

TLS Session Key Extraction from Memory on iOS Devices

Research Project 2

Tom Curran <tom.curran@os3.nl>

Marat Nigmatullin <marat.nigmatullin@os3.nl>



Motivation

- Increase in TLS encryption on iOS devices
- Prevents blackbox testing
- Existing tools *disable* TLS and *rely on jailbreak*
- Is there an alternative approach?

Research Question

Is it possible to extract TLS session keys from the process memory of a device running iOS 9.0 or greater?

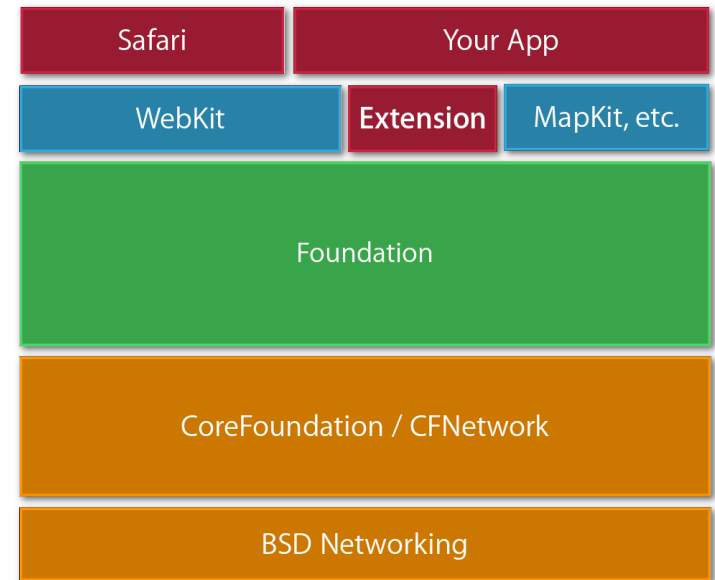
- How is TLS handled in iOS?
- Can it be done with jailbroken *and* non-jailbroken devices?

TLS

- Cryptographic protocol, successor of SSL
- Provides *confidentiality and authentication*
- Uses the *Record* protocol
- Sub-protocols
 1. Handshake
 2. ChangeCipherSpec
 3. Application Data
 4. Alert

