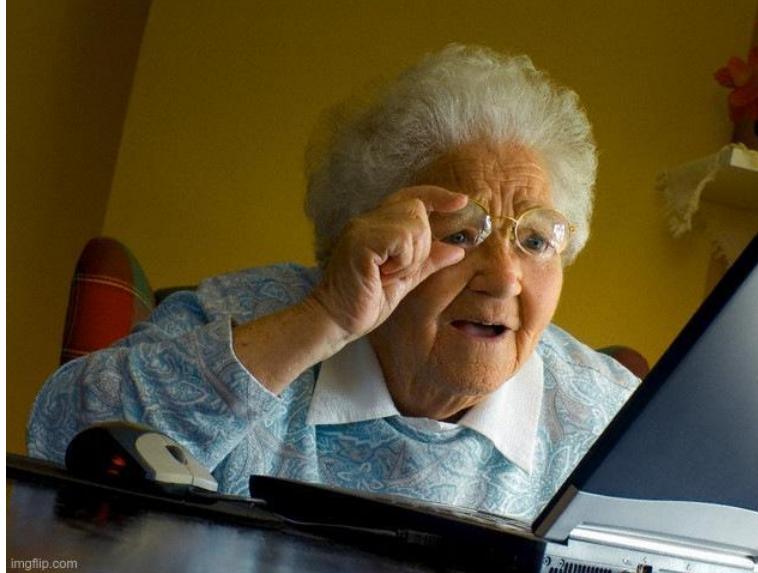
friTap: Decrypted TLS payload as PCAP



Demo: *zoom



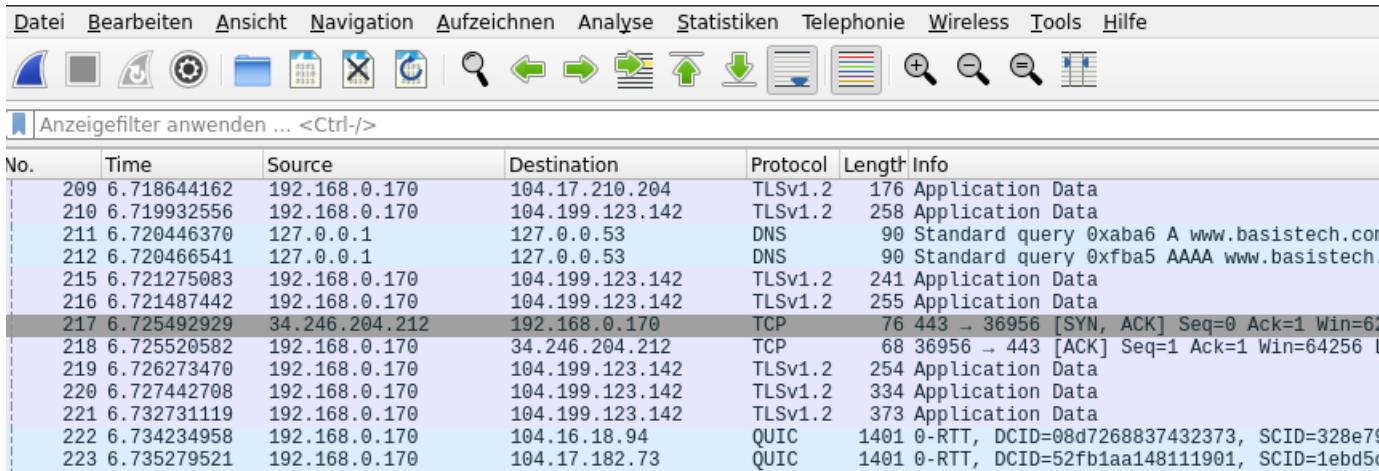
friTap: What else



Hooking functions of TLS libraries to extract TLS keys

THE [SSLKEYLOG-file](#) can also be logged. SSL libraries offers ways to [write the used keys](#) into a [file](#) which can later on be used by Wireshark to decrypt the captured traffic.

friTap: What else



The screenshot shows the friTap software interface. The menu bar includes: Datei, Bearbeiten, Ansicht, Navigation, Aufzeichnen, Analyse, Statistiken, Telefonie, Wireless, Tools, and Hilfe. Below the menu is a toolbar with icons for file operations like Open, Save, Print, and a search function. A search bar below the toolbar contains the text "Anzeigefilter anwenden ... <Ctrl-/>". The main window displays a table of network traffic. The columns are: No., Time, Source, Destination, Protocol, Length, and Info. The table lists 223 entries, with row 217 highlighted in dark grey. Row 217 details a TCP connection from 34.246.204.212 to 192.168.0.170, showing a SYN-ACK packet with sequence number 0, acknowledgement number 1, and window size 62.

