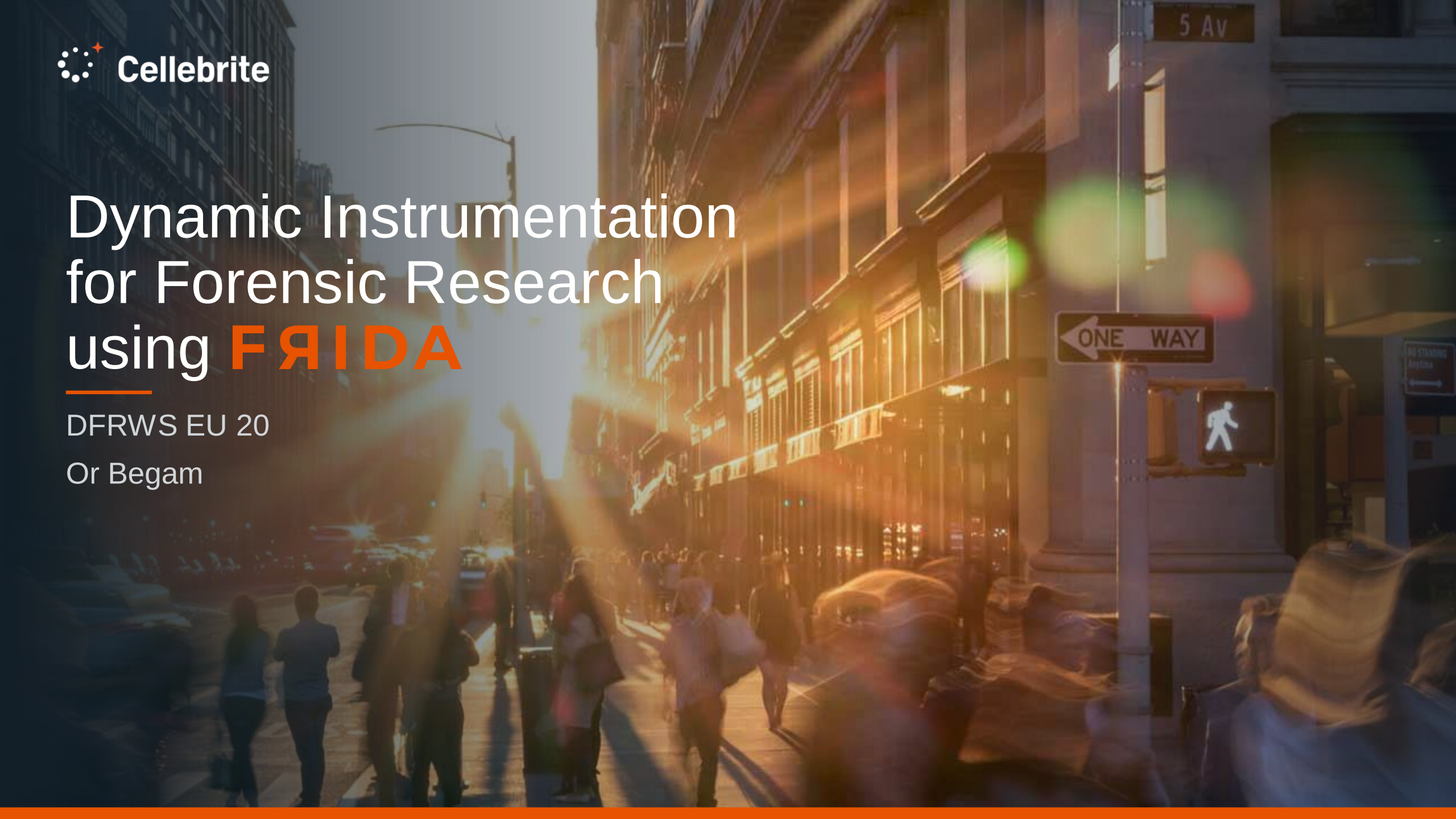


Cellebrite

Dynamic Instrumentation for Forensic Research using FRIDA

DFRWS EU 20

Or Begam

whoami

OR BEGAM

Forensic Research Group, Decoding @ Cellebrite R&D since 2014

- Leader of the advanced forensic research tools team since 2018
- Mobile application and file format reverse engineer

MOD 2007-2013

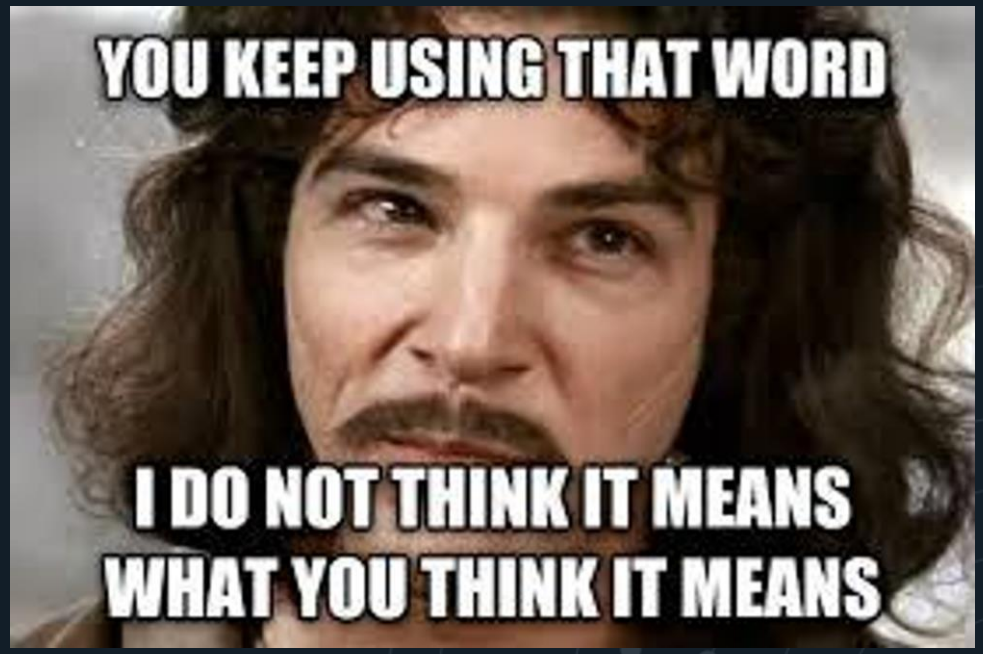
- Communication protocols reverse engineer

Cellebrite

Some Terminology

Decoding

“converting (a coded message) into intelligible language” (Oxford Dictionary)



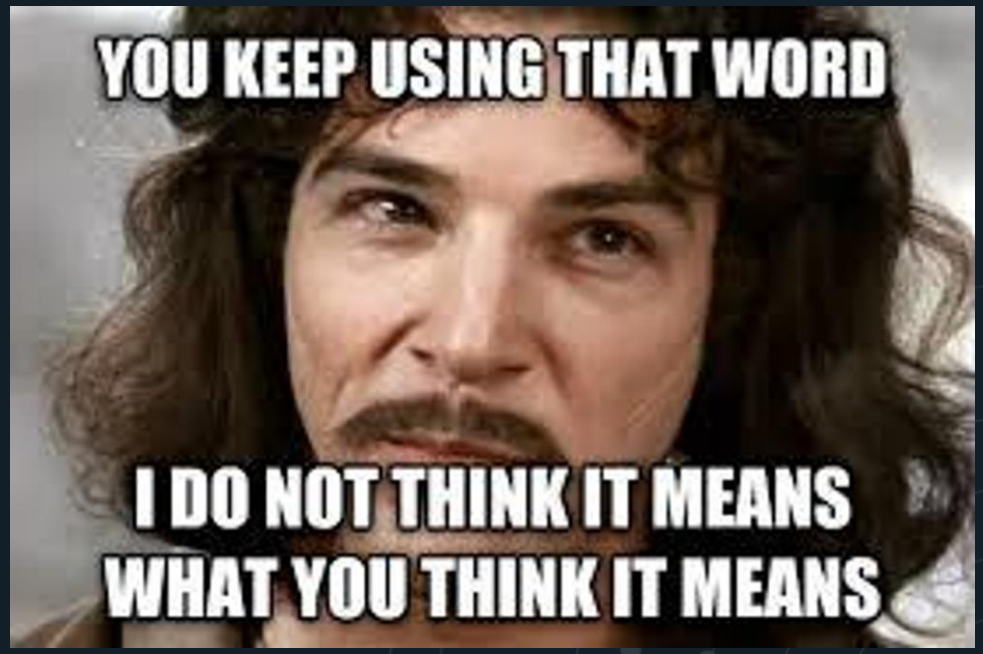
Cellebrite

Some Terminology

Decoding

"converting (a coded message) into intelligible language" (Oxford Dictionary)

Turning **unstructured** data into **structured** data



File Format Decoding

01000100
01100101
01100011
01101111
01100100
01101001
01101110
01100111

File Format Decoding

01000100
01100101
01100011
01101111
01100100
01101001
01101110
01100111

messagesTable (85)

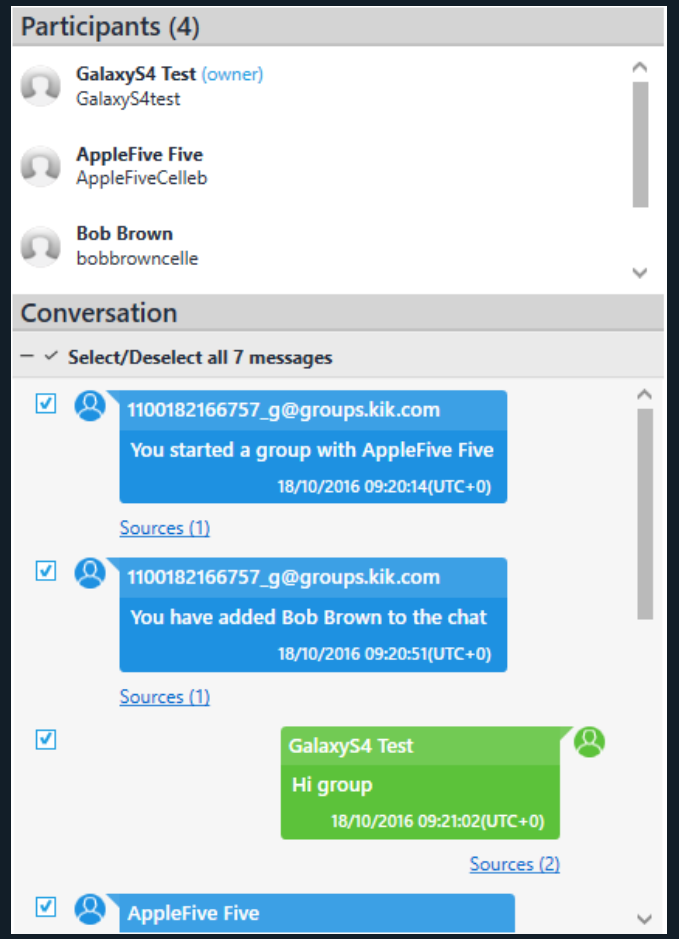
_id	body	partner_jid
15	Yup	bobbrowncelle_cs9@talk.kik.com
52	Yes	1100171362835_g@groups.kik.com
2	Testing KIK version 10.6.0.6560 for pa 5.2	bobbrowncelle_cs9@talk.kik.com
93	Testing kik 10.15.0.9760	applefivecelleb_j8f@talk.kik.com
64	Testing kik 10.14.0.9480	applefivecelleb_j8f@talk.kik.com
42	Testing kik 10.12.1.8849 pa 5.2.5.3 18.08.16	applefivecelleb_j8f@talk.kik.com
30	Testing kik 10.10.0.7817 pa 5.2.0.208 10.08.16	applefivecelleb_j8f@talk.kik.com
75	Testing group	1100073591795_g@groups.kik.com
56	Test KIK 10.13.1.9104 pa 5.3.0.1 13.09.16	applefivecelleb_j8f@talk.kik.com
57	Test back	applefivecelleb_j8f@talk.kik.com
19	Pictur	bobbrowncelle_cs9@talk.kik.com
13	Kik version 10.8.0.7090 for pa 5.2	bobbrowncelle_cs9@talk.kik.com
39	I'm still getting set up, check back soon!	dan_lh8@talk.kik.com
41	I'm still getting set up, check back soon!	dan_lh8@talk.kik.com
44	Hug	applefivecelleb_j8f@talk.kik.com
70	Hlo	applefivecelleb_j8f@talk.kik.com

From Decoding to Forensic Research

01000100
01100101
01100011
01101111
01100100
01101001
01101110
01100111

messagesTable (85)

_id	body	partner_jid
15	Yup	bobbrowncelle_cs9@talk.kik.com
52	Yes	1100171362835_g@groups.kik.com
2	Testing KIK version 10.6.0.6560 for pa 5.2	bobbrowncelle_cs9@talk.kik.com
93	Testing kik 10.15.0.9760	applefivecelleb_j8f@talk.kik.com
64	Testing kik 10.14.0.9480	applefivecelleb_j8f@talk.kik.com
42	Testing kik 10.12.1.8849 pa 5.2.5.3 18.08.16	applefivecelleb_j8f@talk.kik.com
30	Testing kik 10.10.0.7817 pa 5.2.0.208 10.08.16	applefivecelleb_j8f@talk.kik.com
75	Testing group	1100073591795_g@groups.kik.com
56	Test KIK 10.13.1.9104 pa 5.3.0.1 13.09.16	applefivecelleb_j8f@talk.kik.com
57	Test back	applefivecelleb_j8f@talk.kik.com
19	Pictur	bobbrowncelle_cs9@talk.kik.com
13	Kik version 10.8.0.7090 for pa 5.2	bobbrowncelle_cs9@talk.kik.com
39	I'm still getting set up, check back soon!	dan_lh8@talk.kik.com
41	I'm still getting set up, check back soon!	dan_lh8@talk.kik.com
44	Hug	applefivecelleb_j8f@talk.kik.com
70	Hlo	applefivecelleb_j8f@talk.kik.com



Cellebrite

Cellebrite

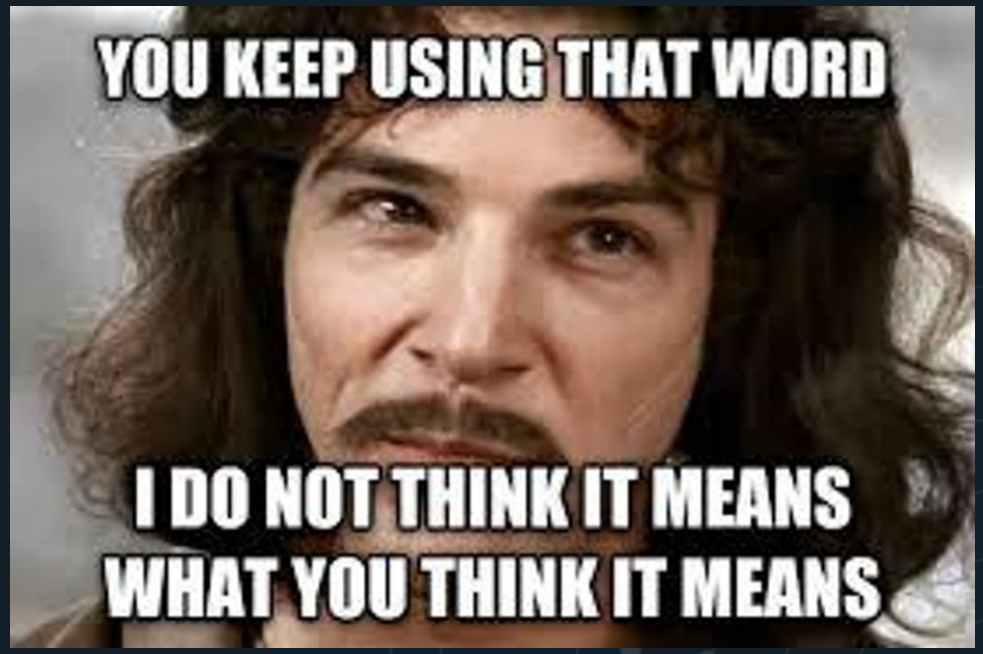
Some Terminology

Automatic Decoding

When your tool does the decoding

Manual Decoding

When you do the decoding (or need to write an automatic tool)