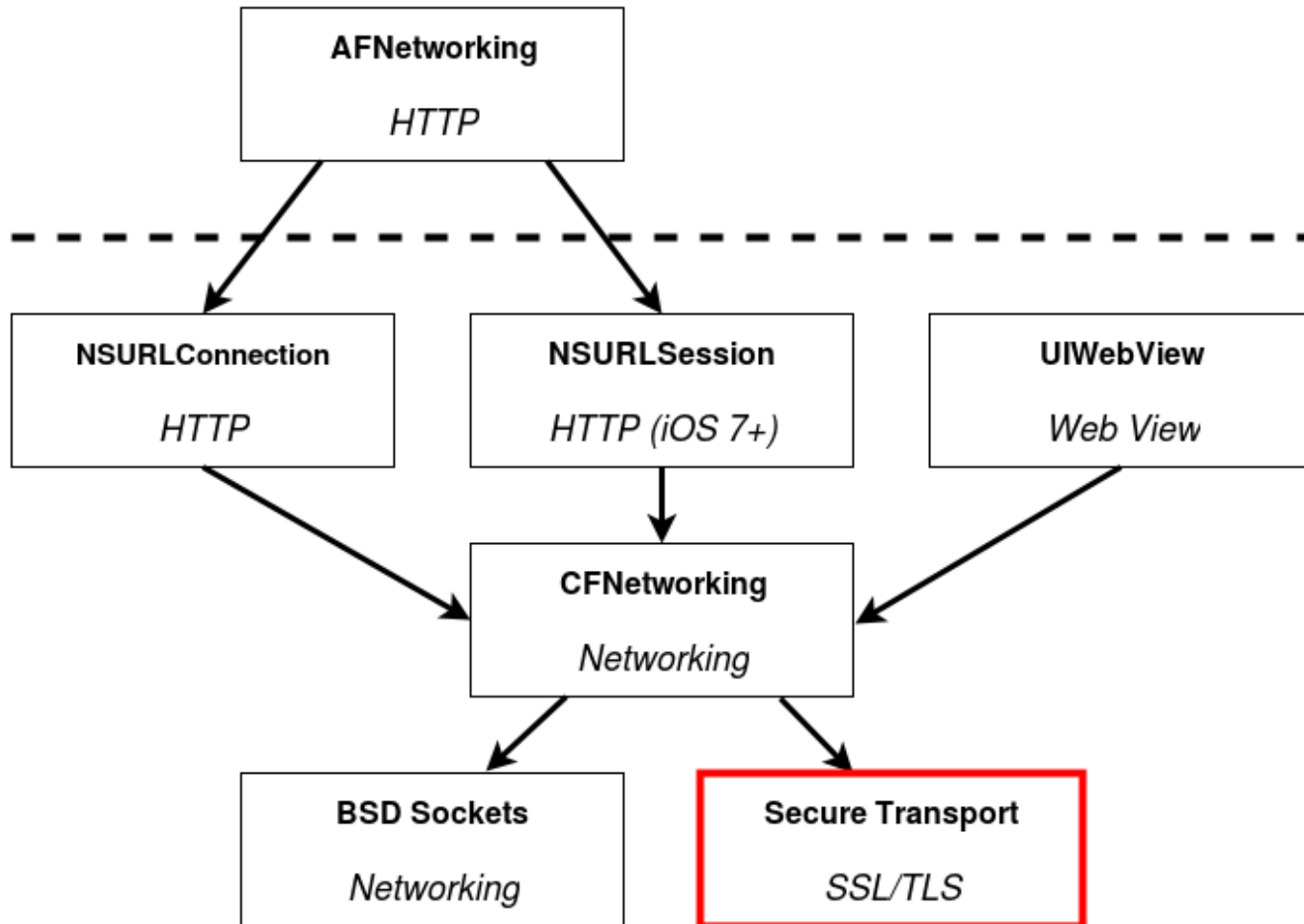
TLS in iOS

- Foundation Networking APIs
- TLS handled via *Secure Transport API*
- App Transport Security (ATS) (9.0+)
 - TLS 1.2
 - Forward secrecy
 - Key Exchange - ECDHE
 - Authentication - ECDSA or RSA
 - Mandatory on App Store from 2017