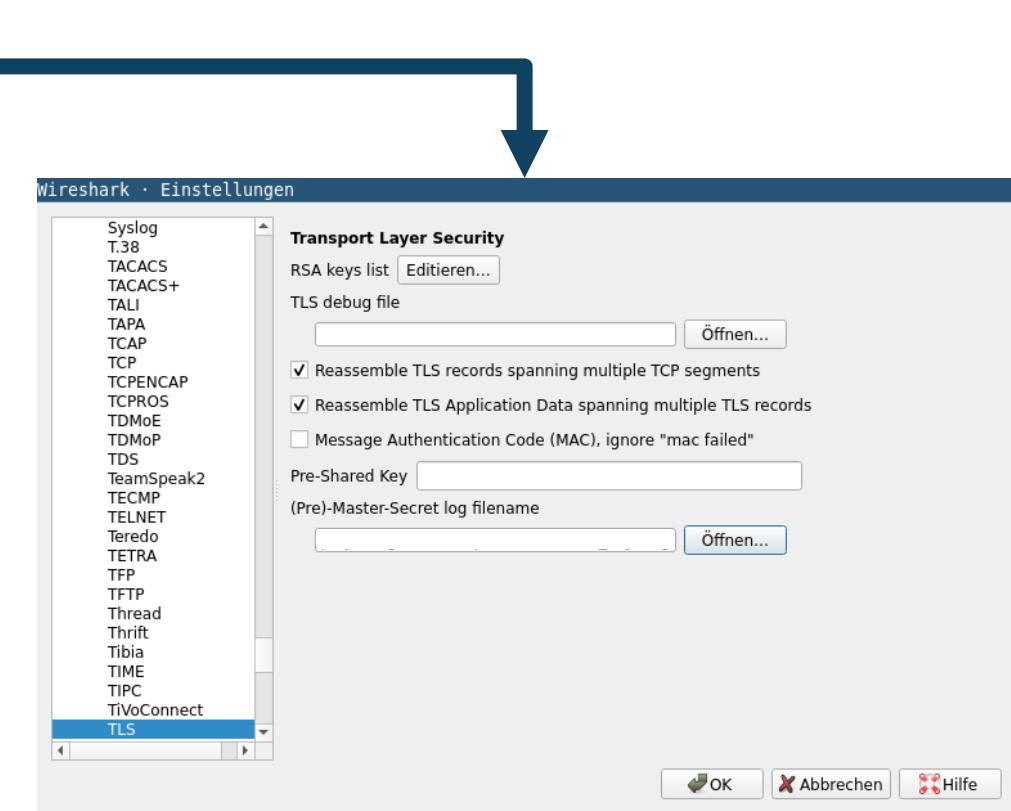
No.	Time	Source	Destination	Protocol	Length	Info
209	6.718644162	192.168.0.170	104.17.210.204	TLSv1.2	176	Application Data
210	6.719932556	192.168.0.170	104.199.123.142	TLSv1.2	258	Application Data
211	6.720446370	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xaba6 A www.basistech.com
212	6.720466541	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xfba5 AAAA www.basistech.com
215	6.721275083	192.168.0.170	104.199.123.142	TLSv1.2	241	Application Data
216	6.721487442	192.168.0.170	104.199.123.142	TLSv1.2	255	Application Data
217	6.725492929	34.246.204.212	192.168.0.170	TCP	76	443 → 36956 [SYN, ACK] Seq=0 Ack=1 Win=62
218	6.725520582	192.168.0.170	34.246.204.212	TCP	68	36956 → 443 [ACK] Seq=1 Ack=1 Win=64256 L
219	6.726273470	192.168.0.170	104.199.123.142	TLSv1.2	254	Application Data
220	6.727442708	192.168.0.170	104.199.123.142	TLSv1.2	334	Application Data
221	6.732731119	192.168.0.170	104.199.123.142	TLSv1.2	373	Application Data
222	6.734234958	192.168.0.170	104.16.18.94	QUIC	1401	0-RTT, DCID=08d7268837432373, SCID=328e79
223	6.735279521	192.168.0.170	104.17.182.73	QUIC	1401	0-RTT, DCID=52fb1aa148111901, SCID=1ebd5c

friTap: What else

Datei Bearbeiten Ansicht Navigation Aufzeichnen Analyse Statistiken Telefonie Wireless Tools Hilfe

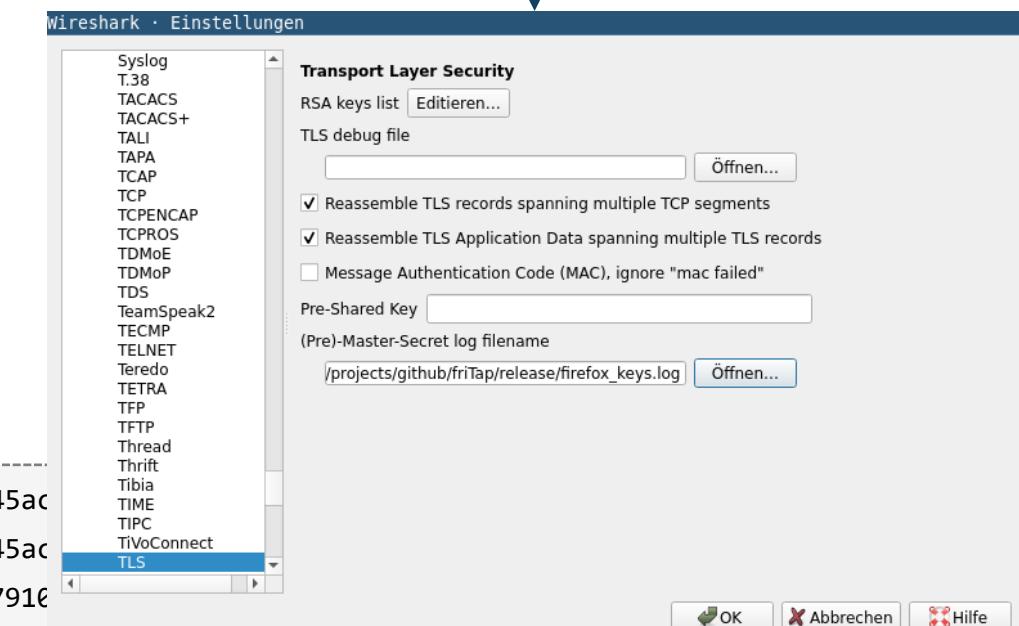
Anzeigefilter anwenden ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
209	6.718644162	192.168.0.170	104.17.210.204	TLSv1.2	176	Application Data
210	6.719932556	192.168.0.170	104.199.123.142	TLSv1.2	258	Application Data
211	6.720446370	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xaba6 A www.basistech.com
212	6.720466541	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xfba5 AAAA www.basistech.com
215	6.721275083	192.168.0.170	104.199.123.142	TLSv1.2	241	Application Data
216	6.721487442	192.168.0.170	104.199.123.142	TLSv1.2	255	Application Data
217	6.725492929	34.246.204.212	192.168.0.170	TCP	76	443 → 36956 [SYN, ACK] Seq=0 Ack=1 Win=62
218	6.725520582	192.168.0.170	34.246.204.212	TCP	68	36956 → 443 [ACK] Seq=1 Ack=1 Win=64256 L
219	6.726273470	192.168.0.170	104.199.123.142	TLSv1.2	254	Application Data
220	6.727442708	192.168.0.170	104.199.123.142	TLSv1.2	334	Application Data
221	6.732731119	192.168.0.170	104.199.123.142	TLSv1.2	373	Application Data
222	6.734234958	192.168.0.170	104.16.18.94	QUIC	1401	0-RTT, DCID=08d7268837432373, SCID=328e79
223	6.735279521	192.168.0.170	104.17.182.73	QUIC	1401	0-RTT, DCID=52fb1aa148111901, SCID=1ebd5c