Agenda

Overview of mobile app decoding

And some technical challenges in it

Introduction to dynamic instrumentation

And how it can solve those challenges

Hands-on work with FRIDA

Solving more hard problems

Cellebrite

Cellebrite

Let’s decode Instagram

A fast introduction to mobile app decoding

Let’s decode Instagram

Generate data

Install app, create users, send some messages

Perform extraction*

*or pull files from rooted/jailbroken device

Dig in

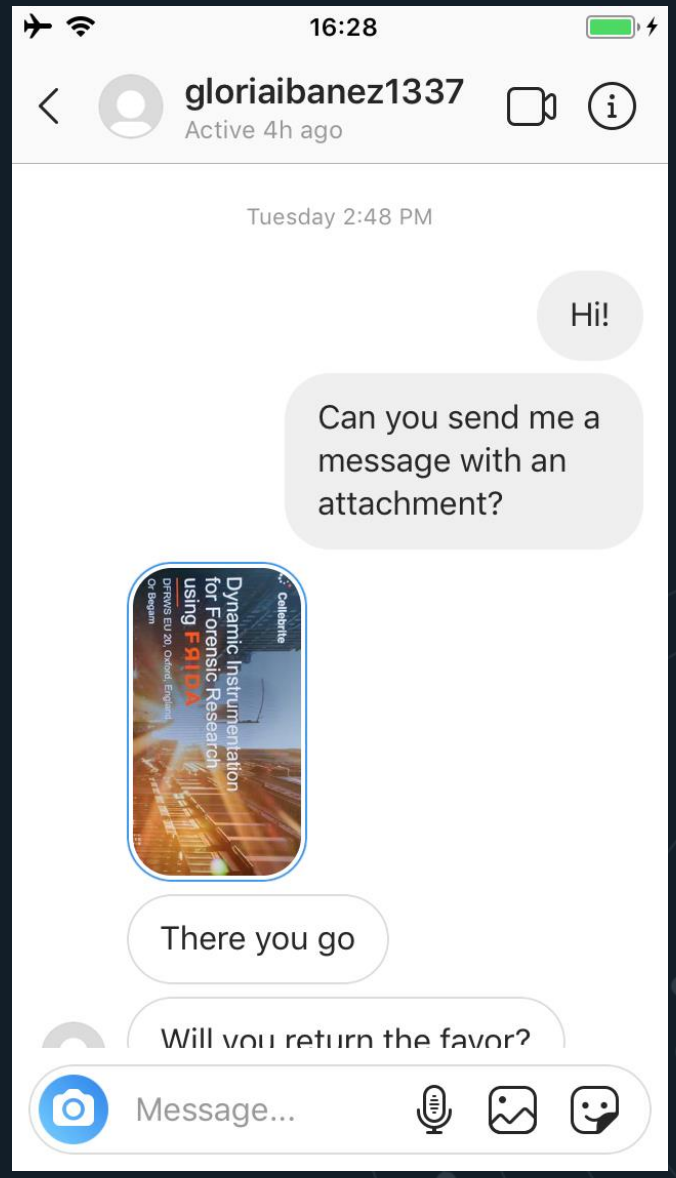
Go through files, look for our data (and more)

Cellebrite

Cellebrite

Chat decoding

Here’s a chat. Where’s the data?



Cellebrite

Databases

Row count	Name	Path
0	Cache.db	test_1302_1600/Library/Caches/com.burbn.instagram/IGCache/Cache.db
6	time_in_app_175017481...	test_1302_1600/Documents/time_in_app_1750174813.db
12	1750174813.db	test_1302_1600/Library/Application Support/DirectSQLiteDatabase/1750174813.db

Cellebrite

Databases >

1750174813.db

Tables

experiment_state	(0)
inbox_metadata	(1)
messages	(7)
mutations	(0)
quick_reply	(0)
sqlite_master	(10)
sqlite_sequence	(3)
thread_client_state	(0)
threads	(1)

Cellebrite

Databases >

1750174813.db

Tables

experiment_state	(0)
inbox_metadata	(1)
messages	(7)
mutations	(0)
quick_reply	(0)
sqlite_master	(10)
sqlite_sequence	(3)
thread_client_state	(0)
threads	(1)

Cellebrite

Databases >
1750174813.db >

Messages table

message_id	thread_id	archive	class_name	row_id
291721804449134575653008460805570 56	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	21
291721801943931700276631797434941 44	340282366841710300949128145361214789666	bplist00▯▯▯▯X$versionX$objectsY$archiverT$top▯▯▯▯▯-▯...	IGDirectPublishedMessage	20
291721783820366097236197619353518 08	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	19
291721779958893480554114526457364 48	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	18
291721774617076108317265714720276 48	340282366841710300949128145361214789666	bplist00▯▯▯▯X$versionX$objectsY$archiverT$top▯▯▯▯▯*▯...	IGDirectPublishedMessage	17
291721482705113162834946896649256 96	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	16
291721478003124982610969361718968 32	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	15

Cellebrite

Databases >
1750174813.db >

Messages table

matches threads table

message_id	thread_id	archive	class_name	row_id
291721804449134575653008460805570 56	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	21
291721801943931700276631797434941 44	340282366841710300949128145361214789666	bplist00▯▯▯▯X$versionX$objectsY$archiverT$top▯▯▯▯▯-▯...	IGDirectPublishedMessage	20
291721783820366097236197619353518 08	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	19
291721779958893480554114526457364 48	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	18
291721774617076108317265714720276 48	340282366841710300949128145361214789666	bplist00▯▯▯▯X$versionX$objectsY$archiverT$top▯▯▯▯▯*▯...	IGDirectPublishedMessage	17
291721482705113162834946896649256 96	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	16
291721478003124982610969361718968 32	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	15

Cellebrite

Databases >
1750174813.db >

Messages table

matches threads table

Looks like content is here

message_id	thread_id	archive	class_name	row_id
291721804449134575653008460805570 56	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	21
291721801943931700276631797434941 44	340282366841710300949128145361214789666	bplist00▯▯▯▯X$versionX$objectsY$archiverT$top▯▯▯▯▯-▯...	IGDirectPublishedMessage	20
291721783820366097236197619353518 08	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	19
291721779958893480554114526457364 48	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	18
291721774617076108317265714720276 48	340282366841710300949128145361214789666	bplist00▯▯▯▯X$versionX$objectsY$archiverT$top▯▯▯▯▯*▯...	IGDirectPublishedMessage	17
291721482705113162834946896649256 96	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	16
291721478003124982610969361718968 32	340282366841710300949128145361214789666	bplist00▯▯OPX$versionX$objectsY$archiverT$top▯▯▯▯▯...	IGDirectPublishedMessage	15

Cellebrite

Databases >
1750174813.db >
messages table >

archive column

Single cell

Cellebrite

Databases >
1750174813.db >
messages table >

archive column

Single cell

Cellebrite

Databases >
1750174813.db >
messages table >

archive column

Single cell

Sender

Timestamp

Body

```
IGDirectPublishedMessage = {
    NSArray<NSString *>*likers : NSArray = [ ]
  IGDirectPublishedMessageMetadata*metadata : IGDirectPublishedMessageMetadata = {
      NSString*senderPk : AsciiString = 1750174813
      NSString*threadId : AsciiString = 340282366841710300949128145361214789666
      NSDate*serverTimestamp : NSDate = 11/02/2020 12:48:21
      NSString*clientContext : AsciiString = 6632978465307266002
      NSString*serverId : AsciiString = 29172147800312498261096936171896832
  IGDirectPublishedMessageContent*content : IGDirectPublishedMessageContent = {
      NSString*string : AsciiString = Hi!
      NSArray<NSString *>*mentionedUserPks : AsciiString = Null
      codedSubtype : AsciiString = SUBTYPE_TEXT
    NSArray<IGDirectMessageReaction *>*reactions : NSArray = [ ]
```

Cellebrite

What about **Attachments?**

I sent this image

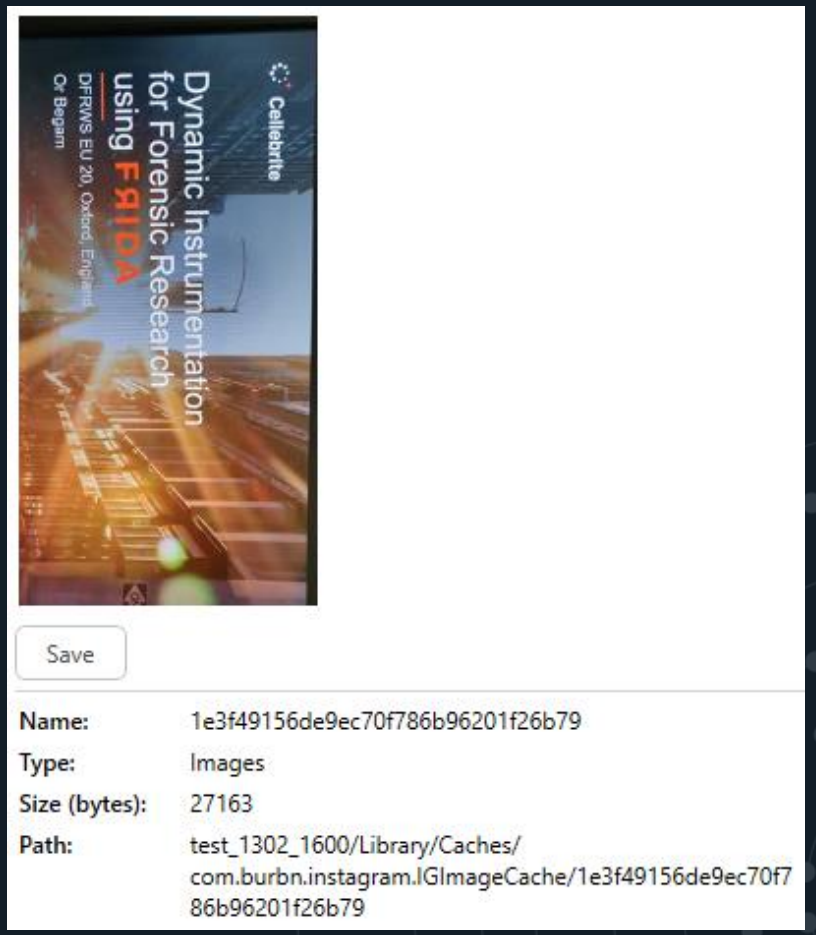
To which message record does it belong?

I sent this image

To which message record does it belong?

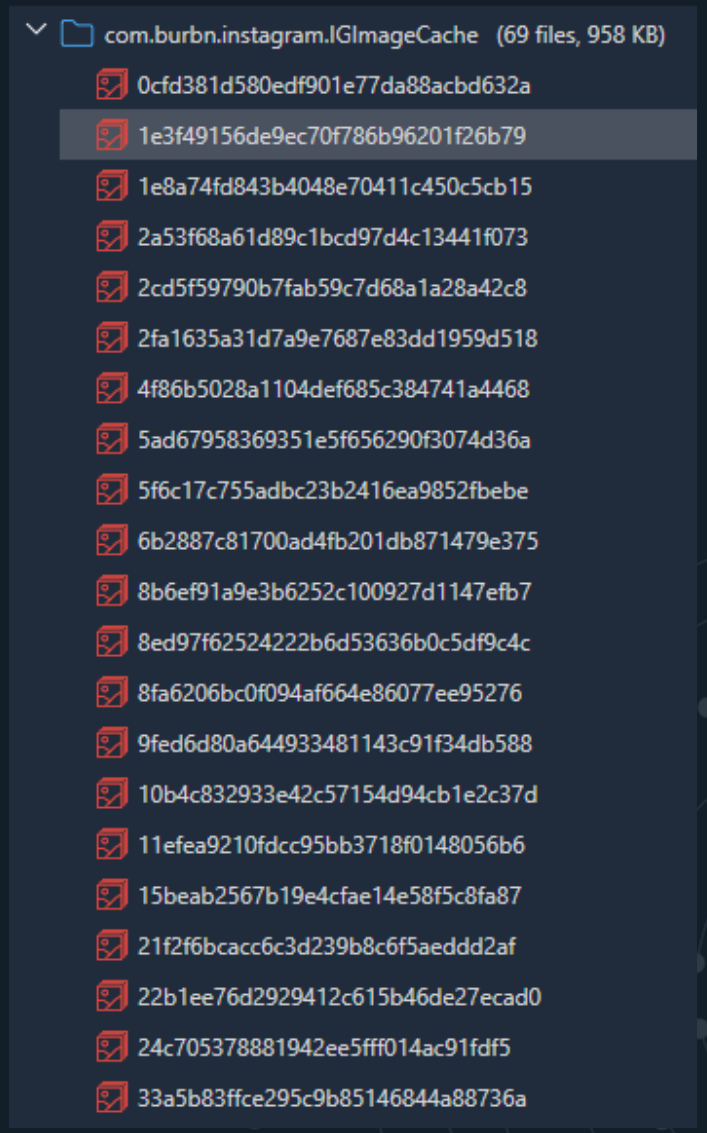
File name looks intriguing…

Stored in some image cache?