Source: Apple.com

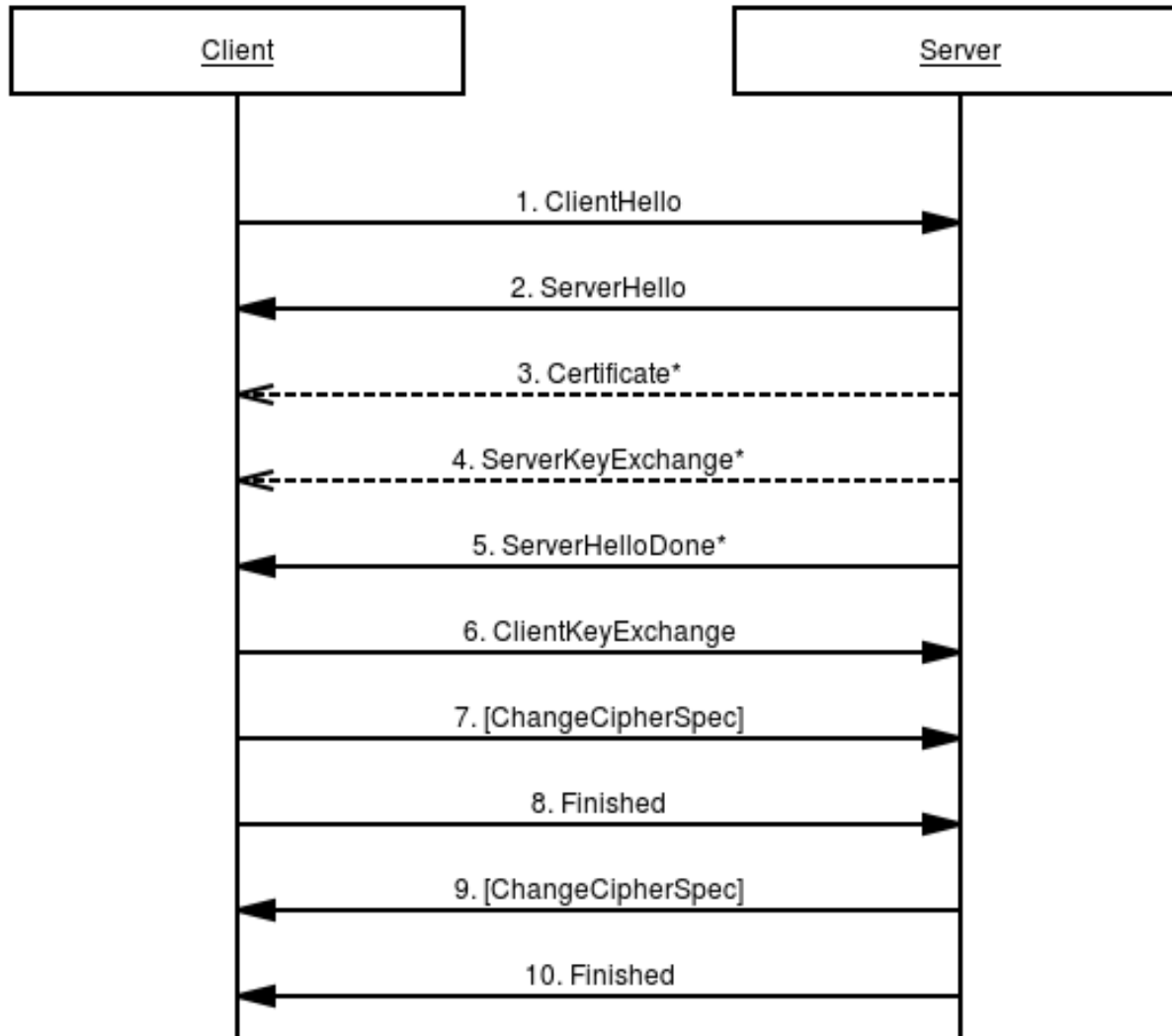
iOS Network Stack



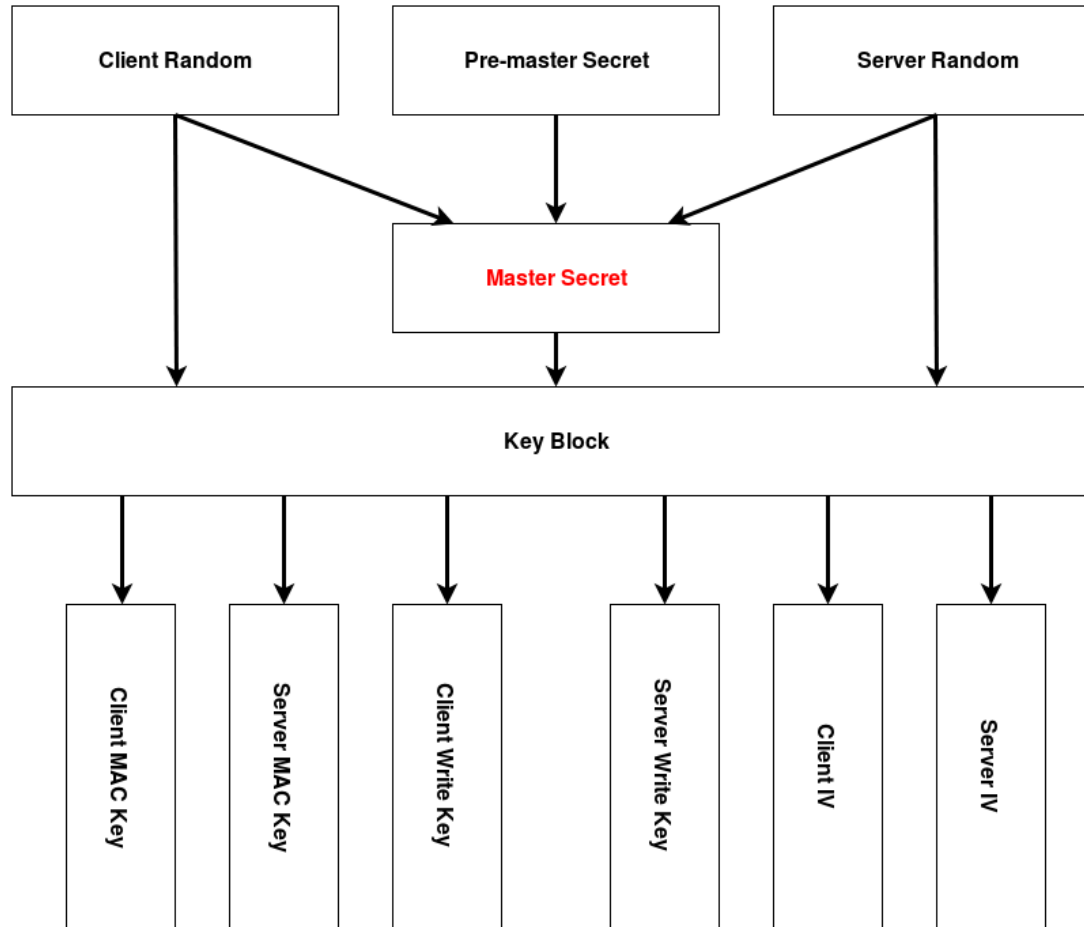
TLS Handshake

- Establish shared secret
- Four phases:
 1. Exchange capabilities and agree on connection parameters
 2. Authentication
 3. Agree on shared secret
 4. Verify handshake messages