friTap: What else

No.	Time	Source	Destination	Protocol	Length	Info
209	6.718644162	192.168.0.170	104.17.210.204	TLSv1.2	176	Application Data
210	6.719932556	192.168.0.170	104.199.123.142	TLSv1.2	258	Application Data
211	6.720446370	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xaba6 A www.basistech.com
212	6.720466541	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xfba5 AAAA www.basistech.com
215	6.721275083	192.168.0.170	104.199.123.142	TLSv1.2	241	Application Data
216	6.721487442	192.168.0.170	104.199.123.142	TLSv1.2	255	Application Data
217	6.725492929	34.246.204.212	192.168.0.170	TCP	76	443 → 36956 [SYN, ACK] Seq=0 Ack=1 Win=62
218	6.725520582	192.168.0.170	34.246.204.212	TCP	68	36956 → 443 [ACK] Seq=1 Ack=1 Win=64256 L
219	6.726273470	192.168.0.170	104.199.123.142	TLSv1.2	254	Application Data
220	6.727442708	192.168.0.170	104.199.123.142	TLSv1.2	334	Application Data
221	6.732731119	192.168.0.170	104.199.123.142	TLSv1.2	373	Application Data
222	6.734234958	192.168.0.170	104.16.18.94	QUIC	1401	0-RTT, DCID=08d7268837432373, SCID=328e79
223	6.735279521	192.168.0.170	104.17.182.73	QUIC	1401	0-RTT, DCID=52fb1aa148111901, SCID=1ebd5c



SSLKEYLOG file extracted by friTap

```

CLIENT_HANDSHAKE_TRAFFIC_SECRET 679e8c731884f1c36200e338816b52f86be6a3699ed83805b3445ac
SERVER_HANDSHAKE_TRAFFIC_SECRET 679e8c731884f1c36200e338816b52f86be6a3699ed83805b3445ac
CLIENT_TRAFFIC_SECRET_0 679e8c731884f1c36200e338816b52f86be6a3699ed83805b3445acd0707910
SERVER_TRAFFIC_SECRET_0 679e8c731884f1c36200e338816b52f86be6a3699ed83805b3445acd0707910
EXPORTER_SECRET 679e8c731884f1c36200e338816b52f86be6a3699ed83805b3445acd0707910c 658e36ef804d07017d798d67cf6157c168604892e66a1a0d640501...
CLIENT_RANDOM 55c984ac6aacdb1af0873a9eb891a88babf2fe1f35b50eb7e93a1515f9db0b36 6a7ecc21ea11f8cf4a7954def08b965bb53903d37e9b95a9ee6dbd271...
...

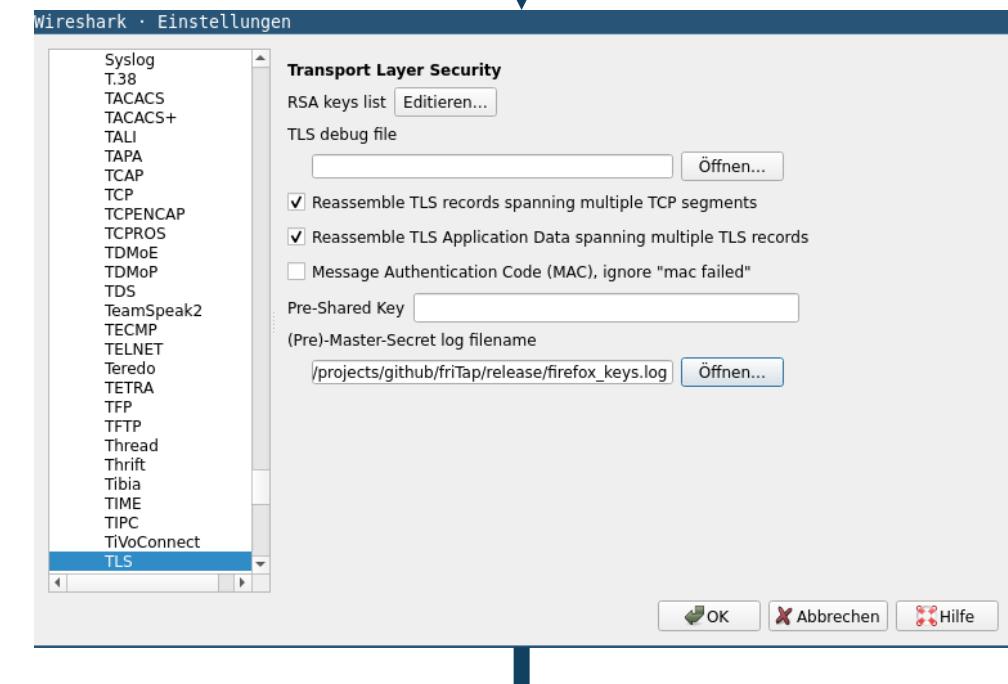
```