Cellebrite

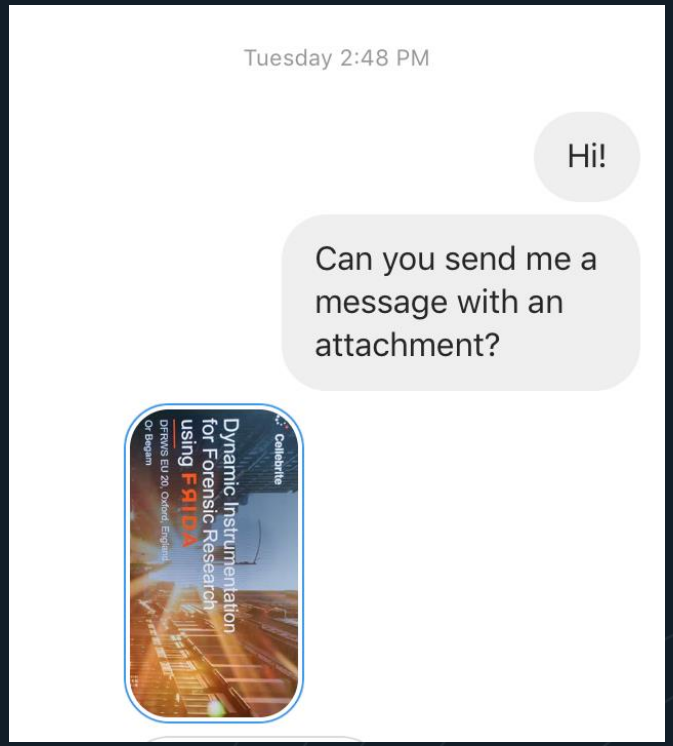
Check the folder

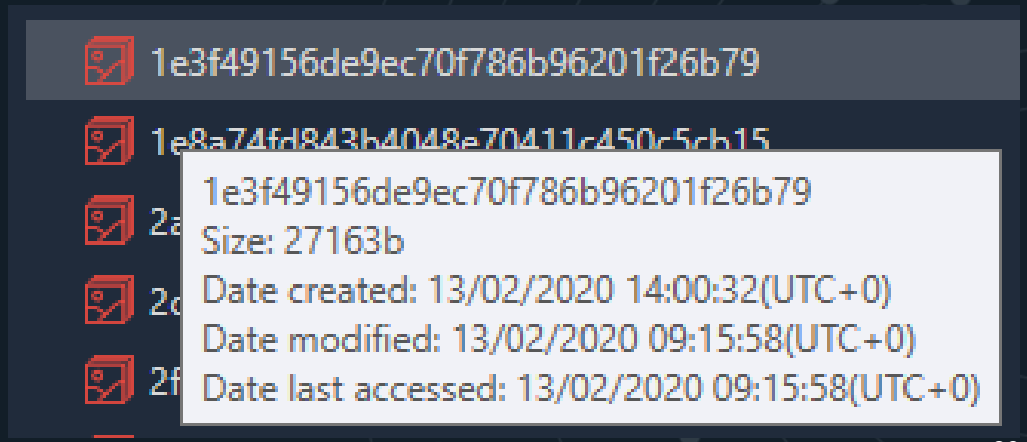


Cellebrite

Compare timestamps

No good if file wasn't created when message was sent…

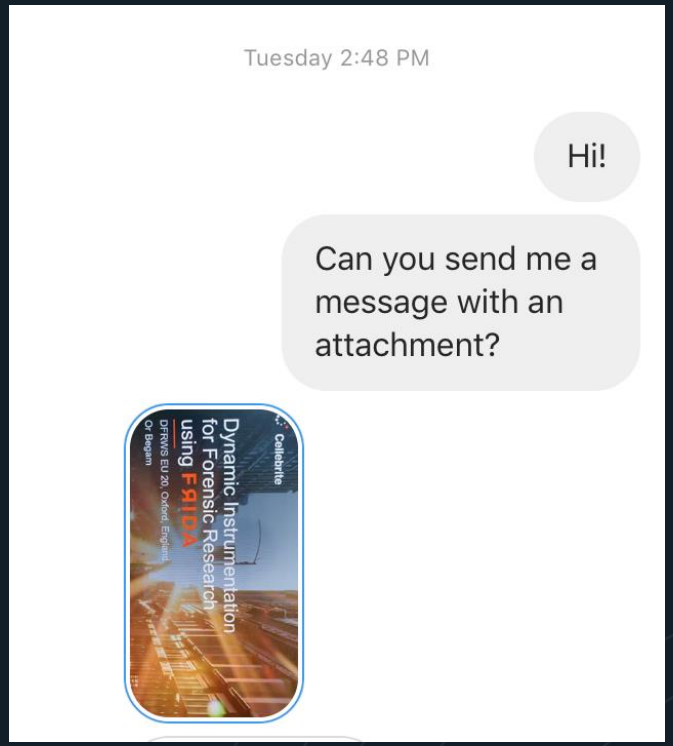


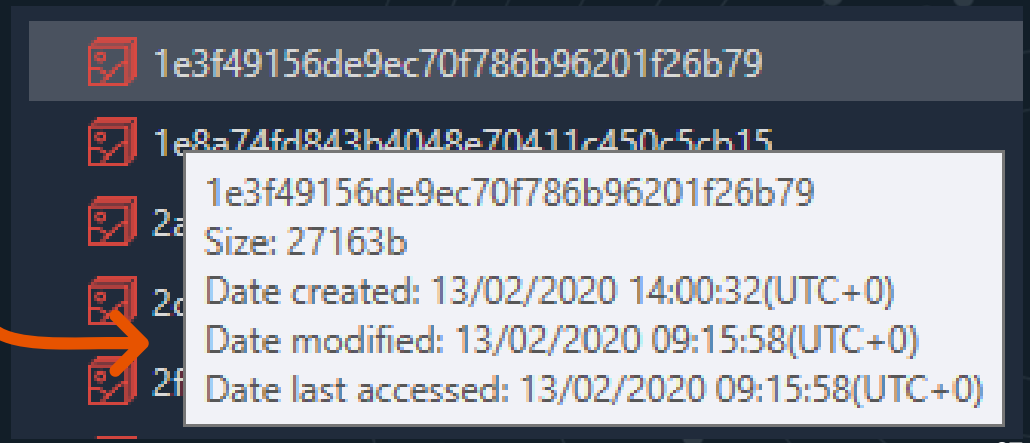


Cellebrite

Compare timestamps

No good if file wasn't created when message was sent…



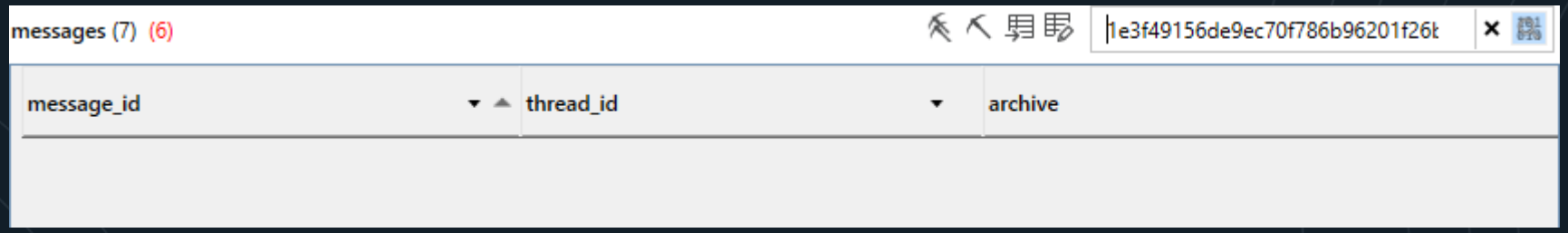


(Not Tuesday)

Cellebrite

grep

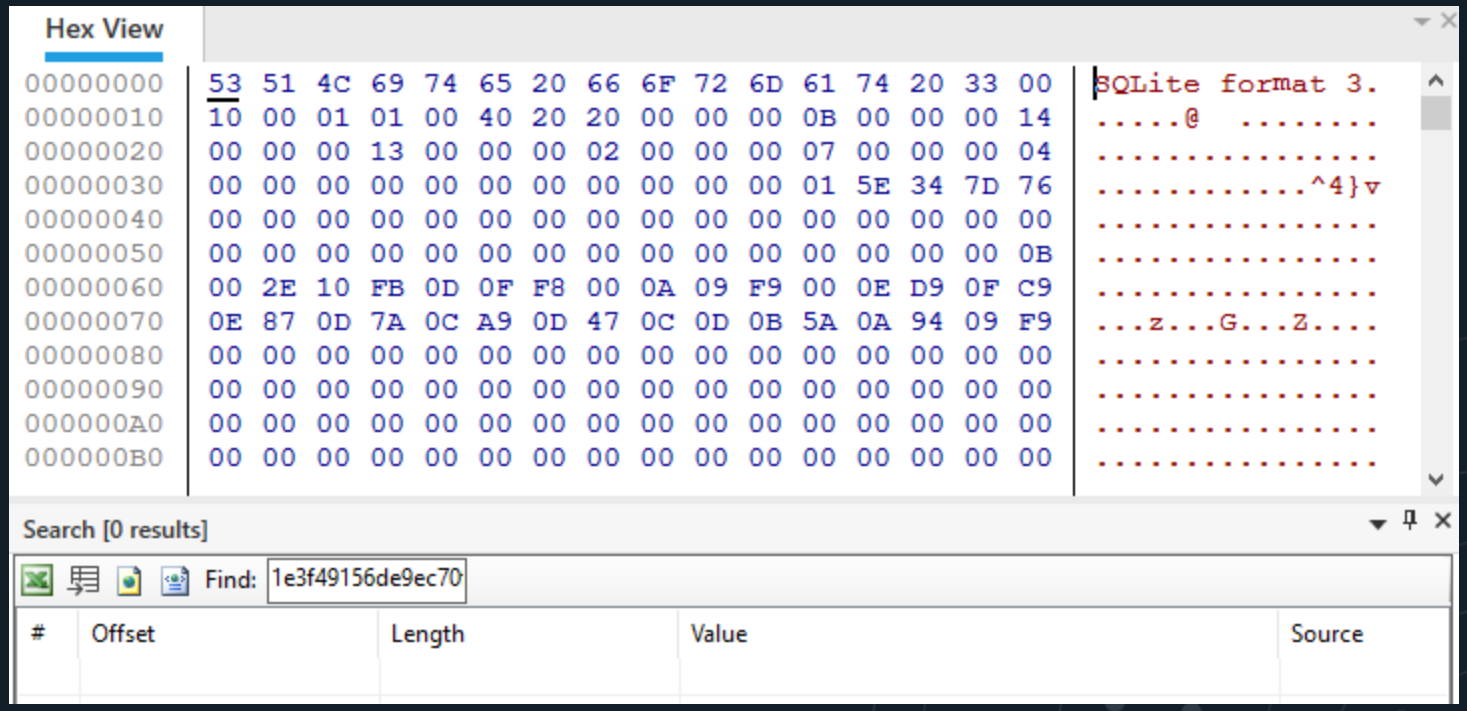
Search file name in database



Cellebrite

grep harder

Search in the raw database data



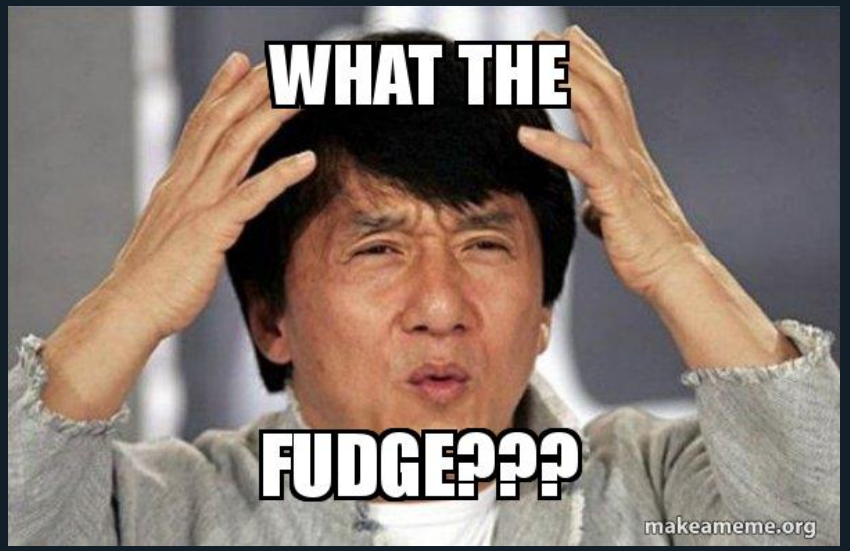
Cellebrite

GREP HARDER

Search in all files in the extraction

```
============== 'ds' is now set to project: test_1302_1600 ===========
>>> fs = ds.FileSystems[0]
>>> nodes = fs.FindFilesContainingValue('1e3f49156de9ec70f786b96201f26b79')
>>> len(nodes)
0
>>> nodes = fs.FindFilesContainingValue('1e3f49156de9ec70f786b96201f26b79'.decode('hex'))
>>> len(nodes)
0
>>>
```

Cellebrite



GREP HARDER

Search in all files in the extraction

```
============== 'ds' is now set to project: test_1302_1600 ===========
>>> fs = ds.FileSystems[0]
>>> nodes = fs.FindFilesContainingValue('1e3f49156de9ec70f786b96201f26b79')
>>> len(nodes)
0
>>> nodes = fs.FindFilesContainingValue('1e3f49156de9ec70f786b96201f26b79'.decode('hex'))
>>> len(nodes)
0
>>>
```

Cellebrite

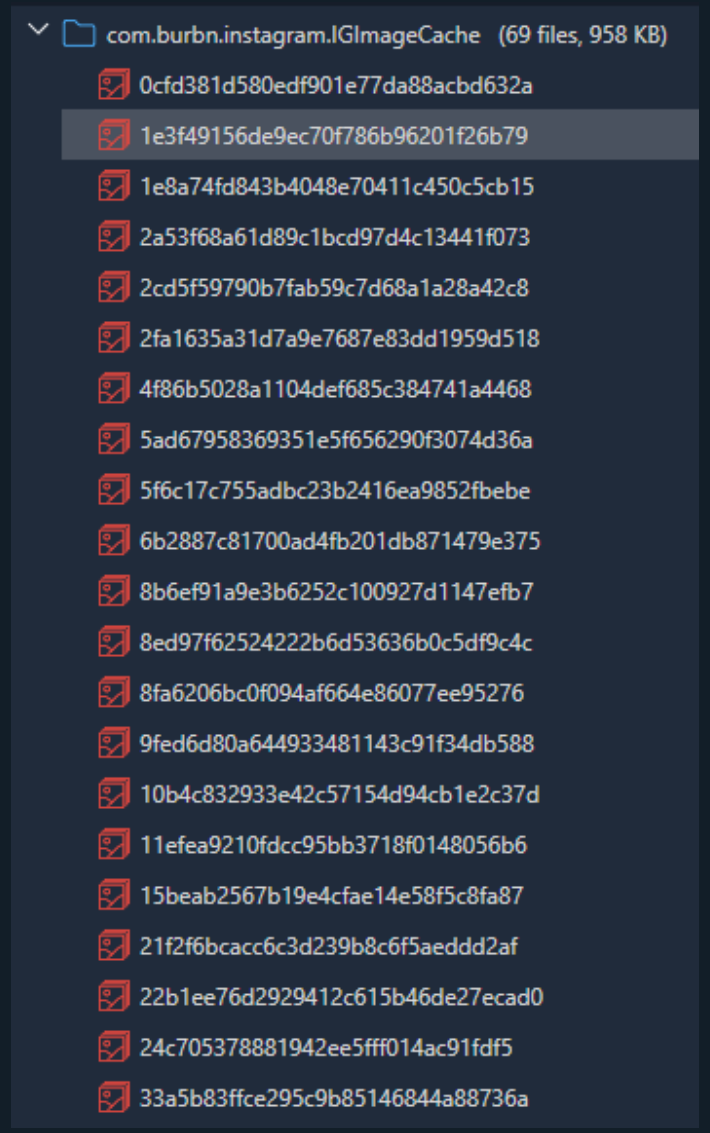
Take a step back

How do caches work?

Cellebrite

Take a step back

How do caches work?

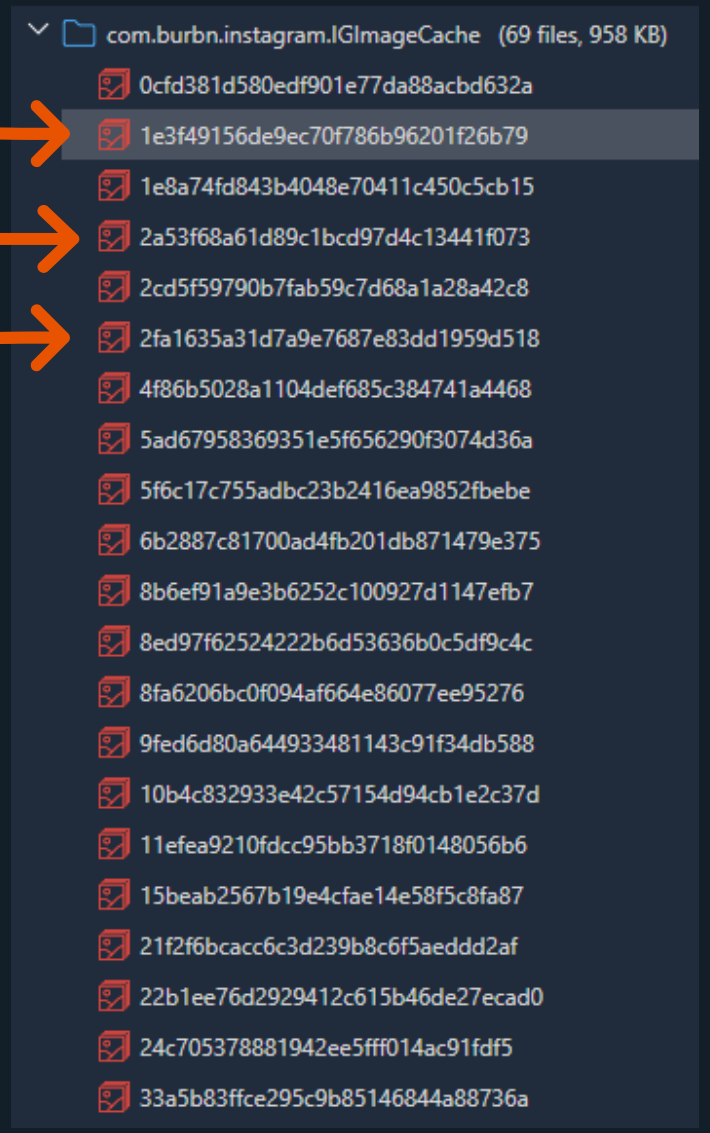


Cellebrite

Take a step back

How do caches work?

Cache keys

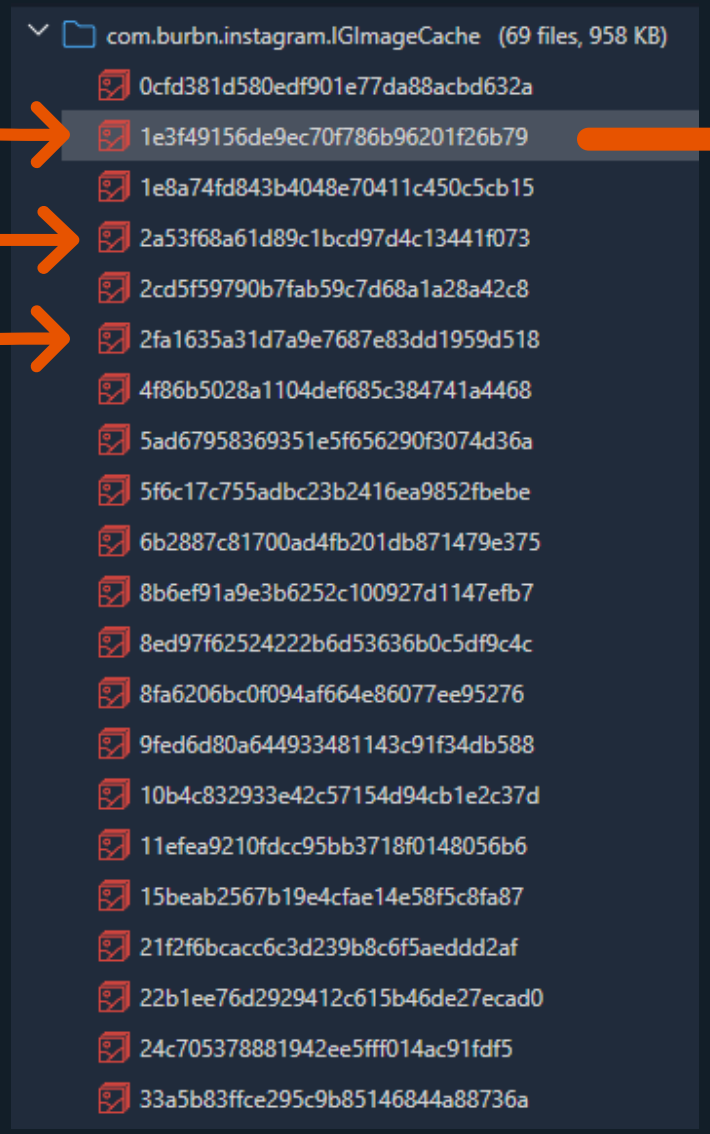


Cellebrite

Take a step back

How do caches work?