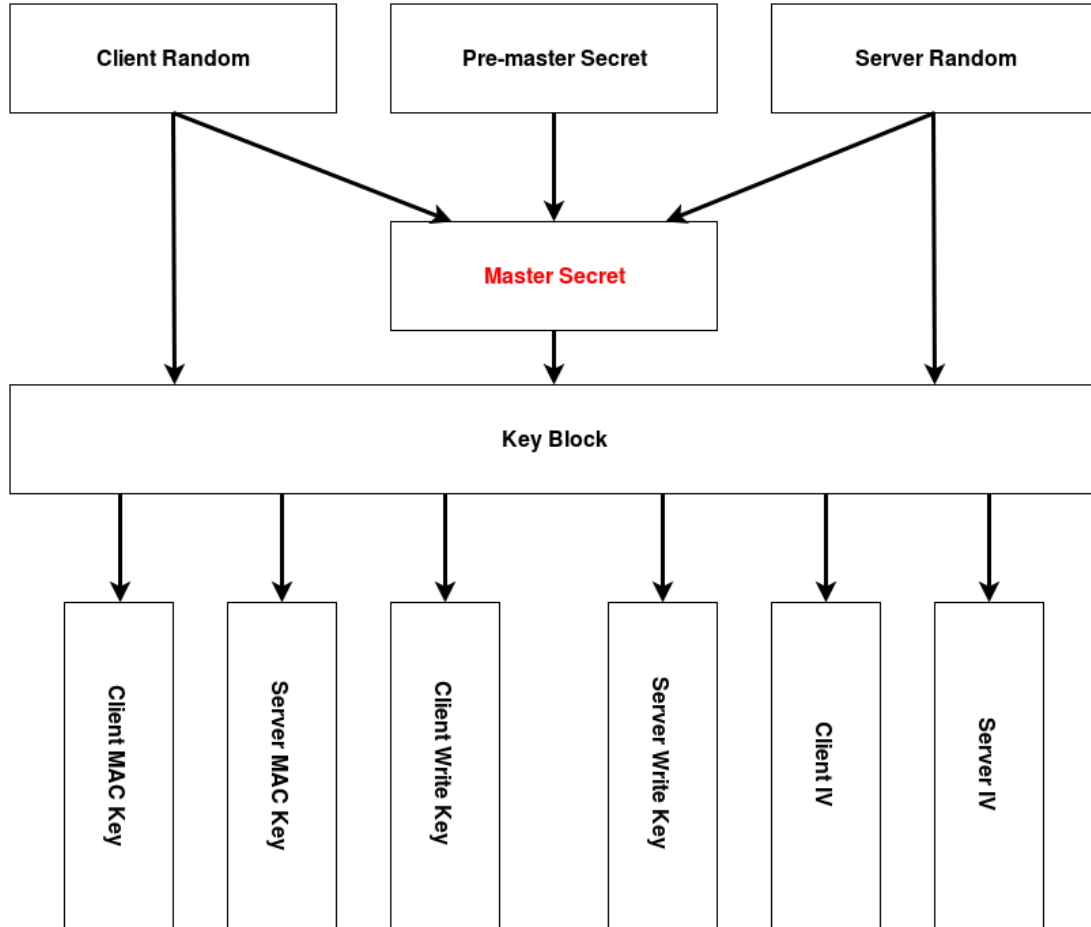
TLS Handshake



Key Material



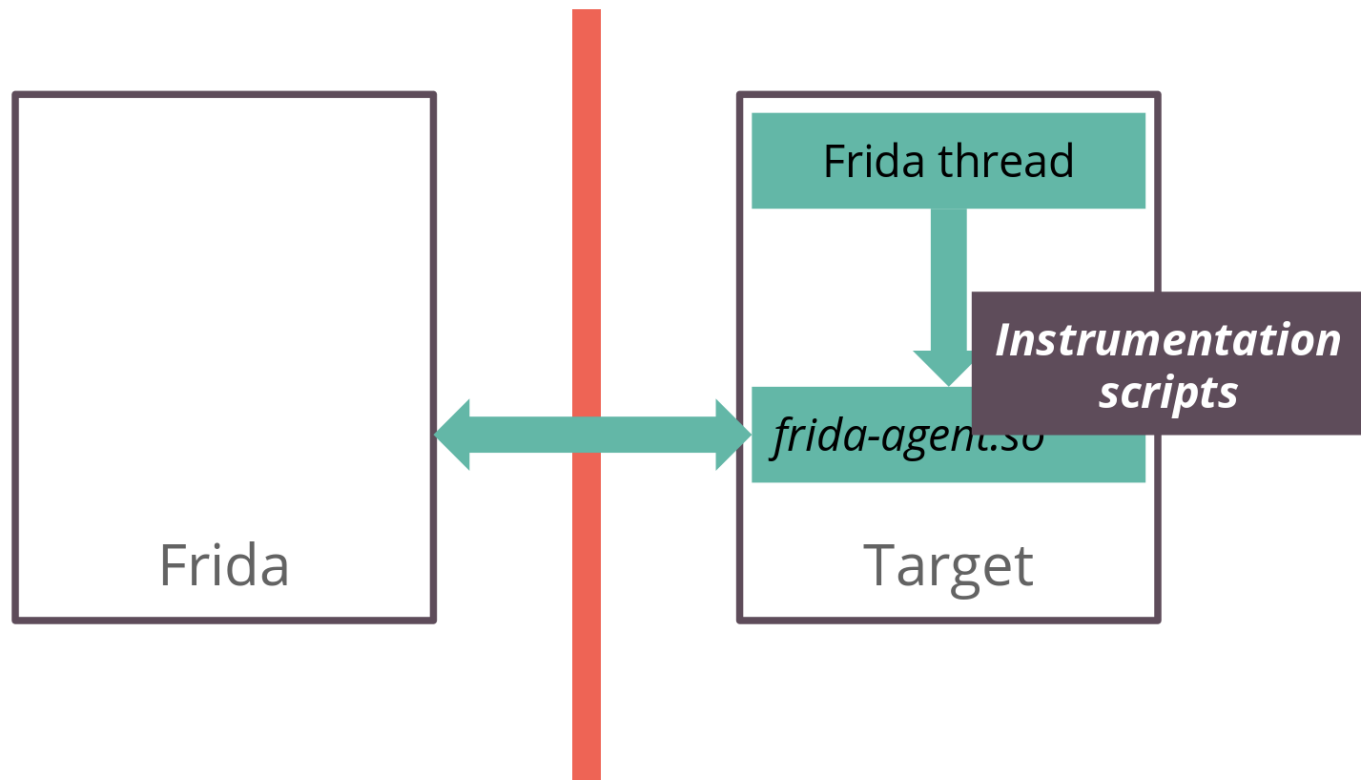
What do we need?