friTap: What else

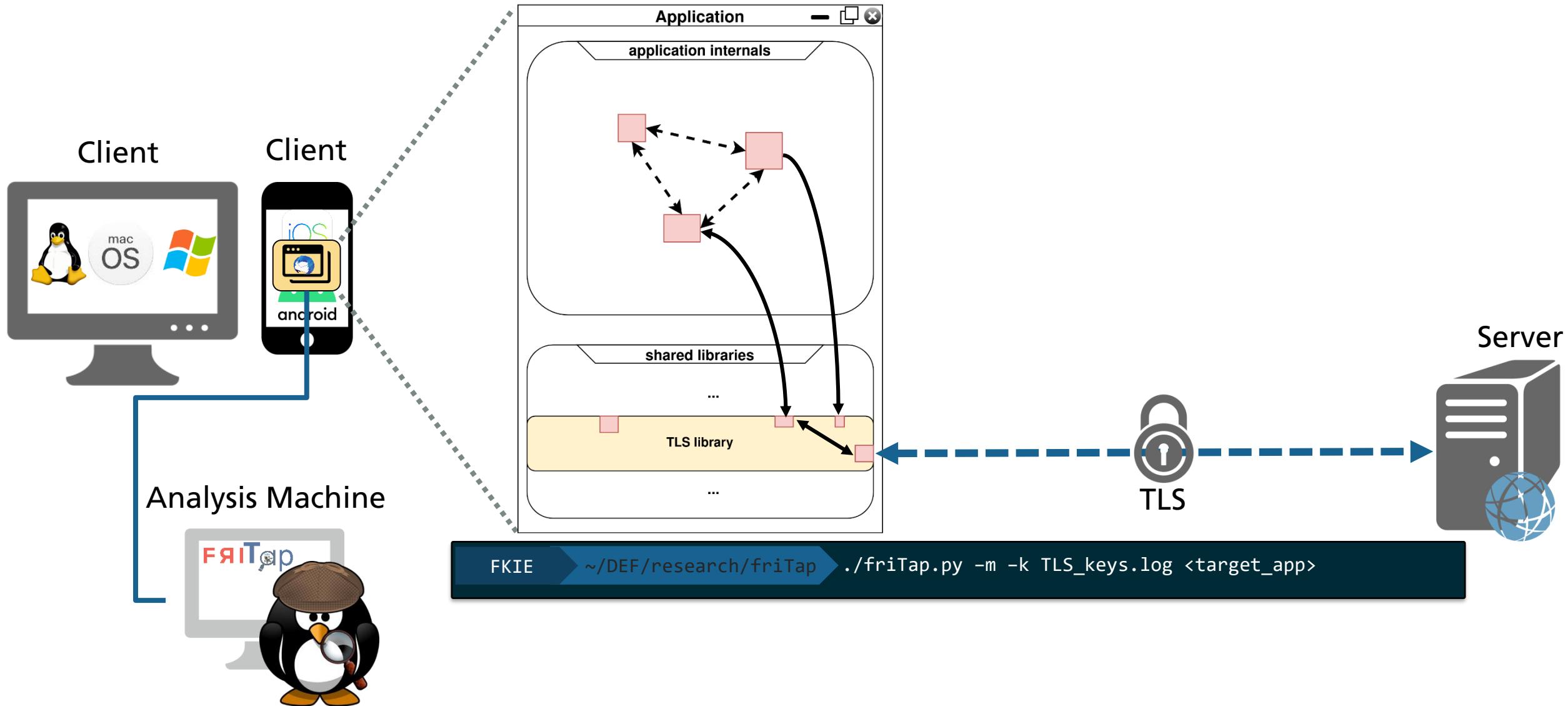
No.	Time	Source	Destination	Protocol	Length	Info
209	6.718644162	192.168.0.170	104.17.210.204	TLSv1.2	176	Application Data
210	6.719932556	192.168.0.170	104.199.123.142	TLSv1.2	258	Application Data
211	6.720446370	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xaba6 A www.basistech.com
212	6.720466541	127.0.0.1	127.0.0.53	DNS	90	Standard query 0xfba5 AAAA www.basistech.com
215	6.721275083	192.168.0.170	104.199.123.142	TLSv1.2	241	Application Data
216	6.721487442	192.168.0.170	104.199.123.142	TLSv1.2	255	Application Data
217	6.725492929	34.246.204.212	192.168.0.170	TCP	76	443 → 36956 [SYN, ACK] Seq=0 Ack=1 Win=62
218	6.725520582	192.168.0.170	34.246.204.212	TCP	68	36956 → 443 [ACK] Seq=1 Ack=1 Win=64256 L
219	6.726273470	192.168.0.170	104.199.123.142	TLSv1.2	254	Application Data
220	6.727442708	192.168.0.170	104.199.123.142	TLSv1.2	334	Application Data
221	6.732731119	192.168.0.170	104.199.123.142	TLSv1.2	373	Application Data
222	6.734234958	192.168.0.170	104.16.18.94	QUIC	1401	0-RTT, DCID=08d7268837432373, SCID=328e79
223	6.735279521	192.168.0.170	104.17.182.73	QUIC	1401	0-RTT, DCID=52fb1aa148111901, SCID=1ebd5c