Cache keys



Cached content

Cellebrite

Analyzing cache keys

Image filenames are oddly specific…

Cellebrite

Analyzing cache keys

Image filenames are oddly specific…

How are they generated?

Cellebrite

1E3F49156DE9EC70F786B96201F26B79

32 Characters

Cellebrite

32 Characters

16 Bytes (hex encoded)

Cellebrite

1E3F49156DE9EC70F786B96201F26B79

32 Characters

16 Bytes (hex encoded)

Looks like a MD5 hash

Cellebrite

Analyzing cache keys

Hash of what?

`1e3f49156de9ec70f786b96201f26b79`

Looks like a MD5 hash

Cellebrite

Analyzing cache keys

Hash of what?

Of the image's data?

`1e3f49156de9ec70f786b96201f26b79`

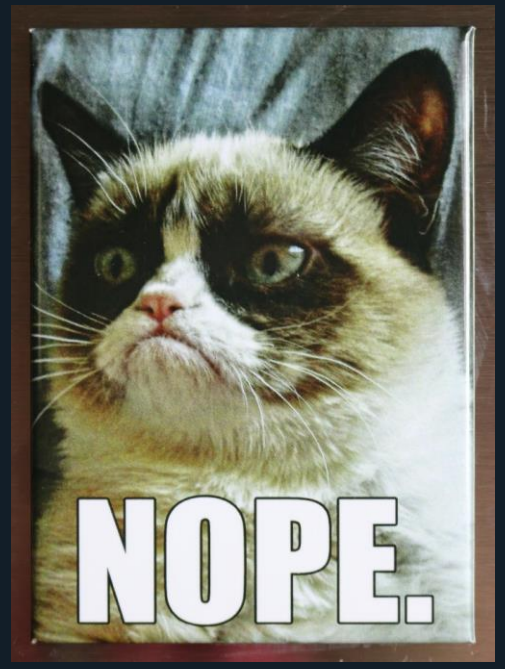
Looks like a MD5 hash

Cellebrite

Analyzing cache keys

Hash of what?

Of the image’s data?



`1e3f49156de9ec70f786b96201f26b79`

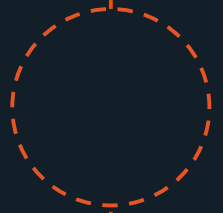
Looks like a MD5 hash

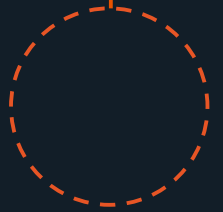
Reverse Engineering Thought Experiment

How does an app fetches images from the cache?

Iterate over message records

Check if message contains media attachments





Cellebrite

Reverse Engineering Thought Experiment

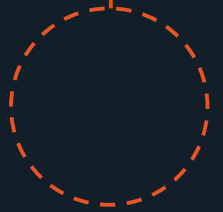
How does an app fetches images from the cache?

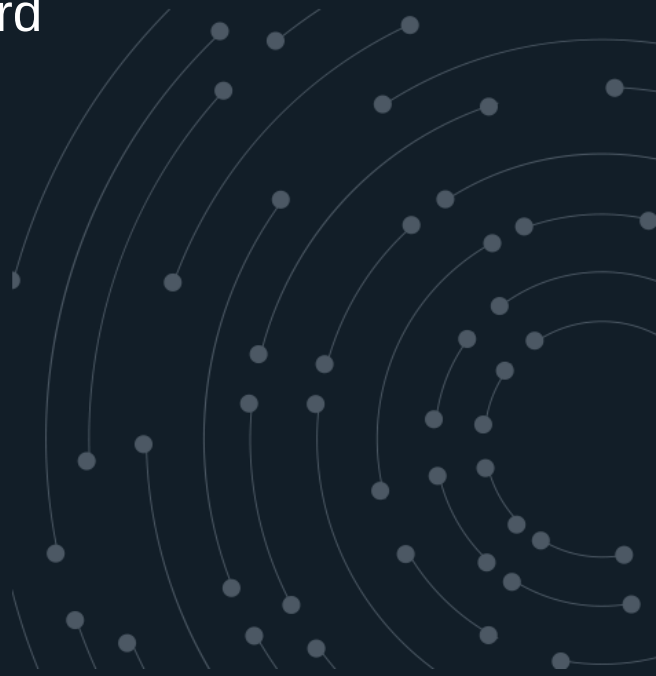
Iterate over message records

Check if message contains media attachments

Calculate cache key hash

From (some of) the content of the record





Cellebrite

Reverse Engineering Thought Experiment

How does an app fetches images from the cache?

Iterate over message records

Check if message contains media attachments

Calculate cache key hash

From (some of) the content of the record

Check if we have a cache hit

i.e. file exists in the cache folder

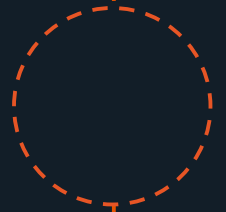
Cellebrite

Reverse Engineering Thought Experiment

How does an app fetches images from the cache?

Cache hit

Take image from cache folder



Cellebrite

Reverse Engineering Thought Experiment

How does an app fetches images from the cache?

Cache hit

Take image from cache folder

Cache miss

Download image from server

Cellebrite

Reverse Engineering Thought Experiment

How does an app fetches images from the cache?

Cache hit

Take image from cache folder

Cache miss

Download image from server

So we need a URL..

Cellebrite

Cellebrite

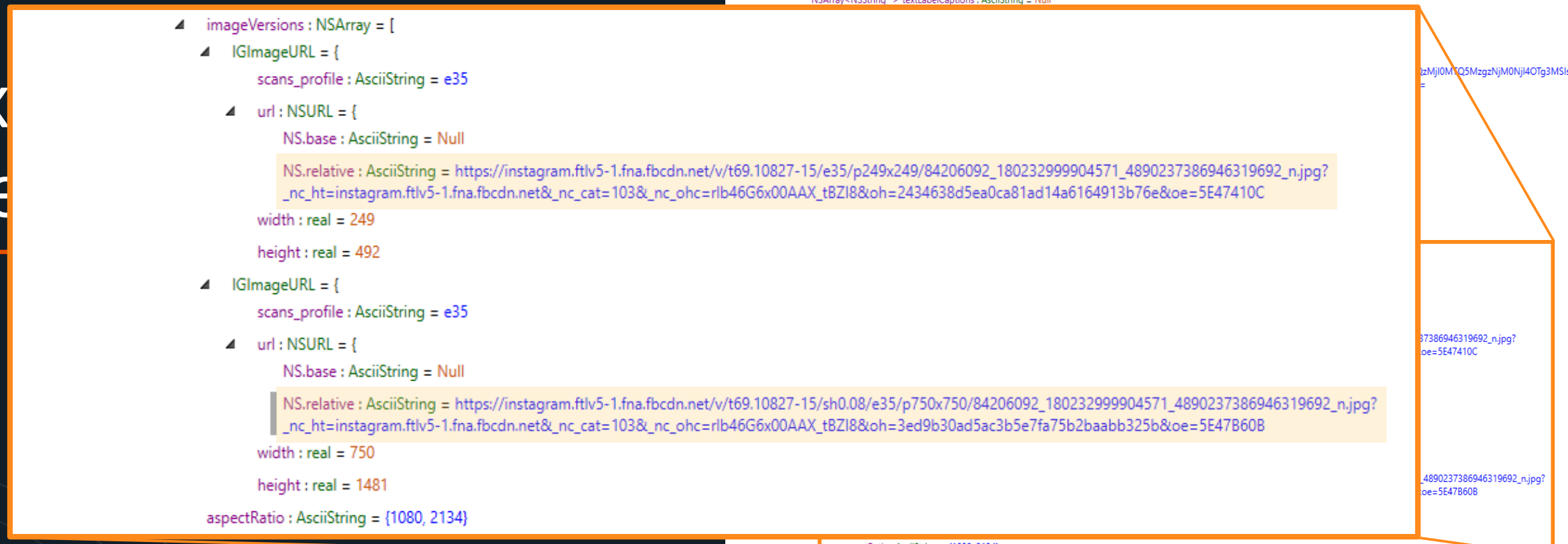
Examining our message record

Cellebrite

Examining our message record

Cellebrite

Ex
me



IGDirectPublishedMessage = {
NSArray<NSString *>*likers : NSArray = [ ]
IGDirectPublishedMessageMetadata*metadata : IGDirectPublishedMessageMetadata = {
NSString*senderPk : AsciiString = 30035802843
NSString*threadId : AsciiString = 340282366841710300949128145361214789666
NSDate*serverTimestamp : NSDate = 11/02/2020 13:15:09
NSString*clientContext : AsciiString = 6632985200461639697
NSString*serverId : AsciiString = 29172177461707610831726571472027648
IGDirectPublishedMessageContent*content : IGDirectPublishedMessageContent = {
IGDirectPublishedMessageMedia*media : IGDirectPublishedMessageMedia = {
IGDirectPublishedMessageVisualMediaInfo*visualMedia : IGDirectPublishedMessageVisualMediaInfo = {
NSArray<NSString *>*textLabelCaptions : AsciiString = Null
aspectRatio : AsciiString = {1080, 2134}
codedSubtype : AsciiString = SUBTYPE_PHOTO
NSArray<IGStoryOverlayTapModel *>*tapModels : NSArray = [ ]
IGCreativeConfig*creativeConfig : AsciiString = Null
NSDate*mediaUrlGoesStaleDate : NSDate = 14/02/2020 01:18:57
NSIntegerseenCountForCurrentUser : integer = 0
IGDirectVisualMessageViewModeviewMode : integer = 2
NSOrderedSet<IGUser *>*viewers : NSOrderedSet = { }
codedSubtype : AsciiString = SUBTYPE_VISUAL_MEDIA
codedSubtype : AsciiString = SUBTYPE_MEDIA
NSArray<IGDirectMessageReaction *>*reactions : NSArray = [ ]

Cellebrite

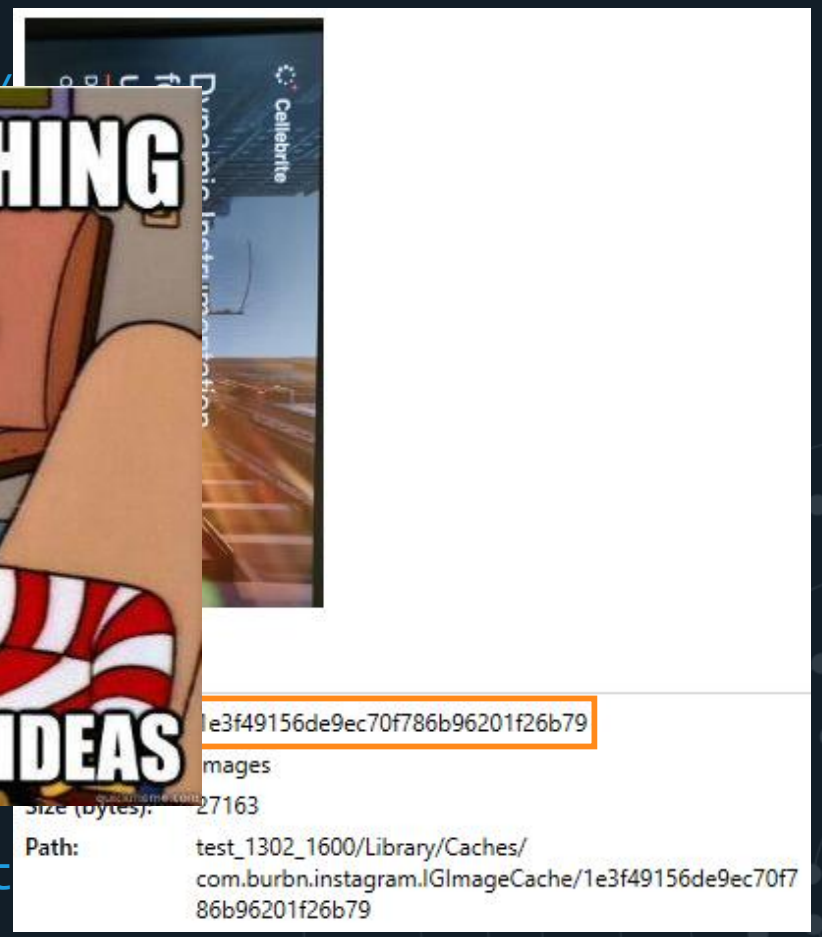
```
>>> from hashlib import md5
>>> md5('https://instagram.ftlv5-1.fna.fbcdn.net/v/t69.10827-
15/sh0.08/e35/p750x750/84206092_180232999904571_4890237386946319692_n.jpg?_nc
_ht=instagram.ftlv5-
1.fna.fbcdn.net&_nc_cat=103&_nc_ohc=rlb46G6x00AAX_tBZI8&oh=3ed9b30ad5ac3b5e7f
a75b2baabb325b&oe=5E47B60B').hexdigest()
'c4e63b6bf7df0ce8d4516fc6160674b3'
>>> md5('https://instagram.ftlv5-1.fna.fbcdn.net/v/t69.10827-
15/e35/p249x249/84206092_180232999904571_4890237386946319692_n.jpg?_nc_ht=ins
tagram.ftlv5-
1.fna.fbcdn.net&_nc_cat=103&_nc_ohc=rlb46G6x00AAX_tBZI8&oh=2434638d5ea0ca81ad
14a6164913b76e&oe=5E47410C').hexdigest()
'9c6c440237826aa3e75a78e4783a64fa'
```

Cellebrite

```
>>> from hashlib import md5
>>> md5('https://instagram.ftlv5-1.fna.fbcdn.net/v/
15/sh0.08/e35/p750x750/84206092_180232999904571_489
_ht=instagram.ftlv5-
1.fna.fbcdn.net&_nc_cat=103&_nc_ohc=rlb46G6x00AAX_t
a75b2baabb325b&oe=5E47B60B').hexdigest()
'c4e63b6bf7df0ce8d4516fc6160674b3'
>>> md5('https://instagram.ftlv5-1.fna.fbcdn.net/v/
15/e35/p249x249/84206092_180232999904571_4890237386
tagram.ftlv5-
1.fna.fbcdn.net&_nc_cat=103&_nc_ohc=rlb46G6x00AAX_t
14a6164913b76e&oe=5E47410C').hexdigest()
'9c6c440237826aa3e75a78e4783a64fa'
```

Cellebrite

```
>>> from hashlib import md5
>>> md5('https://instagra
15/sh0.08/e35/p750x750/84
_ht=instagram.ftlv5-
1.fna.fbcdn.net&_nc_cat=1
a75b2baabb325b&oe=5E47B60
'c4e63b6bf7df0ce8d4516fc6
>>> md5('https://instagra
15/e35/p249x249/84206092_
tagram.ftlv5-
1.fna.fbcdn.net&_nc_cat=103&_nc_ohc=rlb46G6x00AAX_t
14a6164913b76e&oe=5E47410C').hexdigest()
'9c6c440237826aa3e75a78e4783a64fa'
```



Cellebrite

If only we could have X-Ray vision into what the app is doing

Instagram

?