Tools

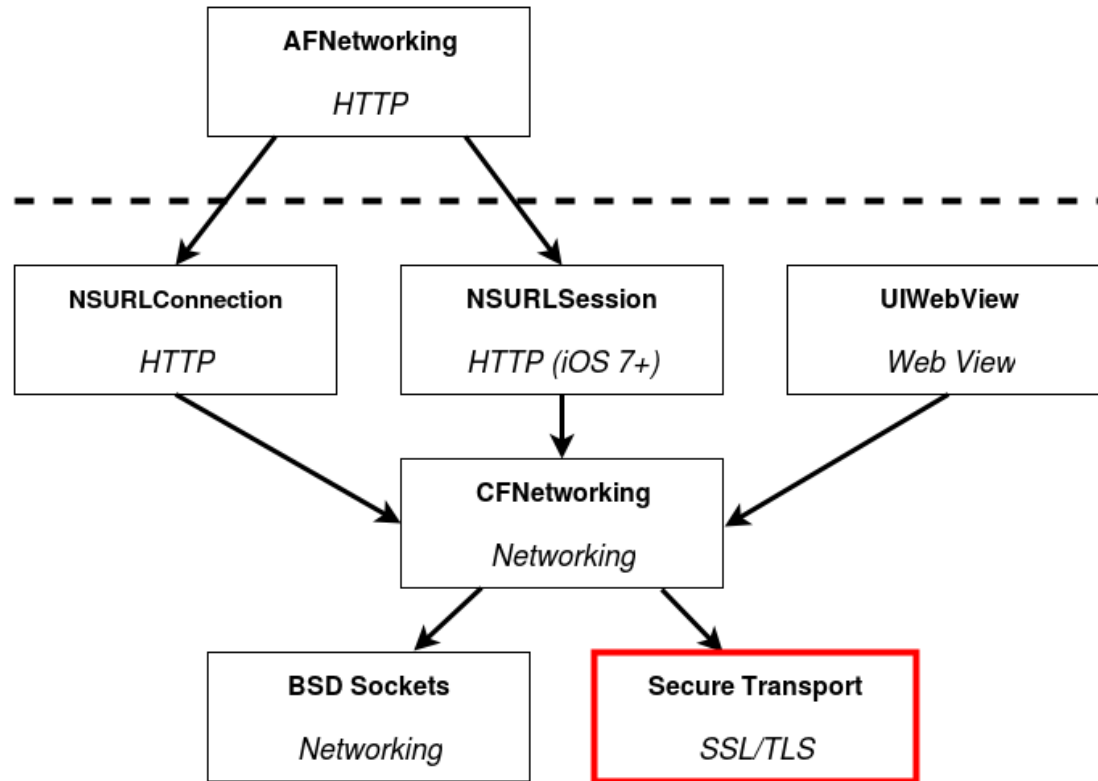
FRIDA

- Dynamic instrumentation toolkit
- Live inspection of processes
- Scriptable
 - Execute own debug scripts *inside* another process
- Used for
 - Attaching to processes
 - Hooking functions
 - *Inspecting memory*



Finding the secret

Targeting iOS Secure Transport



Targeting iOS Secure Transport

- Secure Transport API hides internal handshake operations
- Encryption *actually* handled by coreTLS library
- Source code for older versions available online

CoreTLS source

```
struct _tls_handshake_s {  
  
    tls_protocol_version negProtocolVersion;  
    tls_protocol_version clientReqProtocol;  
    tls_protocol_version minProtocolVersion;  
    tls_protocol_version maxProtocolVersion;  
    ...  
  
    uint8_t      clientRandom[SSL_CLIENT_SRVR_RAND_SIZE];  
    uint8_t      serverRandom[SSL_CLIENT_SRVR_RAND_SIZE];  
    tls_buffer   preMasterSecret;  
    uint8_t      masterSecret[SSL_MASTER_SECRET_SIZE];  
    ...  
}  
  
typedef struct _tls_handshake_s *tls_handshake_t;
```