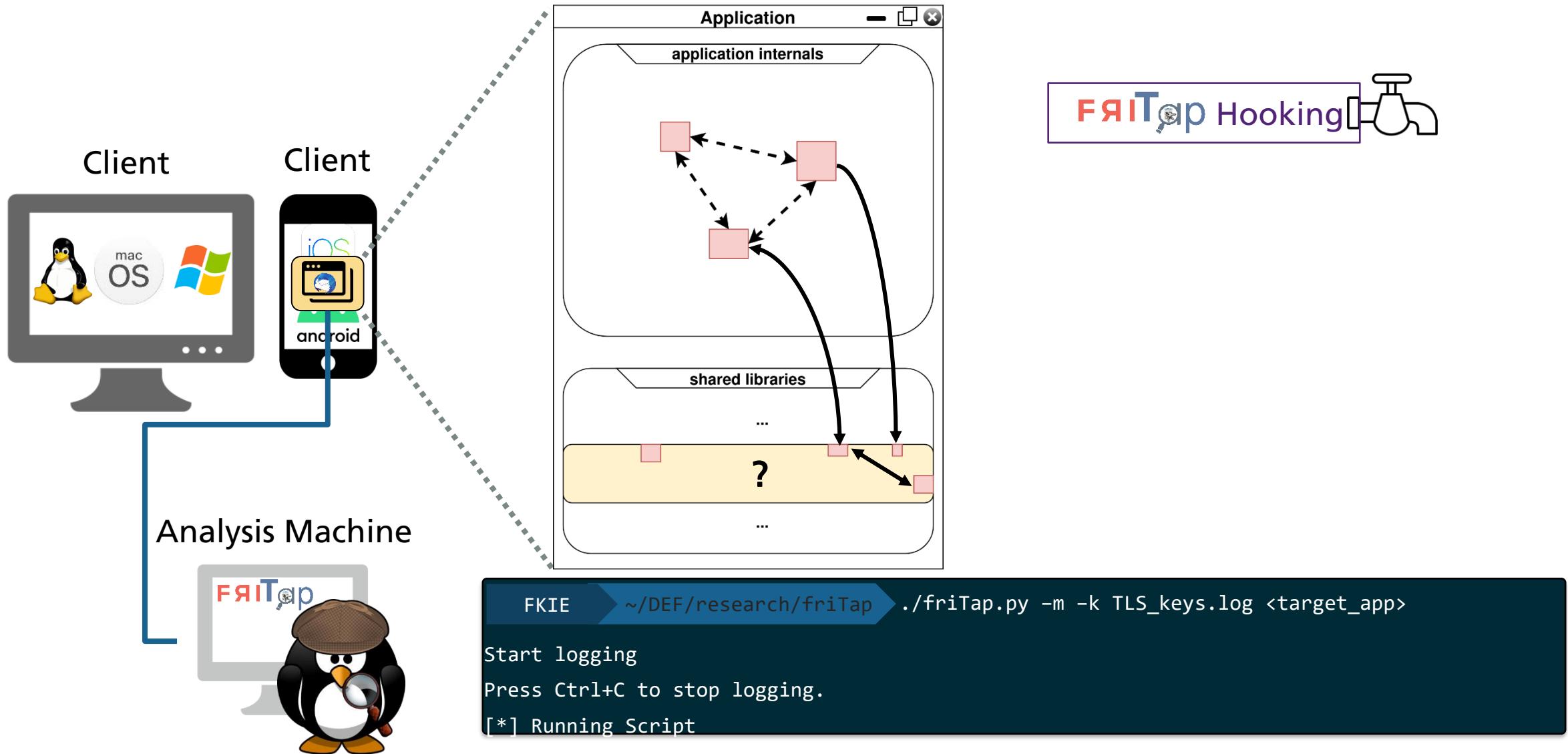
No.	Time	Source	Destination	Protocol	Length	Info
12180	53.291660763	84.53.189.33	192.168.0.170	TCP	68	[TCP Keep-Alive ACK] 443
12181	53.296124259	84.53.189.33	192.168.0.170	TCP	68	[TCP Keep-Alive ACK] 443
12182	53.521730799	192.168.0.170	104.92.72.48	TCP	68	[TCP Keep-Alive] 48430 →
12183	53.521782868	192.168.0.170	184.24.77.54	TCP	68	[TCP Keep-Alive] 38648 →
12184	53.521828243	192.168.0.170	104.18.21.226	TCP	56	[TCP Keep-Alive] 57800 →
12185	53.541952135	184.24.77.54	192.168.0.170	TCP	68	[TCP Keep-Alive ACK] 80 →
12186	53.5439332372	104.18.21.226	192.168.0.170	TCP	56	[TCP Keep-Alive ACK] 80 →
12187	53.548313566	104.92.72.48	192.168.0.170	TCP	68	[TCP Keep-Alive ACK] 443
12188	53.777753280	192.168.0.170	184.24.77.54	TCP	68	[TCP Keep-Alive] 38644 →
12189	53.777803182	192.168.0.170	93.184.220.29	TCP	68	[TCP Keep-Alive] 42002 →
12190	53.777843452	192.168.0.170	34.107.221.82	TCP	68	[TCP Keep-Alive] 44054 →
12191	53.777857783	192.168.0.170	34.107.221.82	TCP	68	[TCP Keep-Alive] 44052 →
12192	53.798136963	184.24.77.54	192.168.0.170	TCP	68	[TCP Keep-Alive ACK] 80 →
12193	53.804458726	93.184.220.29	192.168.0.170	TCP	68	[TCP Keep-Alive ACK] 80 →
12194	53.806633297	34.107.221.82	192.168.0.170	TCP	68	[TCP Keep-Alive ACK] 80 →
12195	53.809288601	34.107.221.82	192.168.0.170	TCP	68	[TCP Keep-Alive ACK] 80 →
12196	53.872826569	192.168.0.170	157.240.223.35	HTTP3	1401	Protected Payload (KP0),