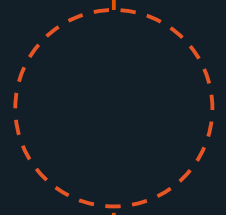
MD5

1e3f49156de9ec70
f786b96201f26b79

Agenda

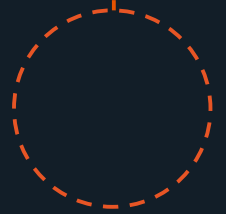
Overview of mobile app decoding

And some technical challenges in it



Introduction to dynamic instrumentation

And how it can solve those challenges



Hands-on work with FRIDA

Solving more hard problems

Cellebrite

Cellebrite

Dynamic Instrumentation

Cellebrite

Instrumentation (Software)

> The ability to monitor or measure the level of a product's performance, to diagnose errors, and to write trace information

> Instrumentation approaches can be of two types: source instrumentation and binary instrumentation

Wikipedia


Cellebrite
FRIDA

Cellebrite

Frida

> A dynamic code instrumentation toolkit. It lets you inject snippets of JavaScript or your own library into native apps on Windows, macOS, GNU/Linux, iOS, Android, and QNX. Frida also provides you with some simple tools built on top of the Frida API.

frida.re/docs

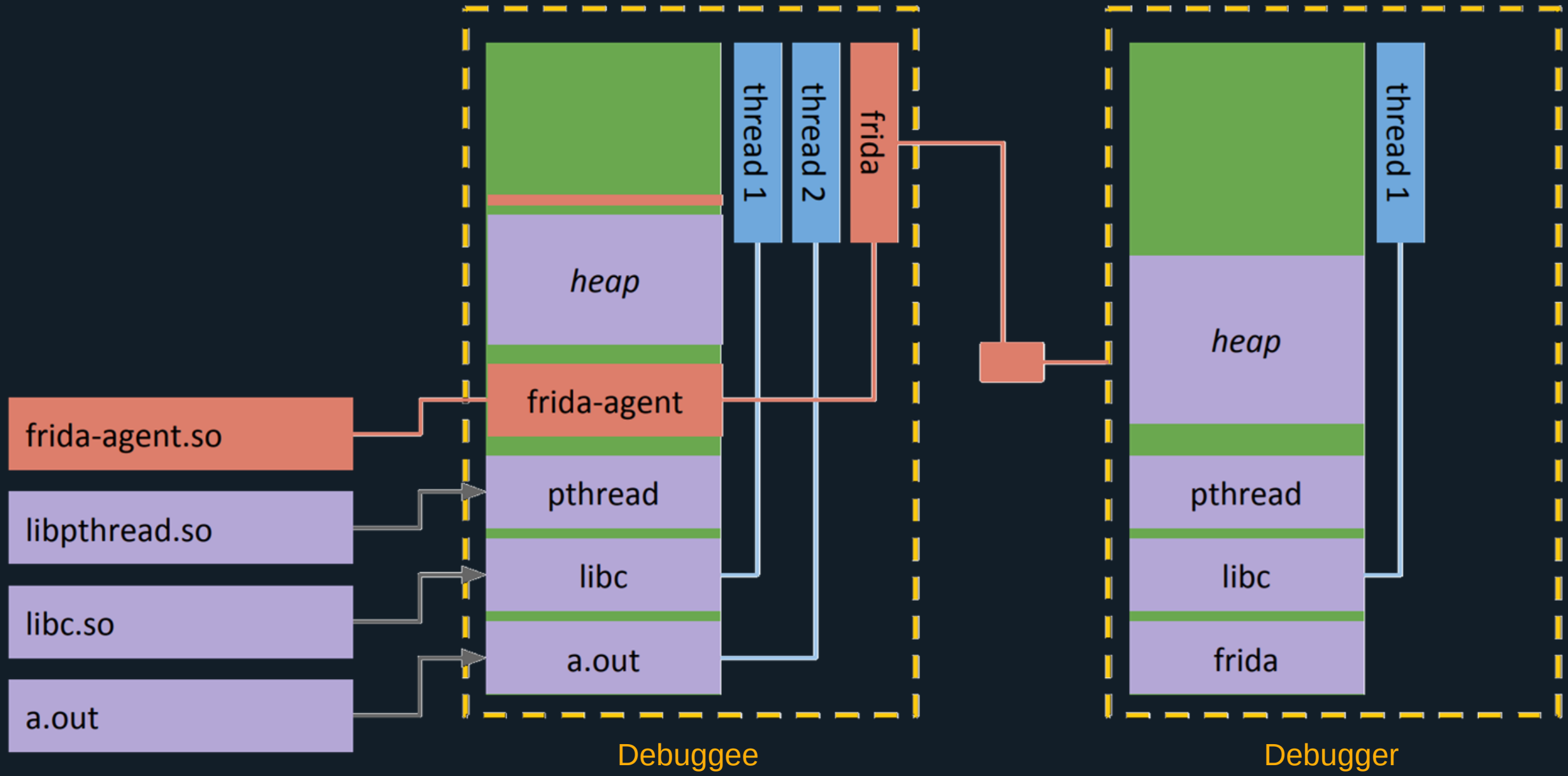
Cellebrite

Built by this guy

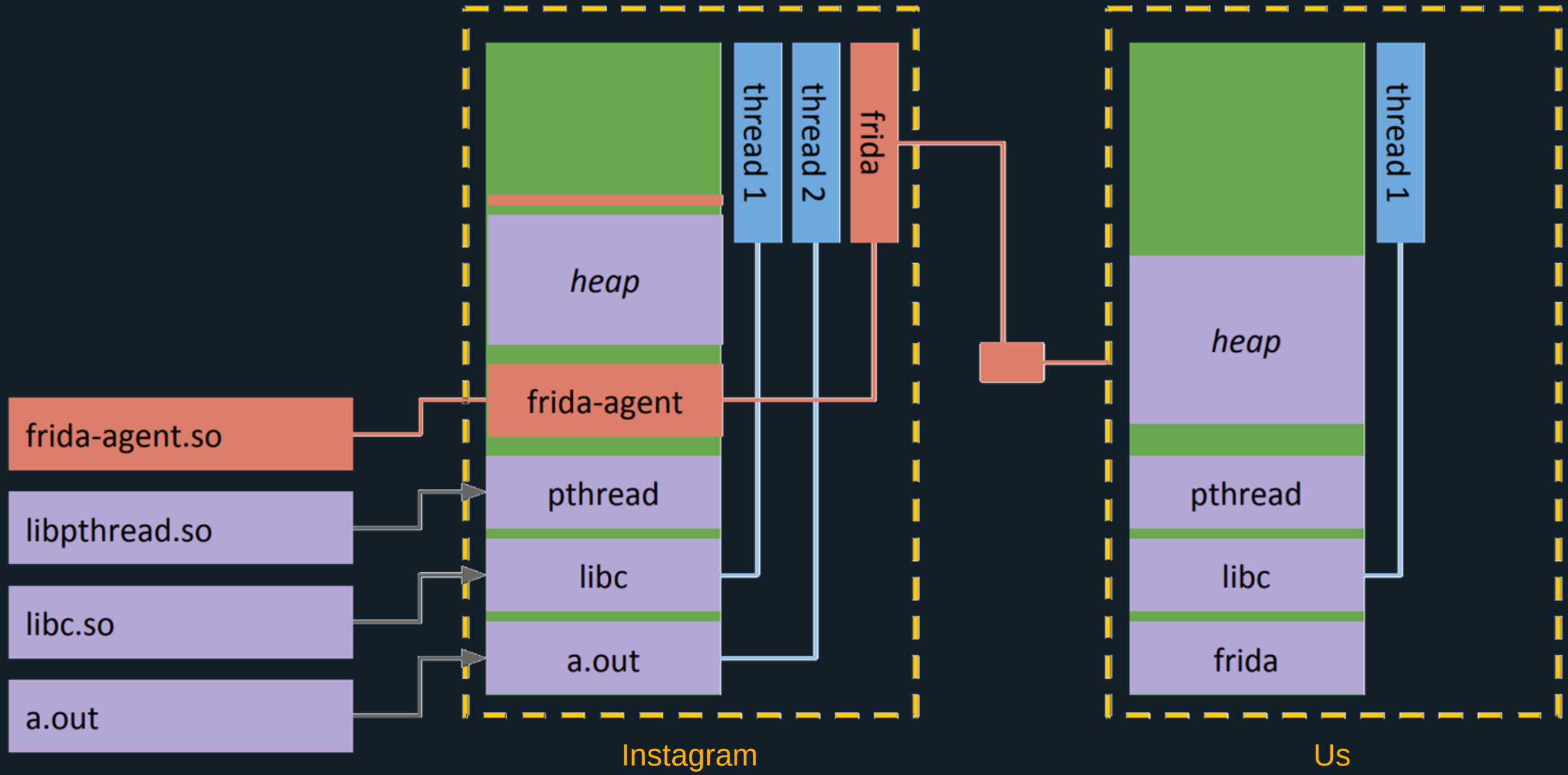
How does it work?

Injects Google’s V8 Javascript engine into a process

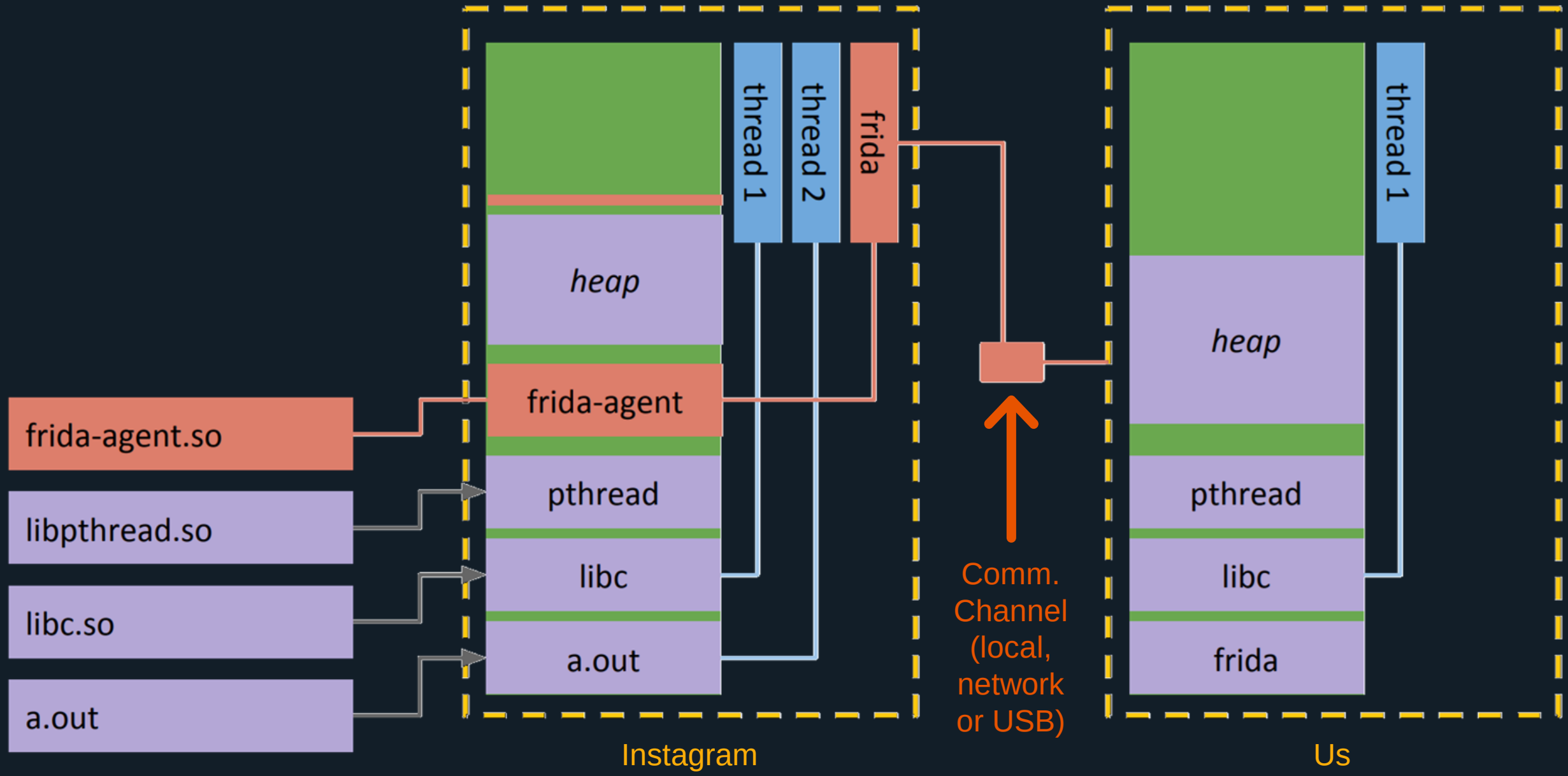
Creates a two-way communication channel with the JS engine



(Source: https://frida.re/slides/osdc-2015-the-engineering-behind-the-reverse-engineering.pdf)



(Source: https://frida.re/slides/osdc-2015-the-engineering-behind-the-reverse-engineering.pdf)



Cellebrite

(Source: https://frida.re/slides/osdc-2015-the-engineering-behind-the-reverse-engineering.pdf)

Cellebrite

What can we do with it?

Cellebrite

What can we do with it?

Hooks!

Cellebrite

What can we do with it?

Hooks!

And also:

Attaching to processes

Modifying function arguments

Calling functions

Inspecting and modifying memory

Cellebrite

Hooking

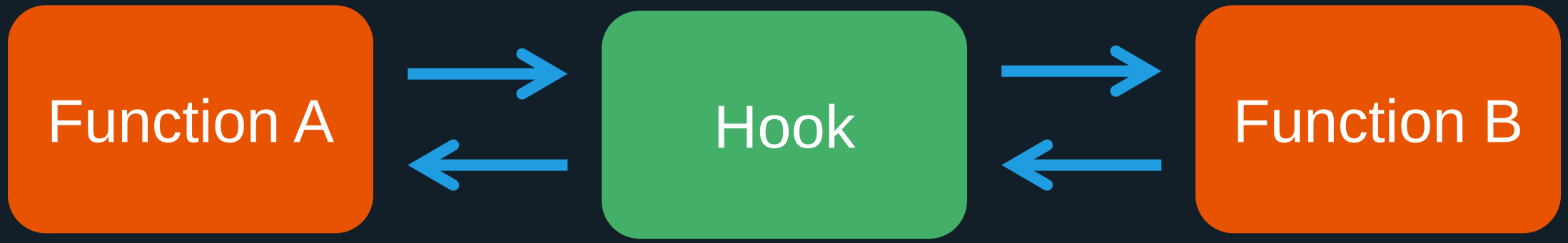
> Altering or augmenting the behaviour of software components by intercepting function calls, messages or events passed between software components

> Code that handles such intercepted function calls, events or messages is called a hook