CoreTLS source

tls_handshake.h

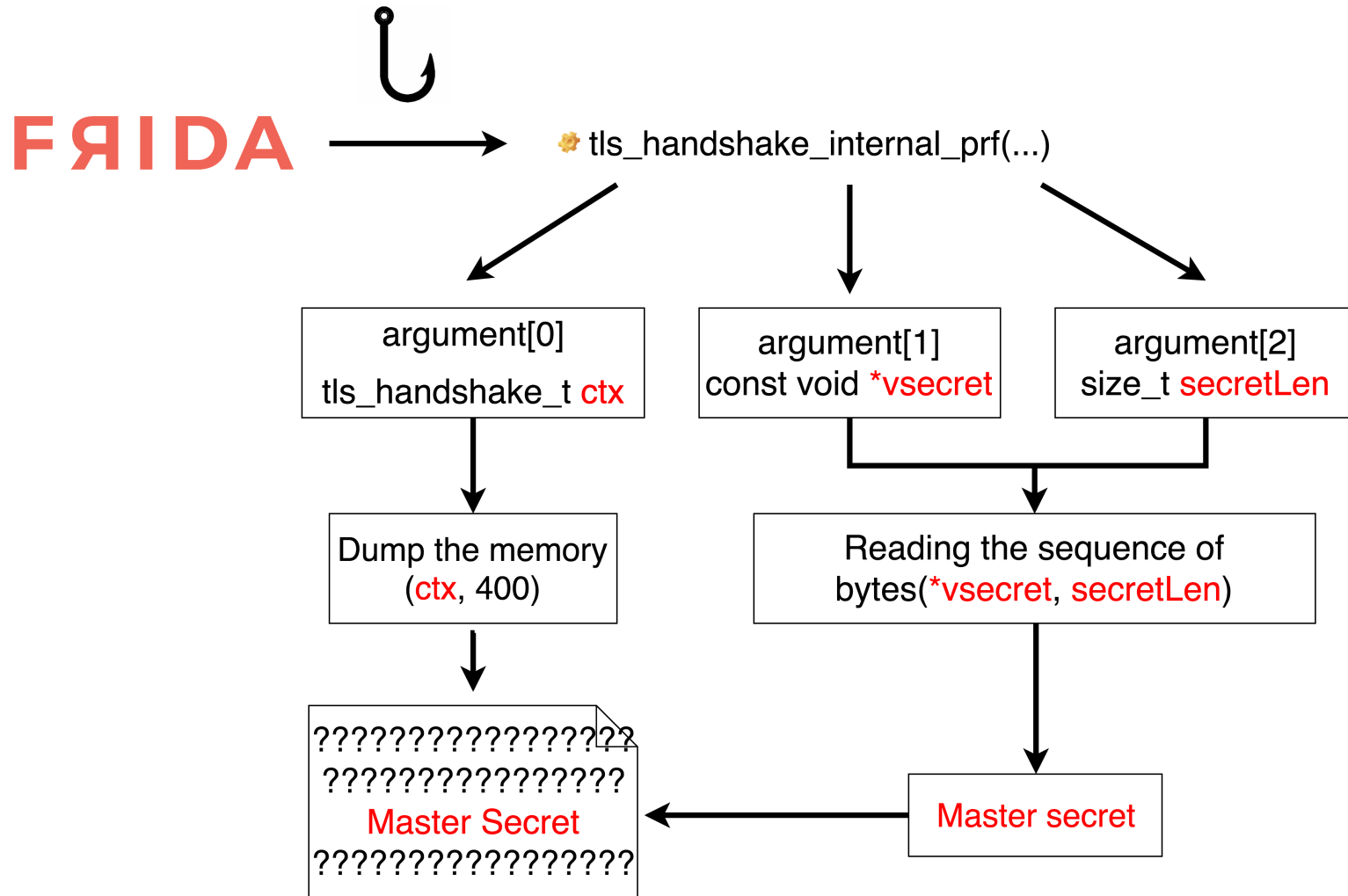
```
int tls_handshake_internal_prf(tls_handshake_t ctx,
    const void *vsecret,
    size_t secretLen,
    const void *label,
    size_t labelLen,
    const void *seed,
    size_t seedLen,
    void *vout,
    size_t outLen);
```

Main object

Master secret

MS length

Using Frida



Memory dump with Frida

```
5540 ms  tls_handshake_internal_prf()
5540 ms  0x18bf2c00
- offset -  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0x00000000  03 03 00 00 03 03 00 00 03 03 00 00 03 03 00 00  .....
0x00000010  00 00 00 00 c0 31 b2 3b 00 00 00 00 00 00 00 00  ....1.;.....
0x00000020  00 00 00 00 00 00 00 00 40 7d f4 17 00 00 00 00  ....@}.....
0x00000030  00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0x00000040  00 00 00 00 d0 de f4 17 03 00 00 00 00 00 00 00  .....
0x00000050  00 00 00 00 ff ff ff ff 00 00 00 00 00 00 00 00  .....
0x00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0x00000070  00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0x00000080  00 00 00 00 00 00 00 00 00 00 00 00 47 00 00 00  .....G...
0x00000090  a0 42 f1 17 31 10 00 00 00 4e dd 18 01 00 00 00  .B..1....N....
0x000000a0  0e 00 00 00 30 f6 f4 17 0e 00 00 00 30 88 03 19  ....0.....0...
0x000000b0  01 00 00 00 00 01 00 00 30 c0 00 00 30 c0 00 00  ....0...0...
0x000000c0  11 00 00 00 20 0c 0c 00 04 00 00 00 50 99 e9 17  ....P...
0x000000d0  30 00 00 00 00 00 00 00 00 00 00 00 0f 00 00 00  0.....
0x000000e0  00 00 00 00 1b 00 00 00 b0 fa 00 19 00 00 00 00  .....
0x000000f0  02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0x00000100  57 72 87 62 1f ab 7f 15 c0 41 c8 41 dd 2e 52 18  Wr.b....A.A..R.
0x00000110  51 fa 6d a4 7a ff fa 5c c8 cf 6a 16 25 d8 02 68  Q.m.z.\.j.%..h
0x00000120  57 72 87 62 a8 07 3b 57 26 2c 50 99 7a 69 f7 7e  Wr.b.;W&,P.zi.~
0x00000130  ea 2c bc 28 bd 18 9d 85 96 de 9b 65 14 0d aa 27  .,.(.....e...!
0x00000140  00 00 00 00 00 00 00 00 3f f4 37 1f 35 a5 9d d0  .....7.5...
0x00000150  70 04 98 f6 e0 58 d4 2b 00 d3 74 d6 49 d1 a6 cf  p.....rdV,..!
0x00000160  6d ae 3f 96 b4 06 65 4e 09 9f 72 64 56 2c b3 27  m.?...enrdV,..!
0x00000170  d9 68 e0 5e 4b 36 04 f3 60 00 00 00 60 9c f3 17  .h.^K6..`...`...
0x00000180  5c 00 00 00 80 76 f4 17 6c 00 00 00 e0 25 f1 17  \....v..l....%..
```