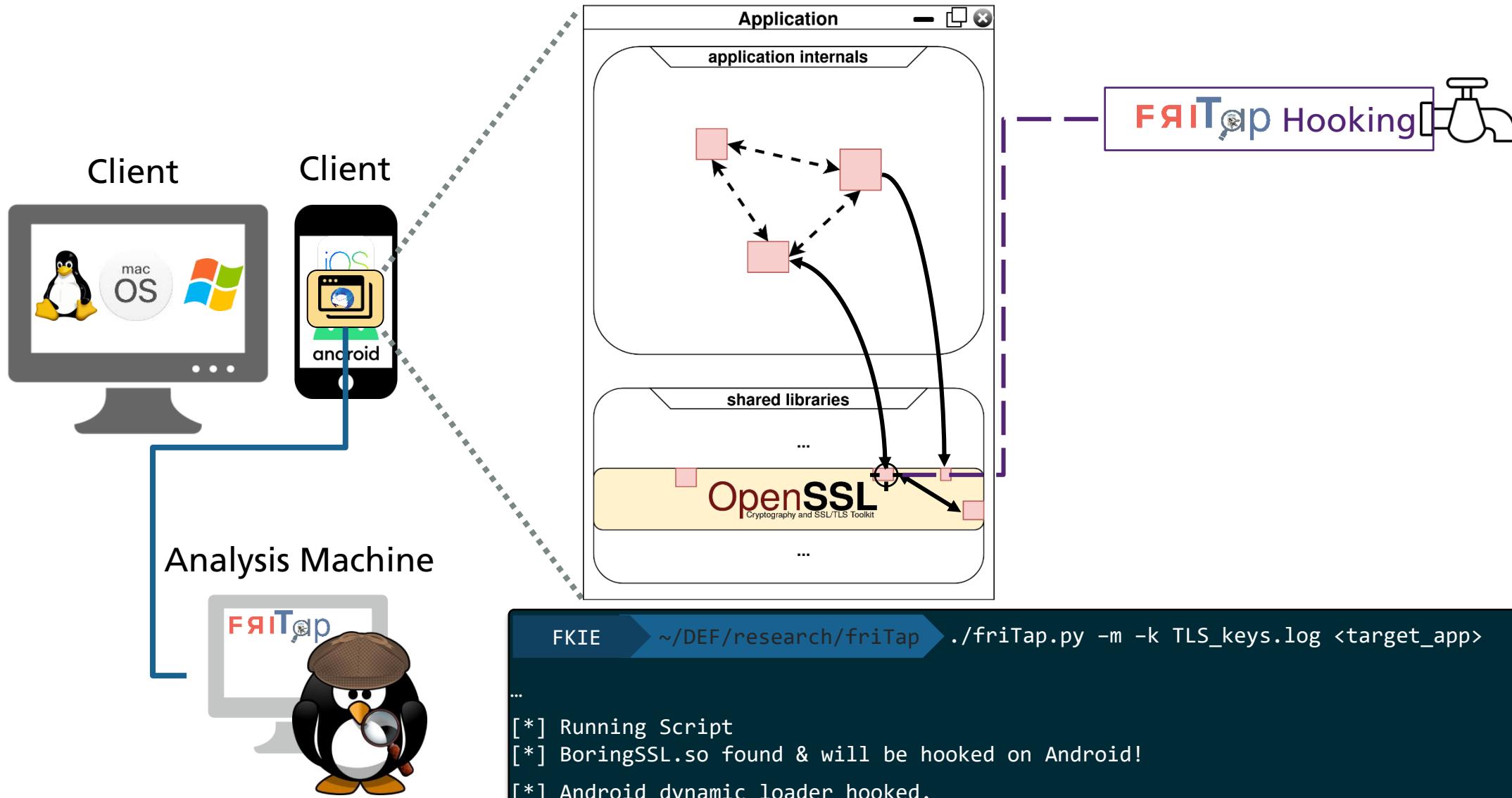
friTap: TLS key extraction



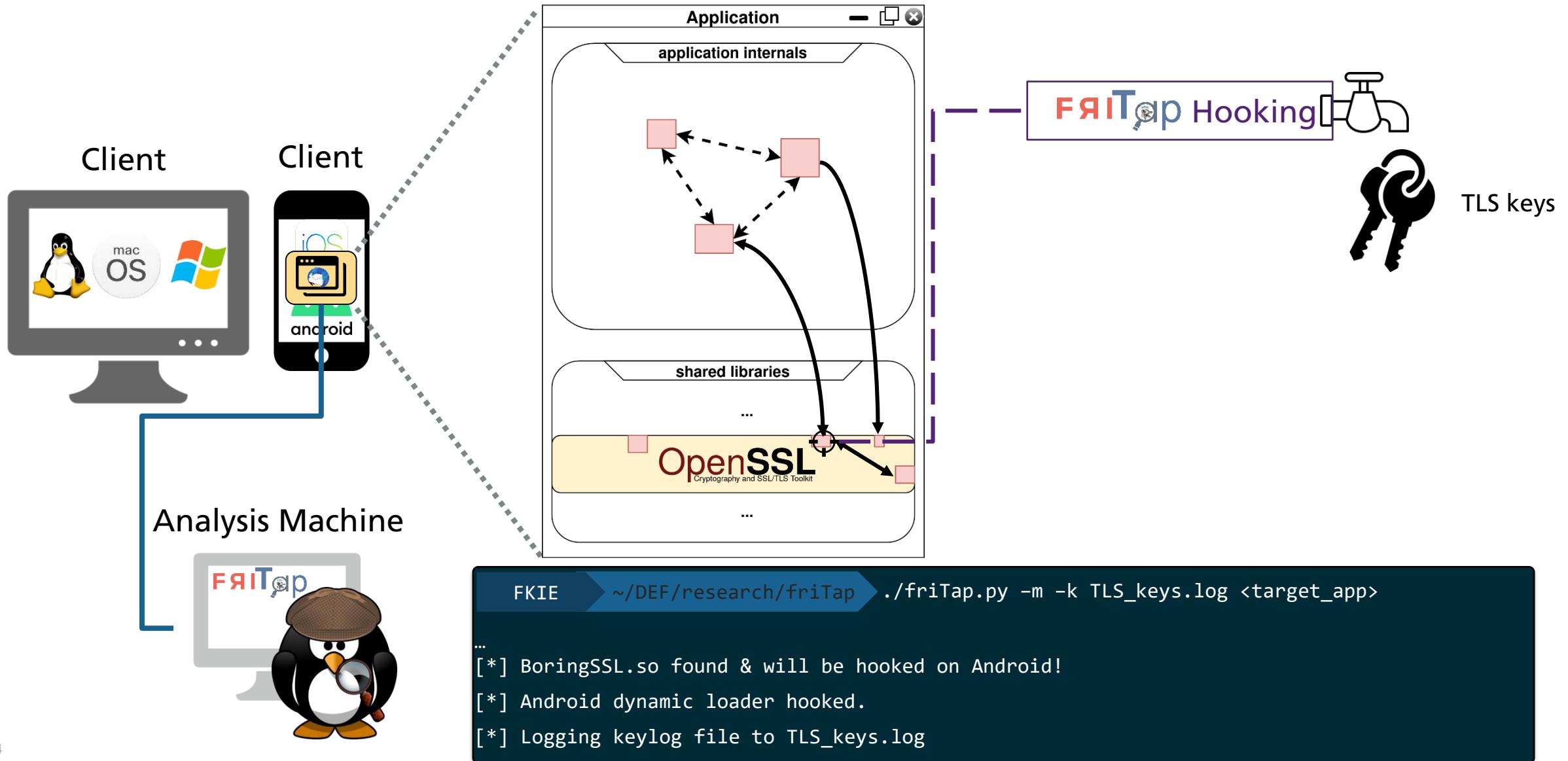
friTap: TLS key extraction



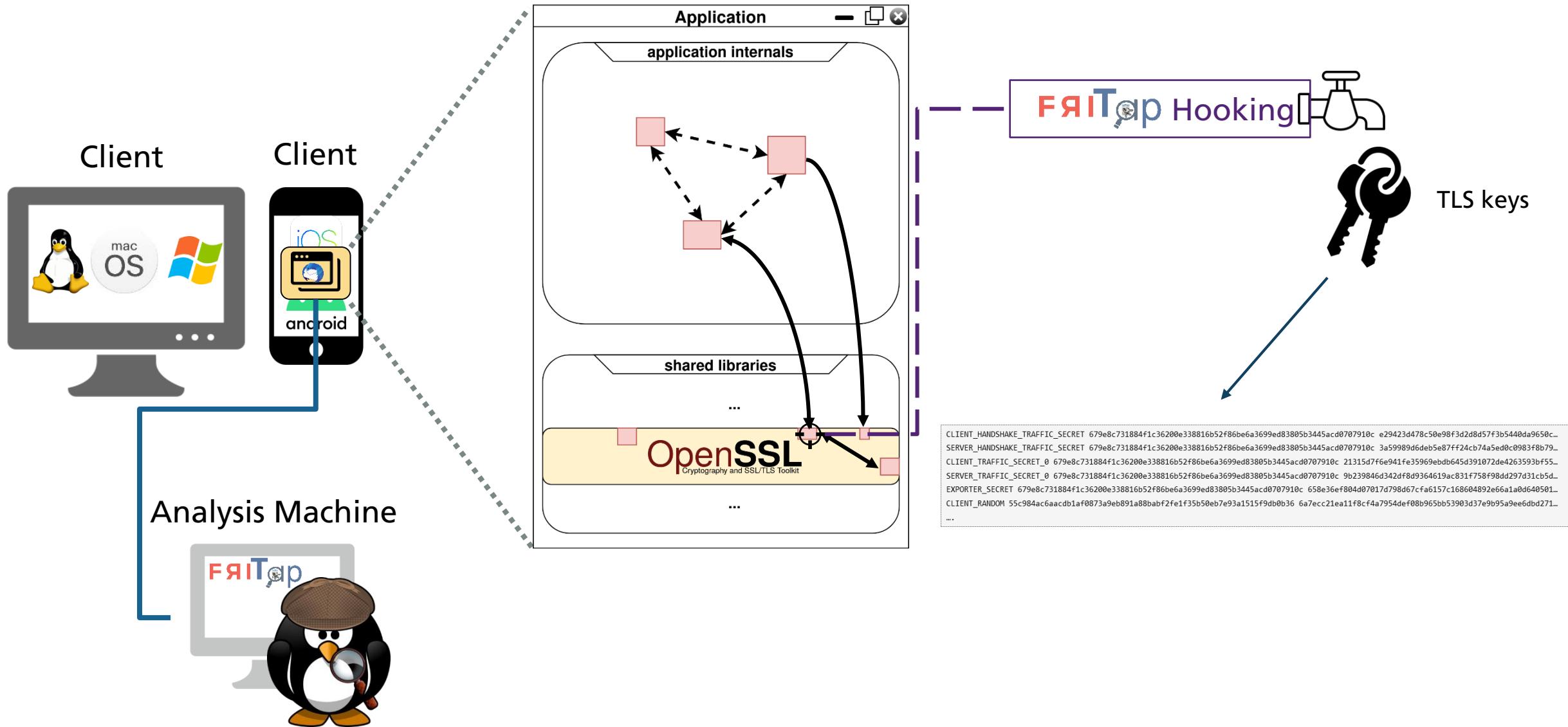
friTap: TLS key extraction



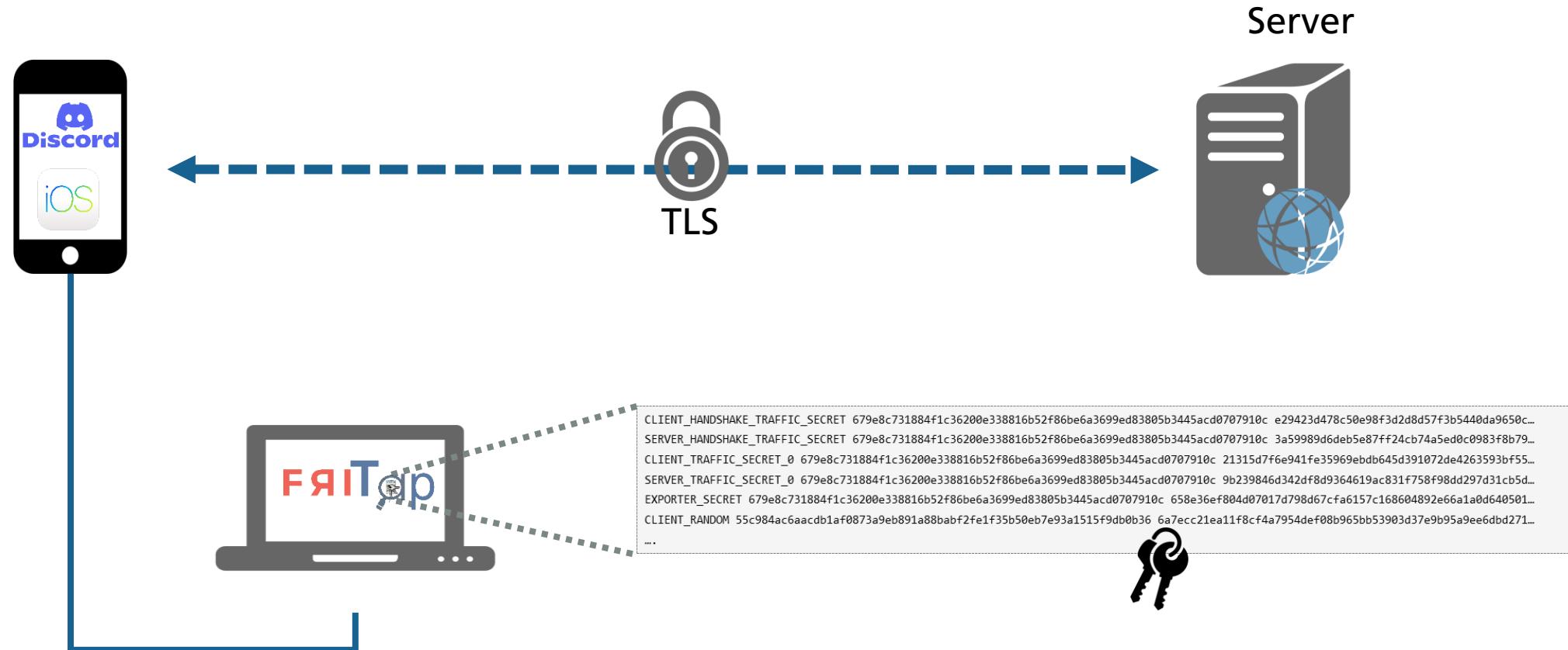
friTap: TLS key extraction



friTap: TLS key extraction