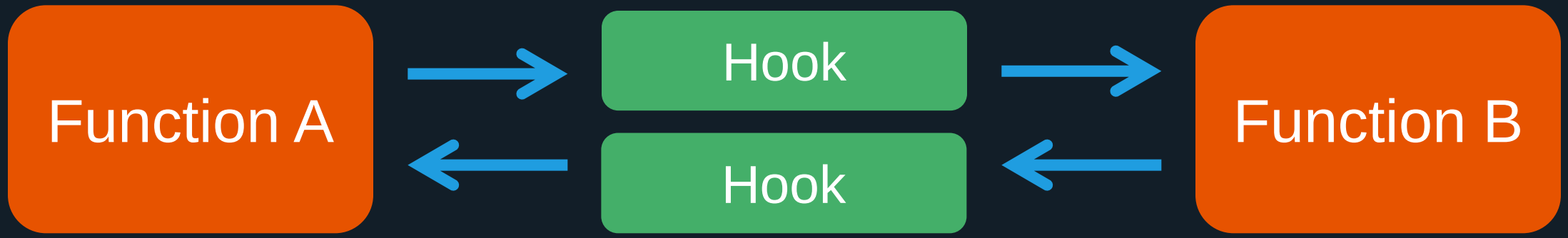
Wikipedia

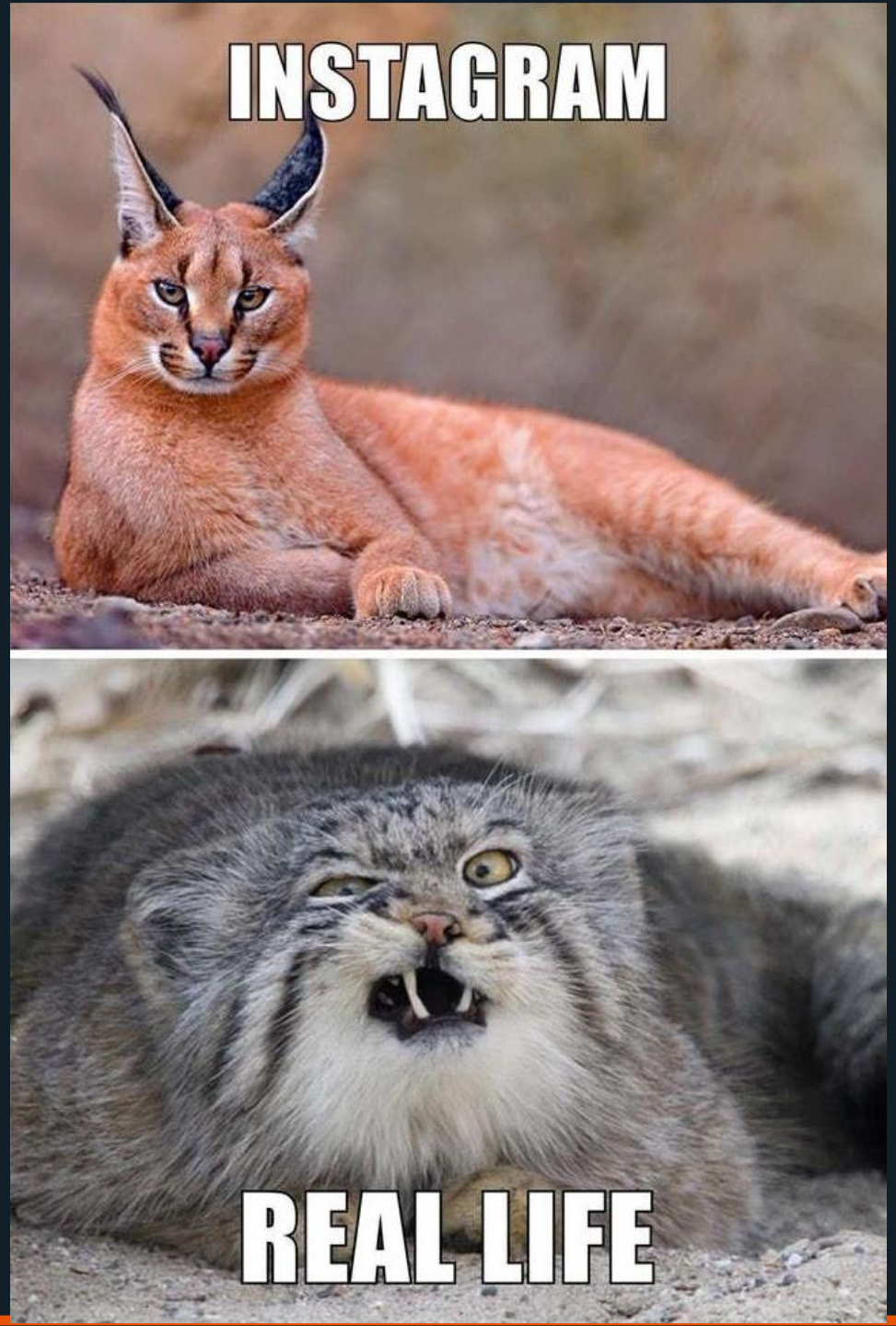
Hooking - Abstraction

Hooking - Abstraction

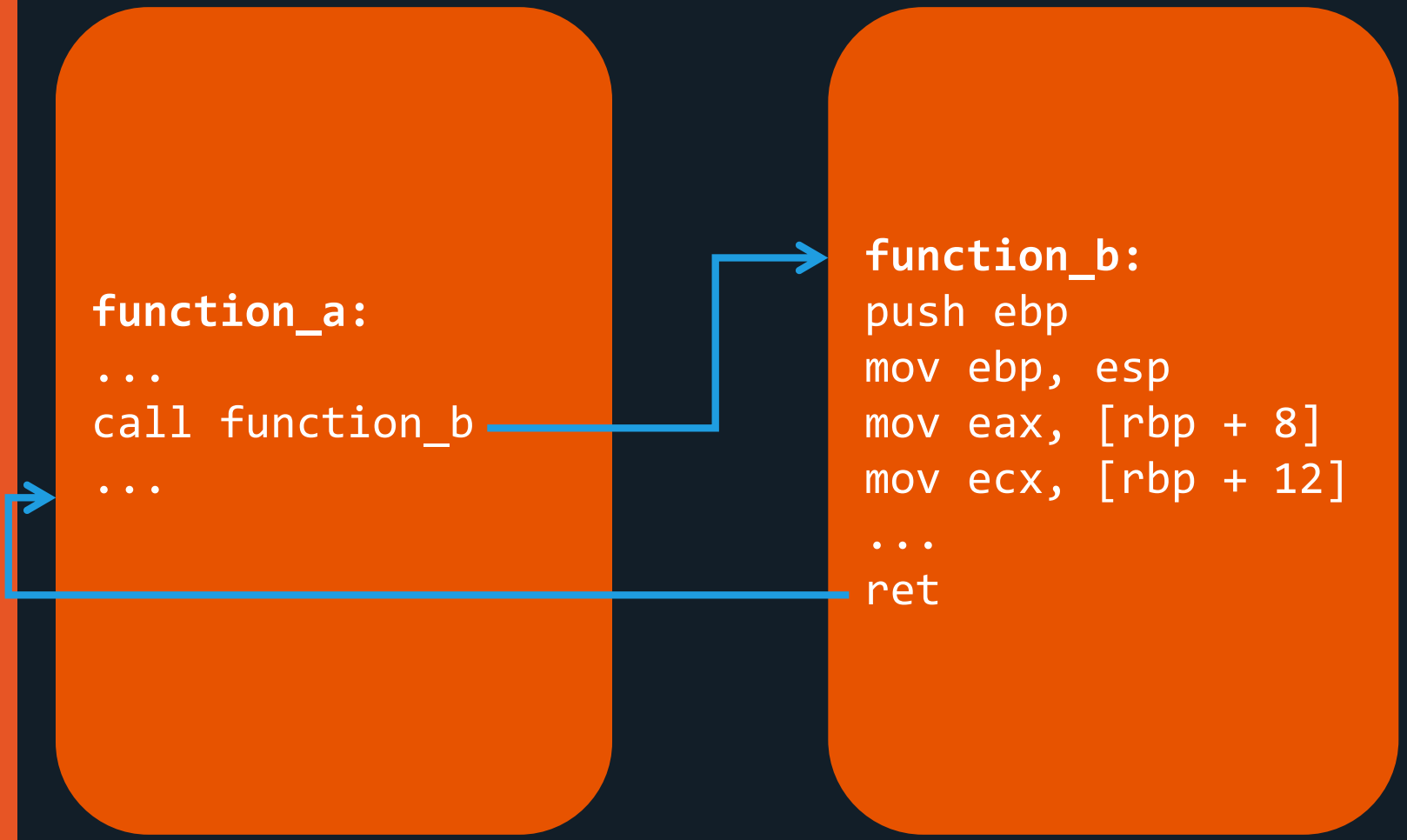


Hooking - Abstraction

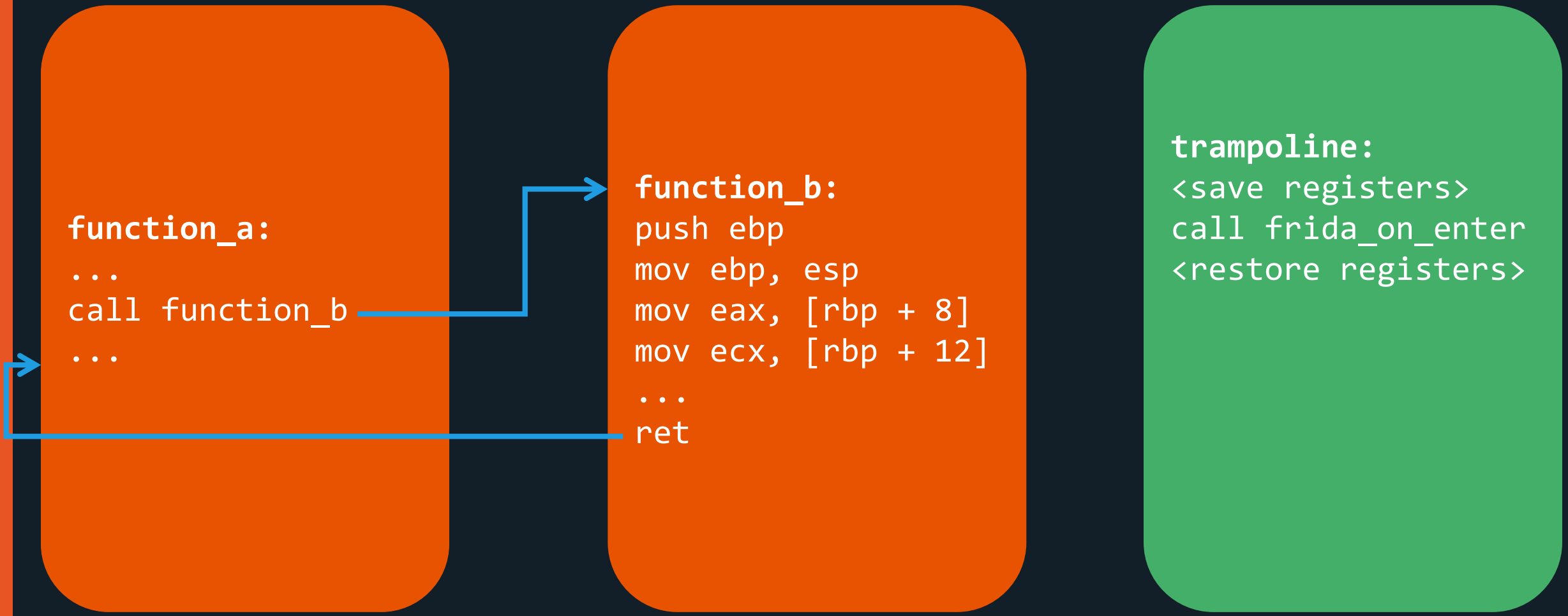



INSTAGRAM
REAL LIFE

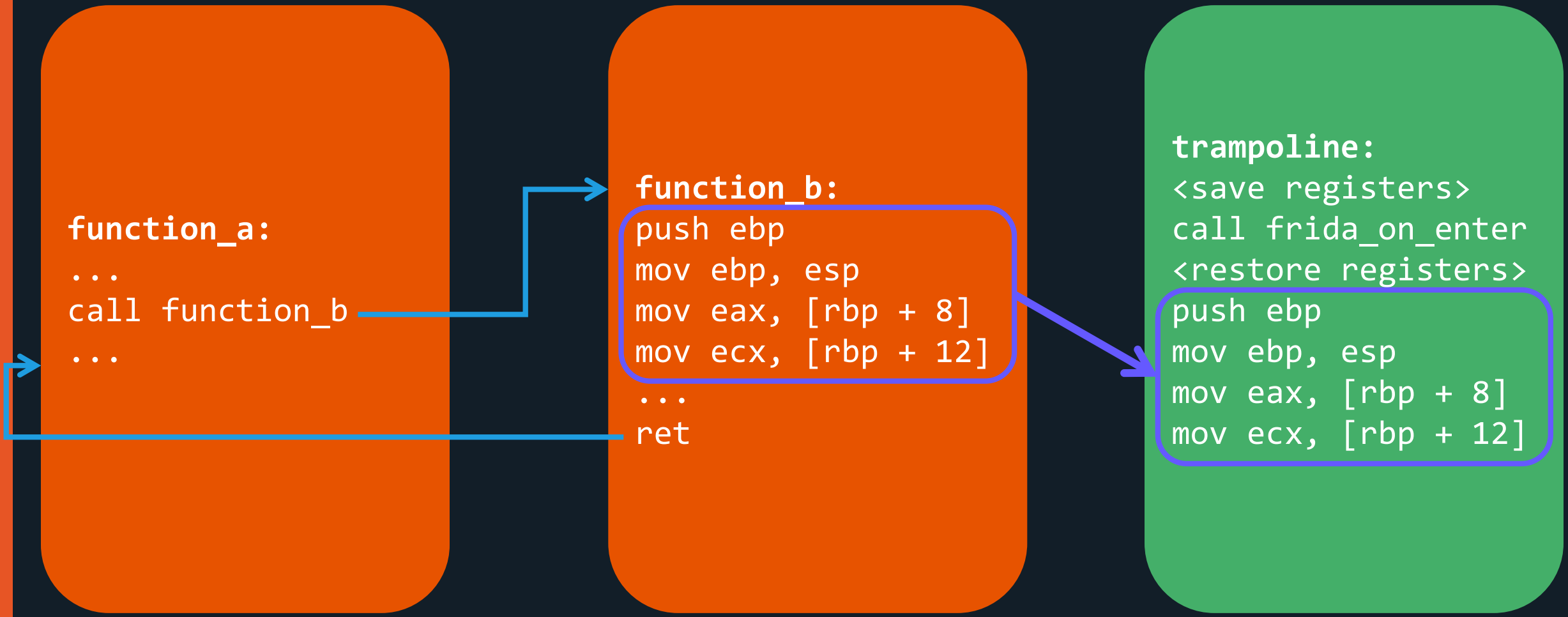
Hooking – In real life



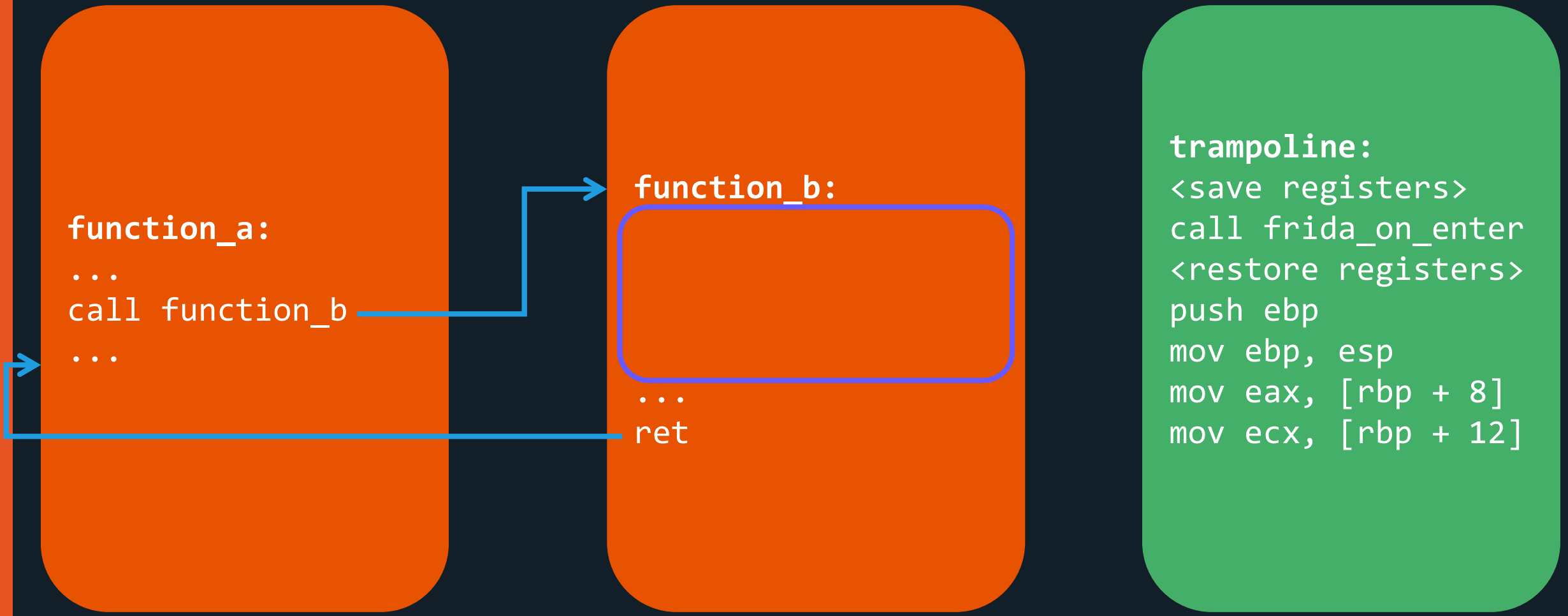
Hooking – In real life



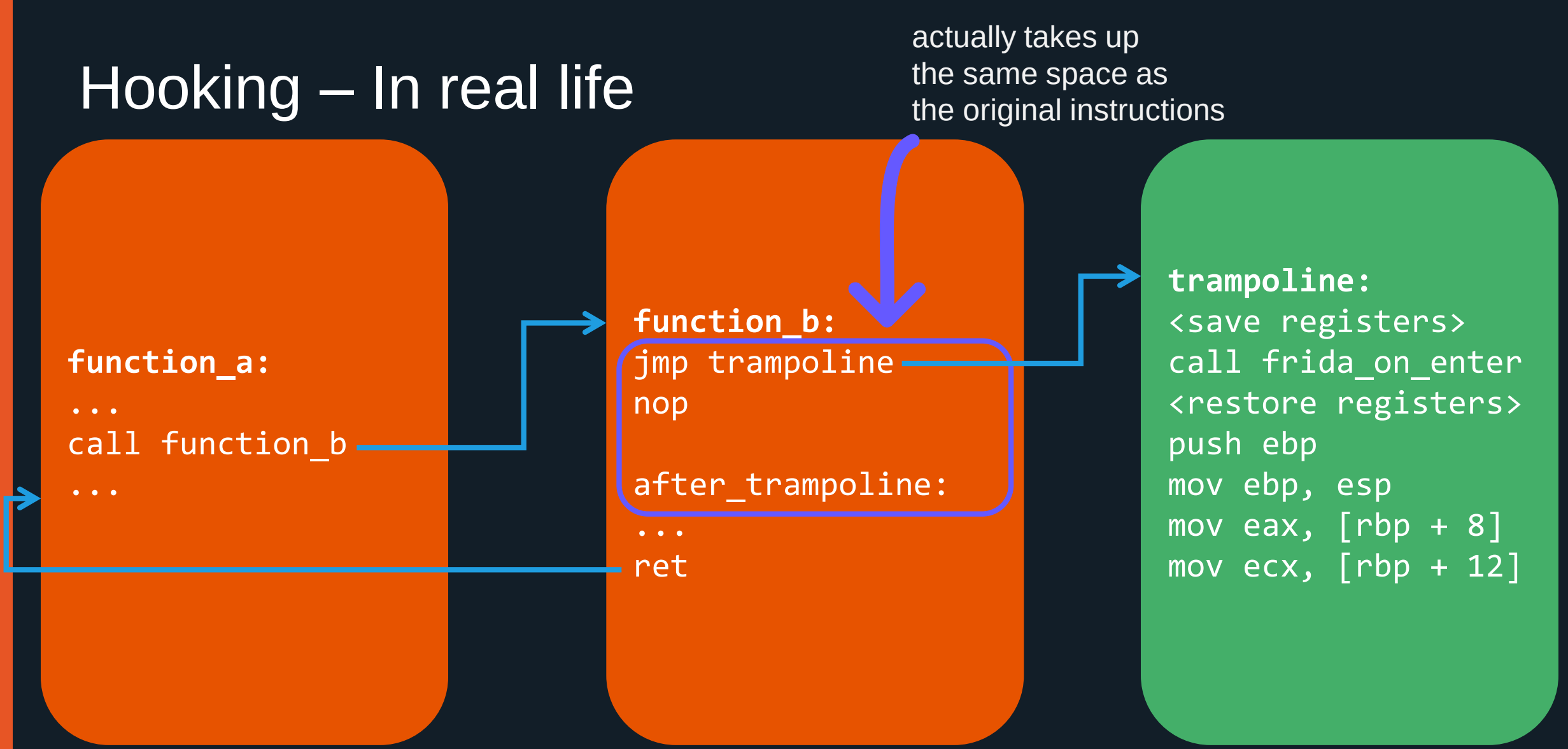
Hooking – In real life