tls_types.h

```
...
TLS_1_0 = 0x0301
TLS_1_1 = 0x0302
TLS_1_2 = 0x0303
...
```



Master secret

Finding the Identifiers

- Captured packets with Wireshark whilst running Frida
- Compared hex outputs to match identifiers

```
5540 ms  tls_handshake_internal_prf()
5540 ms  0x18bf2c00
- offset -  0 1 2 3 4 5 6 7 8 9 A B C D E F  0123456789ABCDEF
0x00000000  03 03 00 00 03 03 00 00 03 03 00 00 03 03 00 00  .....
0x00000010  00 00 00 00 c0 31 b2 3b 00 00 00 00 00 00 00 00  .....1;.....
0x00000020  00 00 00 00 00 00 00 00 00 40 7d f4 17 00 00 00 00  .....@}.....
0x00000030  00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0x00000040  00 00 00 00 d0 de f4 17 03 00 00 00 00 00 00 00  .....
0x00000050  00 00 00 00 ff ff ff ff 00 00 00 00 00 00 00 00  .....
0x00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0x00000070  00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0x00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 47 00 00  .....G...
0x00000090  a0 42 f1 17 31 10 00 00 00 00 4e dd 18 01 00 00 00  ..B..1...N....
0x000000a0  0e 00 00 00 30 f6 f4 17 0e 00 00 00 30 88 03 19  ....0.....0...
0x000000b0  01 00 00 00 01 00 00 00 30 c0 00 00 30 c0 00 00  ..0.....0...
0x000000c0  11 00 00 00 20 0c 0c 00 04 00 00 00 50 99 e9 17  ....P....
0x000000d0  30 00 00 00 00 00 00 00 00 00 00 00 00 0f 00 00  ..0.....
0x000000e0  00 00 00 00 1b 00 00 00 b0 fa 00 19 00 00 00 00  .....
0x000000f0  02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0x00000100  57 72 87 62 1f ab 7f 15 c0 41 c8 41 dd 2e 52 18  ..Wr.b... .A.A..R
0x00000110  51 fa 6d a4 7a ff fa 5c c8 cf 6a 16 25 d8 02 68  ..Q.m.z.. \.j.%..h
0x00000120  57 72 87 62 a8 07 3b 57 26 2c 50 99 7a 69 f7 7e  ..Wr.b..;W&,P.zi.~
0x00000130  ea 2c bc 28 bd 18 9d 85 96 de 9b 65 14 0d aa 27  ..,.(.....e...!
0x00000140  00 00 00 00 00 00 00 00 3f f4 37 1f 35 a5 9d d0  .....?.7.5...
0x00000150  70 04 98 f6 e0 58 d4 2b 00 d3 74 d6 49 d1 a6 cf  ..p...X+.t.I...
0x00000160  6d ae 3f 96 b4 06 65 4e 09 9f 72 64 56 2c b3 27  ..m.?...eN..rdV,..!
0x00000170  d9 68 e0 5e 4b 36 04 f3 60 00 00 00 60 9c f3 17  ..h.^K6...`...
0x00000180  5c 00 00 00 80 76 f4 17 6c 00 00 00 e0 25 f1 17  ..\....v..l....%..
```

▼ Handshake Protocol: Client Hello
Handshake Type: Client Hello (1)
Length: 508
Version: TLS 1.2 (0x0303)