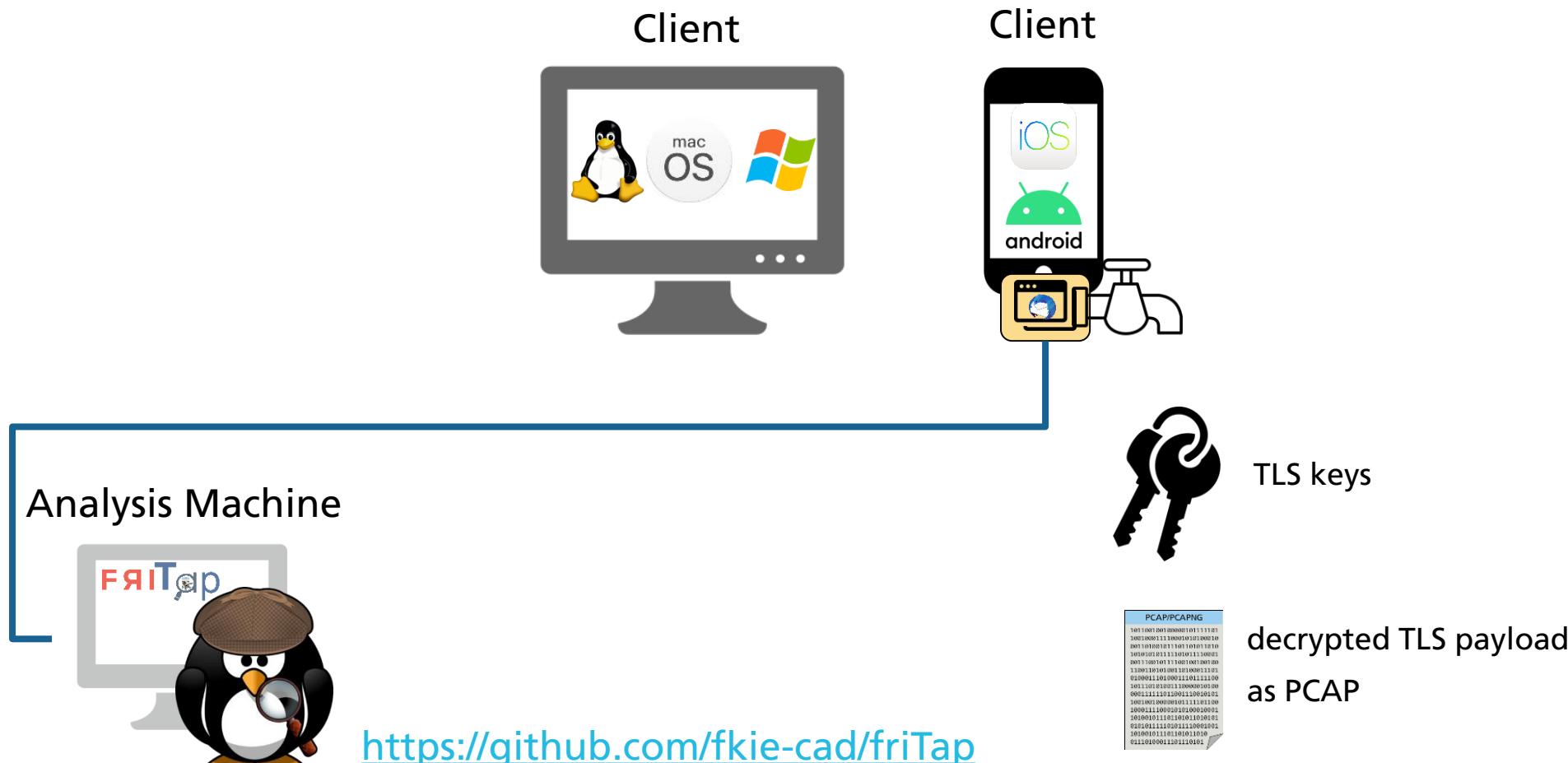


Demo: * Discord



Conclusion

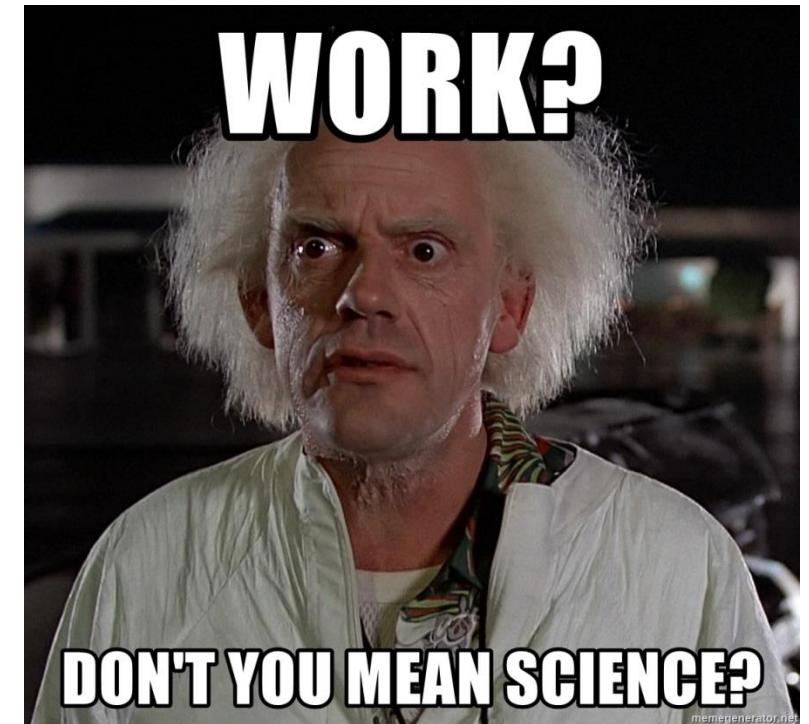
friTap provides us with TLS keys and the decrypted TLS payload



Future Work

There is still a lot of work to do....

- Support for other SSL libraries
- Add feature to prototype TLS Read/Write/SSLKEY function
- Capturing all traffic of an application directly with friTap
- Add the capability to alter the decrypted payload
 - integration with  mitmproxy
 - integration with  Burp Suite
- Working with static linked libraries
- ...



Q&A

Thanks for your time!

<https://github.com/fkie-cad/friTap>