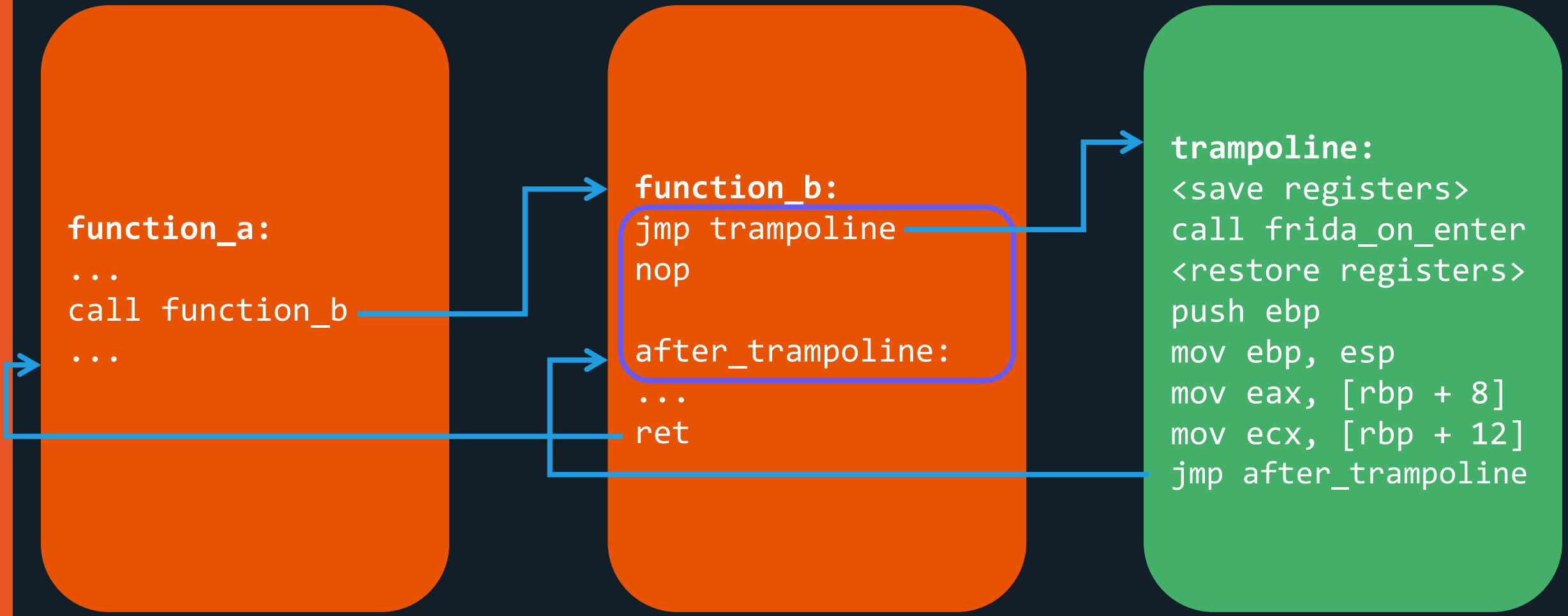
Hooking – In real life



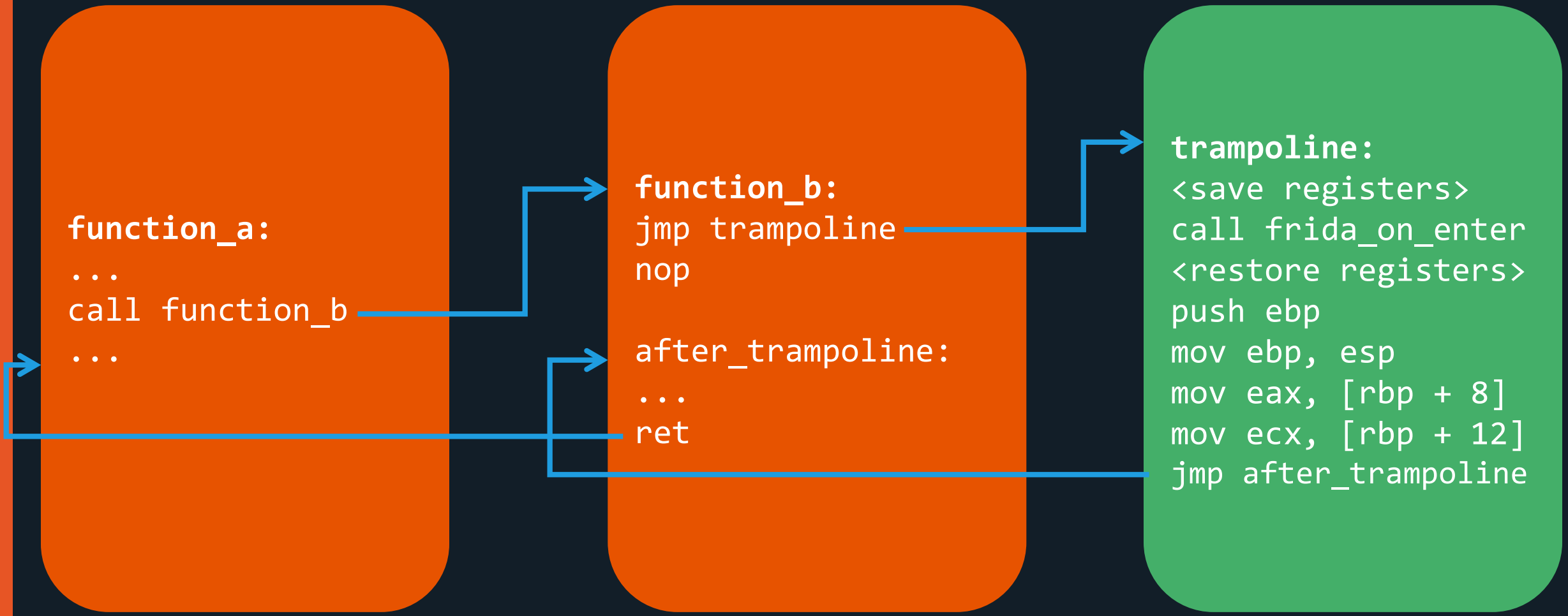
Hooking – In real life



Hooking – In real life



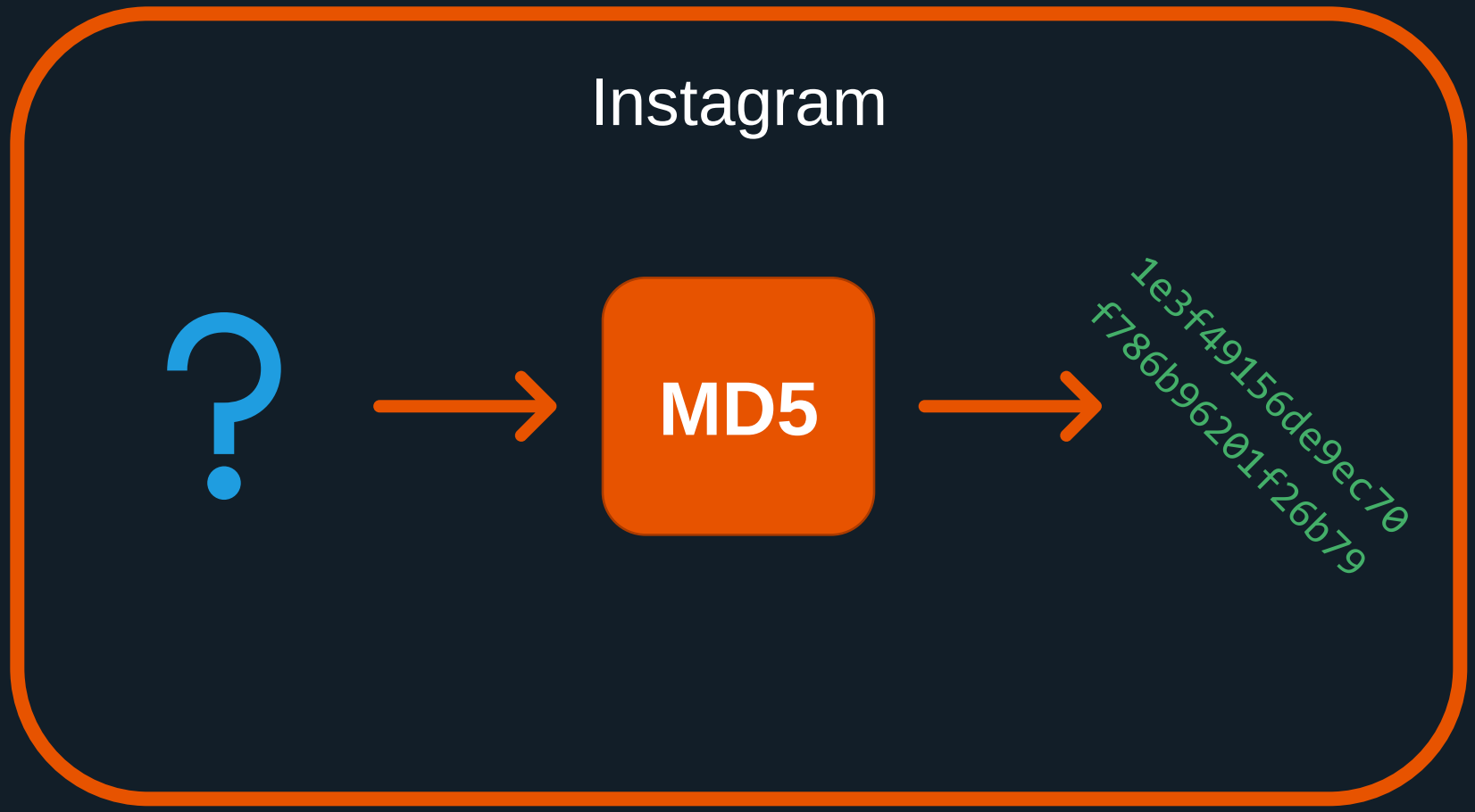
Hooking – In real life



We can do the same trick for returning from **function_b**

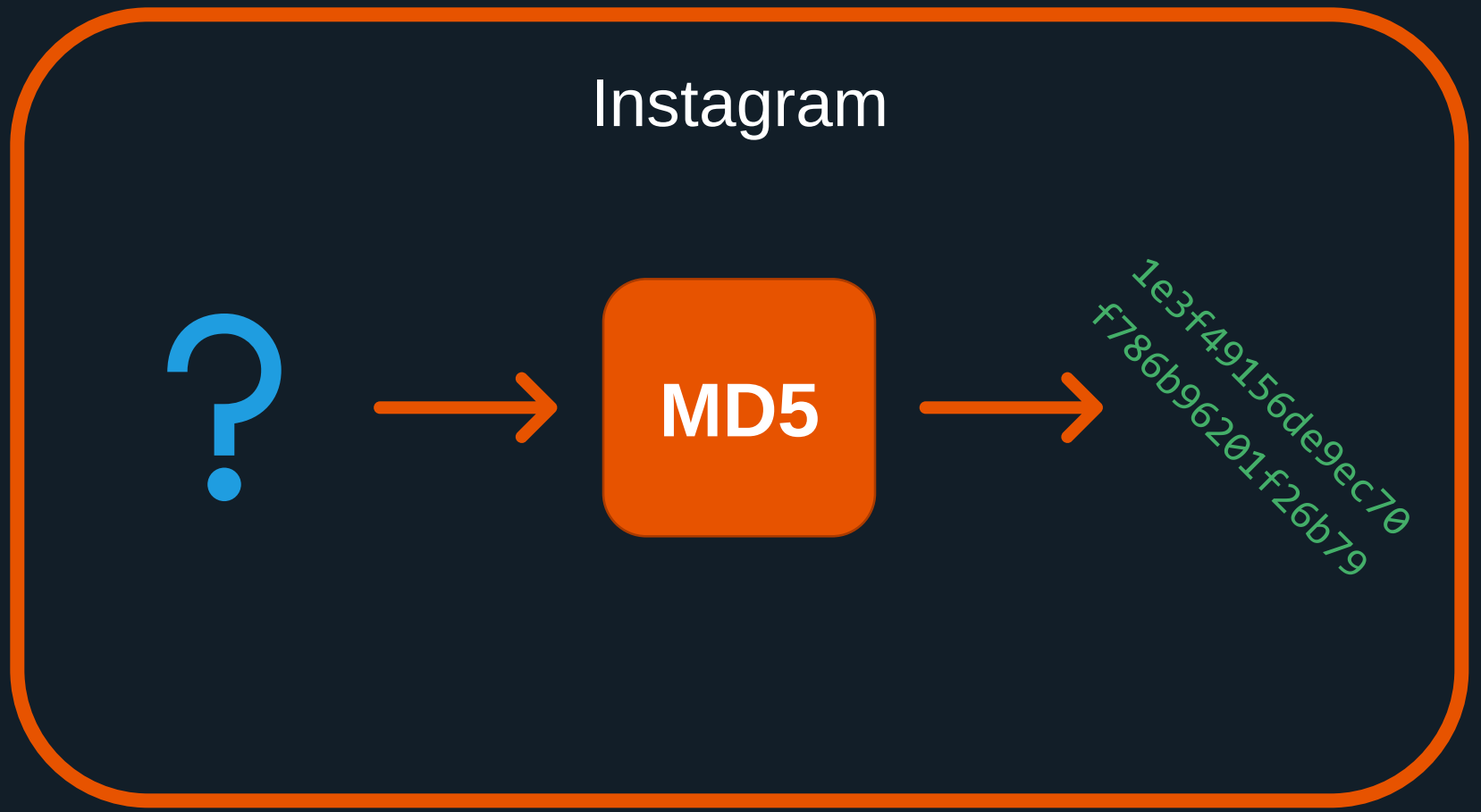
Cellebrite

If only we could have
X-Ray vision
into what the app is doing…



If only we could have
X-Ray vision
into what the app is doing…

Now we do!