▼ Random

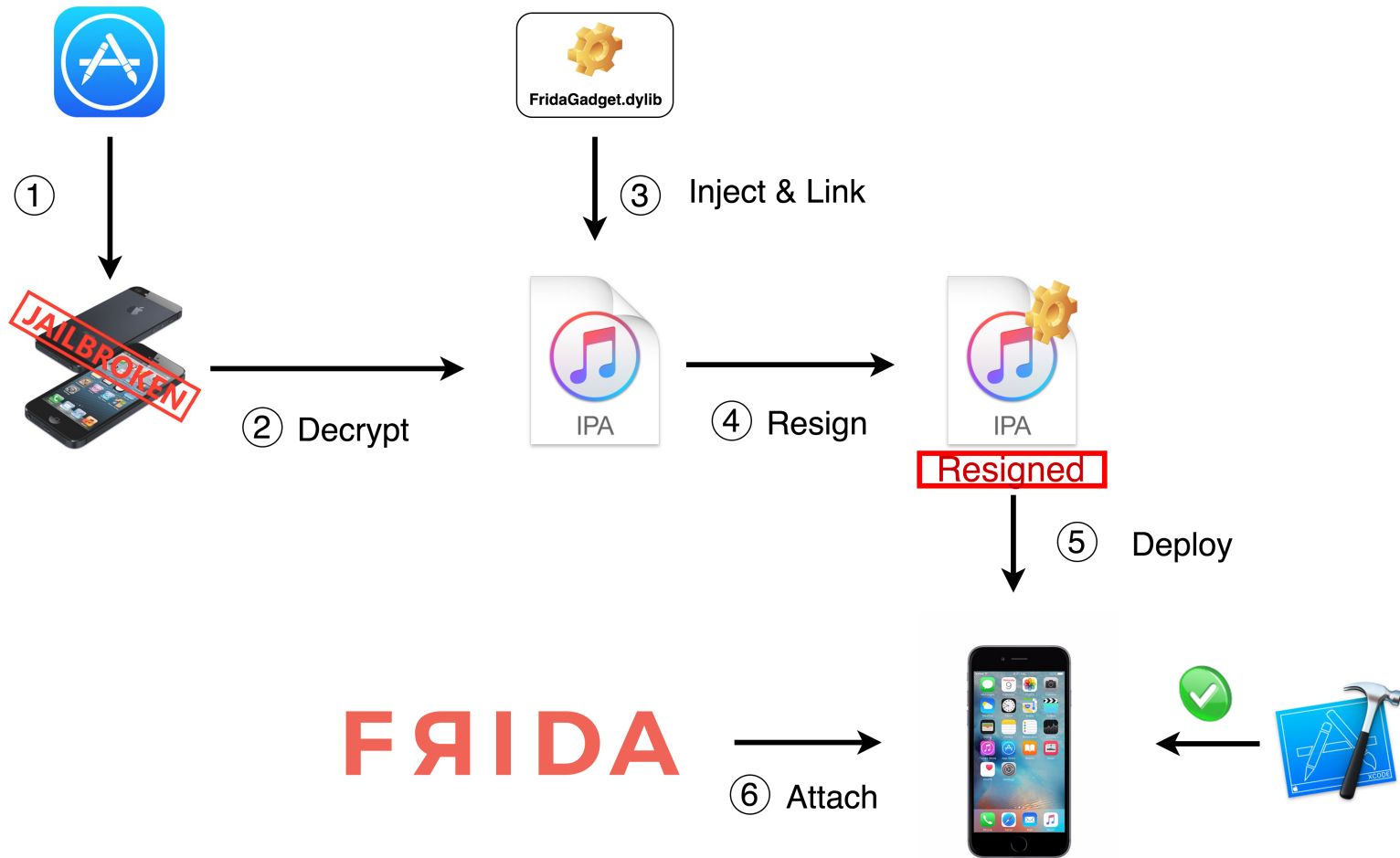
GMT Unix Time: Jun 28, 2016 16:19:14.00000000 CEST
Random Bytes: 1fab7f15c041c841dd2e521851fa6da47afffa5cc8cf6a16...

0020	2b f6 c0 d4 01 bb 0e 90	2b 19 00 00 00 00 d0 02	+..... +.....
0030	ff ff b2 fb 00 00 02 04	05 b4 01 03 03 05 01 01
0040	08 0a 23 13 f3 03 00 00	00 00 04 02 22 0a 4c 8f	..#.....".L.
0050	77 61 9b c6 8b c3 16 03	03 02 00 01 00 01 fc 03	wa.....
0060	03 57 72 87 62 1f ab 7f	15 c0 41 c8 41 dd 2e 52	.Wr.b... .A.A..R
0070	18 51 fa 6d a4 7a ff fa	5c c8 cf 6a 16 25 d8 02	.Q.m.z.. \.j.%..
0080	68 0e 53 45 53 53 49 4f	4e 2d 54 49 43 4b 45 54	h.SESSION-TICKET



Non-jailbroken devices

Compiling Frida into an Application



Demo

Concluding Remarks

Is it possible to extract TLS session keys from the process memory of a device running iOS 9.0 or greater?

- Yes, both with jailbroken and non-jailbroken
- TLS APIs in iOS rely on coreTLS library
- Relies on Frida, also possible with lldb

Future Work

- iOS 10
- Support for OpenSSL in iOS?
- TLS 1.3 in Draft

With special thanks to
Cedric van Bockhaven @ Deloitte

Thank you for your attention!

Questions?