Cellebrite
Using FRIDA

Ways to use Frida

Frida CLI

Interactive JavaScript shell for prototyping and debugging

Frida Tools

Set of command line tools that wrap Frida's API

Frida API

With bindings in Python, C, Node.js, .NET, Swift and more

Cellebrite

Frida requires root!

iOS – Jailbreak

Android – root / emulator

Installing FRIDA

Using the python installer

Install Frida API

```
pip install frida
```

Install Frida Tools

```
pip install frida-tools
```

Upload frida-server to your device

Download from github.com/frida/frida/releases

(or use Cydia on iOS)

Cellebrite

frida-server

Installing on Android

```
$ adb root # might be required
$ adb push frida-server /data/local/tmp/
$ adb shell "chmod 755 /data/local/tmp/frida-server“
$ adb shell "/data/local/tmp/frida-server &“
```

Cellebrite

Frida Tools

The basics

Cellebrite

frida-ls-devices

List devices

```
C:\Users\orb
λ frida-ls-devices
Id                                        Type    Name
----------------------------------------  ------  ------------
local                                     local   Local System
c3833d471c07461e9d72b4869cd18f032e3831e0  usb     Apple iPhone
tcp                                       remote  Local TCP
```

Cellebrite

frida-ps

List processes

-U for USB

```
C:\Users\orb
λ frida-ps -U
 PID  Name
----  ------------------------------------------------------
1786  Cydia
3252  Instagram
1768  Mail
1772  Settings
2731  AppPredictionWidget
  76  AppleCredentialManagerDaemon
 182  AssetCacheLocatorService
 108  BlueTool
 185  CMFSyncAgent
1820  CacheDeleteAppContainerCaches
2961  CacheDeleteDaily
  97  CloudKeychainProxy
  84  CommCenter
 206  ContainerMetadataExtractor
 104  ContextService
1880  EscrowSecurityAlert
 202  HeuristicInterpreter
 127  IMDPersistenceAgent
  45  IOMFB_bics_daemon
1769  LocalStorageFileProvider
1817  MTLCompilerService
1774  MTLCompilerService
1773  MTLCompilerService
1714  MTLCompilerService
 137  MTLCompilerService
 136  MTLCompilerService
1825  MailCacheDeleteExtension
2734  MapsWidget
1821  MobileBackupCacheDeleteService
 110  MobileGestaltHelper
 125  MobileStorageMounter
 103  OTATaskingAgent
3012  ProtectedCloudKeySyncing
```

Cellebrite

frida-ps

List processes

-U for USB

-a for applications only

```
C:\Users\orb
λ frida-ps -U -a
 PID  Name       Identifier
----  ---------  --------------------
1786  Cydia      com.saurik.Cydia
3252  Instagram  com.burbn.instagram
1768  Mail       com.apple.mobilemail
1772  Settings   com.apple.Preferences
```

Cellebrite

frida-ps

List processes

-U USB

-a Applications only

-i All installed apps

```
C:\Users\orb
λ frida-ps -U -a -i
 PID  Name          Identifier
----  ------------  ----------------------------
1786  Cydia         com.saurik.Cydia
3252  Instagram     com.burbn.instagram
1768  Mail          com.apple.mobilemail
1772  Settings      com.apple.Preferences
   -  App Store     com.apple.AppStore
   -  Books         com.apple.iBooks
   -  Calculator    com.apple.calculator
   -  Calendar      com.apple.mobilecal
   -  Camera        com.apple.camera
   -  Clock         com.apple.mobiletimer
   -  Compass       com.apple.compass
   -  Contacts      com.apple.MobileAddressBook
   -  FaceTime      com.apple.facetime
   -  Files         com.apple.DocumentsApp
   -  Find Friends  com.apple.mobileme.fmf1
   -  Find iPhone   com.apple.mobileme.fmip1
   -  Health        com.apple.Health
   -  Home          com.apple.Home
   -  Maps          com.apple.Maps
   -  Measure       com.apple.measure
   -  Messages      com.apple.MobileSMS
   -  Music         com.apple.Music
   -  Notes         com.apple.mobilenotes
   -  Phone         com.apple.mobilephone
   -  Photos        com.apple.mobileslideshow
   -  Podcasts      com.apple.podcasts
   -  Reminders     com.apple.reminders
   -  Safari        com.apple.mobilesafari
   -  Stocks        com.apple.stocks
   -  TV            com.apple.tv
   -  Tips          com.apple.tips
   -  Voice Memos   com.apple.VoiceMemos
   -  Wallet        com.apple.Passbook
```

Cellebrite

Frida CLI

Exploring the Instagram process

Cellebrite

Frida CLI

Attach to the Instagram process in the connected USB device

```
C:\Users\orb
λ frida -U Instagram
     ____
    / _  |   Frida 12.8.10 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Waiting for USB device to appear...

[Apple iPhone::Instagram]->
```

Cellebrite

Frida CLI

Enumerate modules in process

```
C:\Users\orb
λ frida -U Instagram
     ____
    / _  |   Frida 12.8.10 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Waiting for USB device to appear...

[Apple iPhone::Instagram]->
[Apple iPhone::Instagram]-> var allModules = Process.enumerateModules();
```

Cellebrite

Frida CLI

Use JavaScript to filter results

```
C:\Users\orb
λ frida -U Instagram

    ____
   / _  |   Frida 12.8.10 - A world-class dynamic instrumentation toolkit
  | (_| |
   > _  |   Commands:
  /_/ |_|       help      -> Displays the help system
  . . . .       object?   -> Display information about 'object'
  . . . .       exit/quit -> Exit
  . . . .
  . . . .   More info at https://www.frida.re/docs/home/
Waiting for USB device to appear...

[Apple iPhone::Instagram]->
[Apple iPhone::Instagram]-> var allModules = Process.enumerateModules();
[Apple iPhone::Instagram]-> var cryptoModules = allModules.filter(function(m) {
return m.name.toLowerCase().includes("crypt") });
```

Cellebrite

Frida CLI

Print results to the console

```
C:\Users\orb
λ frida -U Instagram

     ____
    / _  |   Frida 12.8.10 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Waiting for USB device to appear...

[Apple iPhone::Instagram]->
[Apple iPhone::Instagram]-> var allModules = Process.enumerateModules();
[Apple iPhone::Instagram]-> var cryptoModules = allModules.filter(function(m) {
return m.name.toLowerCase().includes("crypt") });
[Apple iPhone::Instagram]-> cryptoModules
[
    {
        "base": "0x1e00f9000",
        "name": "libcorecrypto.dylib",
        "path": "/usr/lib/system/libcorecrypto.dylib",
        "size": 409600
    },
    {
        "base": "0x1e00df000",
        "name": "libcommonCrypto.dylib",
        "path": "/usr/lib/system/libcommonCrypto.dylib",
        "size": 49152
    },
    {
        "base": "0x2038d0000",
        "name": "CryptoTokenKit",
        "path": "/System/Library/PrivateFrameworks/CryptoTokenKit.framework/Crypt
oTokenKit",
        "size": 200704
    }
]
```

Cellebrite

Frida CLI

Find functions that have “md5” in their names

```
[Apple iPhone::Instagram]-> for (i in cryptoModules) {
  var allSymbols = cryptoModules[i].enumerateSymbols();
  var md5Symbols = allSymbols.filter(function(s) {
    return s.name.toLowerCase().includes("md5")
  });
  for (j in md5Symbols) {
    console.log(cryptoModules[i].name + " -> " + md5Symbols[j].name);
  }
 }

libcorecrypto.dylib -> ccmd5_di
libcorecrypto.dylib -> ccmd5_ltc_di
libcommonCrypto.dylib -> CC_MD5
libcommonCrypto.dylib -> CC_MD5_Final
libcommonCrypto.dylib -> CC_MD5_Init
libcommonCrypto.dylib -> CC_MD5_Update
libcommonCrypto.dylib -> MD5Final
libcommonCrypto.dylib -> ccmd5_di
[Apple iPhone::Instagram]->
```

Cellebrite

Frida CLI

Find functions that have “md5” in their names

```
[Apple iPhone::Instagram]-> for (i in cryptoModules) {
  var allSymbols = cryptoModules[i].enumerateSymbols();
  var md5Symbols = allSymbols.filter(function(s) {
    return s.name.toLowerCase().includes("md5")
  });
  for (j in md5Symbols) {
    console.log(cryptoModules[i].name + " -> " + md5Symbols[j].name);
  }
 }

libcorecrypto.dylib -> ccmd5_di
libcorecrypto.dylib -> ccmd5_ltc_di
libcommonCrypto.dylib -> CC_MD5
libcommonCrypto.dylib -> CC_MD5_Final
libcommonCrypto.dylib -> CC_MD5_Init
libcommonCrypto.dylib -> CC_MD5_Update
libcommonCrypto.dylib -> MD5Final
libcommonCrypto.dylib -> ccmd5_di
[Apple iPhone::Instagram]->
```

Celle

```
[Apple iPhone::Instagram]-> for (i in cryptoModules) {
```

stackoverflow Products Customers Use cases Search...

Home

PUBLIC

How can I convert a String to an MD5 hash in iOS using Swift?

Asked 4 years, 6 months ago Active 3 months ago Viewed 73k times

```
.name);
```

```
libcorecrypto.dylib -> ccmd5_di
libcorecrypto.dylib -> ccmd5_ltc_di
libcommonCrypto.dylib -> CC_MD5
libcommonCrypto.dylib -> CC_MD5_Final
libcommonCrypto.dylib -> CC_MD5_Init
libcommonCrypto.dylib -> CC_MD5_Update
libcommonCrypto.dylib -> MD5Final
libcommonCrypto.dylib -> ccmd5_di
[Apple iPhone::Instagram]->
```

Frida CLI

Internet to the rescue!

111

This is the category I use:

NSString+MD5.m

```
#import <CommonCrypto/CommonDigest.h>

@implementation NSString (MD5)

- (NSString *)MD5String {
    const char *cStr = [self UTF8String];
    unsigned char result[CC_MD5_DIGEST_LENGTH];
    CC_MD5( cStr, (CC_LONG)strlen(cStr), result );
```

Celle

```
[Apple iPhone::Instagram]-> for (i in cryptoModules) {
```

stackoverflow Products Customers Use cases Search...

Home

PUBLIC

How can I convert a String to an MD5 hash in iOS using Swift?

Asked 4 years, 6 months ago Active 3 months ago Viewed 73k times

```
.name);
```

Frida CLI

```
libcorecrypto.dylib -> ccmd5_di
libcorecrypto.dylib -> ccmd5_ltc_di
libcommonCrypto.dylib -> CC_MD5
libcommonCrypto.dylib -> CC_MD5_Final
libcommonCrypto.dylib -> CC_MD5_Init
libcommonCrypto.dylib -> CC_MD5_Update
libcommonCrypto.dylib -> MD5Final
libcommonCrypto.dylib -> ccmd5_di
[Apple iPhone::Instagram]->
```

Internet to the rescue!

111

This is the category I use:

NSString+MD5.m

```
#import <CommonCrypto/CommonDigest.h>

@implementation NSString (MD5)

- (NSString *)MD5String {
    const char *cStr = [self UTF8String];
    unsigned char result[CC_MD5_DIGEST_LENGTH];
    CC_MD5( cStr, (CC_LONG)strlen(cStr), result );
```

Cellebrite

Frida CLI

Instrument!

```
C:\Users\orb
λ frida -U Instagram

     ____
    / _  |   Frida 12.8.10 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Waiting for USB device to appear...

[Apple iPhone::Instagram]-> var commonCrypto = Process.getModuleByName("libcommonCrypto
.dylib")
[Apple iPhone::Instagram]-> var cc_md5_addr = commonCrypto.getExportByName("CC_MD5")
[Apple iPhone::Instagram]-> cc_md5_addr
"0x1e00e3c80"
```

Cellebrite

Frida CLI

Instrument!

```
C:\Users\orb
λ frida -U Instagram

     ____
    / _  |   Frida 12.8.10 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Waiting for USB device to appear...

[Apple iPhone::Instagram]-> var commonCrypto = Process.getModuleByName("libcommonCrypto
.dylib")
[Apple iPhone::Instagram]-> var cc_md5_addr = commonCrypto.getExportByName("CC_MD5")
[Apple iPhone::Instagram]-> cc_md5_addr
"0x1e00e3c80"
[Apple iPhone::Instagram]-> var cc_md5 = new NativeFunction(cc_md5_addr, 'pointer', ['p
ointer', 'int', 'pointer'])
```

Cellebrite

Frida CLI

Instrument!

```
C:\Users\orb
λ frida -U Instagram

     ____
    / _  |   Frida 12.8.10 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Waiting for USB device to appear...

[Apple iPhone::Instagram]-> var commonCrypto = Process.getModuleByName("libcommonCrypto
.dylib")
[Apple iPhone::Instagram]-> var cc_md5_addr = commonCrypto.getExportByName("CC_MD5")
[Apple iPhone::Instagram]-> cc_md5_addr
"0x1e00e3c80"
[Apple iPhone::Instagram]-> var cc_md5 = new NativeFunction(cc_md5_addr, 'pointer', ['p
ointer', 'int', 'pointer'])
[Apple iPhone::Instagram]-> var hashResult = Memory.alloc(16)
[Apple iPhone::Instagram]-> var s = Memory.allocUtf8String("12345678")
[Apple iPhone::Instagram]-> cc_md5(s, 8, hashResult)
"0x10ea2df40"
```

Cellebrite

Frida CLI

Instrument!

```
C:\Users\orb
λ frida -U Instagram

     ____
    / _  |   Frida 12.8.10 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Waiting for USB device to appear...

[Apple iPhone::Instagram]-> var commonCrypto = Process.getModuleByName("libcommonCrypto
.dylib")
[Apple iPhone::Instagram]-> var cc_md5_addr = commonCrypto.getExportByName("CC_MD5")
[Apple iPhone::Instagram]-> cc_md5_addr
"0x1e00e3c80"
[Apple iPhone::Instagram]-> var cc_md5 = new NativeFunction(cc_md5_addr, 'pointer', ['p
ointer', 'int', 'pointer'])
[Apple iPhone::Instagram]-> var hashResult = Memory.alloc(16)
[Apple iPhone::Instagram]-> var s = Memory.allocUtf8String("12345678")
[Apple iPhone::Instagram]-> cc_md5(s, 8, hashResult)
"0x10ea2df40"
[Apple iPhone::Instagram]-> hashResult
"0x10ea2df40"
```

Cellebrite

Frida CLI

Instrument!

```
C:\Users\orb
λ frida -U Instagram

     ____
    / _  |   Frida 12.8.10 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Waiting for USB device to appear...

[Apple iPhone::Instagram]-> var commonCrypto = Process.getModuleByName("libcommonCrypto
.dylib")
[Apple iPhone::Instagram]-> var cc_md5_addr = commonCrypto.getExportByName("CC_MD5")
[Apple iPhone::Instagram]-> cc_md5_addr
"0x1e00e3c80"
[Apple iPhone::Instagram]-> var cc_md5 = new NativeFunction(cc_md5_addr, 'pointer', ['p
ointer', 'int', 'pointer'])
[Apple iPhone::Instagram]-> var hashResult = Memory.alloc(16)
[Apple iPhone::Instagram]-> var s = Memory.allocUtf8String("12345678")
[Apple iPhone::Instagram]-> cc_md5(s, 8, hashResult)
"0x10ea2df40"
[Apple iPhone::Instagram]-> hashResult
"0x10ea2df40"
[Apple iPhone::Instagram]-> hashResult.readByteArray(16)
0000  25 d5 5a d2 83 aa 40 0a f4 64 c7 6d 71 3c 07 ad   %.Z...@..d.mq<..
[Apple iPhone::Instagram]->
```

Cellebrite

Frida CLI

Instrument!

```
>>> from hashlib import md5
>>> md5("12345678").hexdigest()
'25d55ad283aa400af464c76d713c07ad'
>>>
```

```
C:\Users\orb
λ frida -U Instagram

     ____
    / _  |   Frida 12.8.10 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Waiting for USB device to appear...

[Apple iPhone::Instagram]-> var commonCrypto = Process.getModuleByName("libcommonCrypto
.dylib")
[Apple iPhone::Instagram]-> var cc_md5_addr = commonCrypto.getExportByName("CC_MD5")
[Apple iPhone::Instagram]-> cc_md5_addr
"0x1e00e3c80"
[Apple iPhone::Instagram]-> var cc_md5 = new NativeFunction(cc_md5_addr, 'pointer', ['p
ointer', 'int', 'pointer'])
[Apple iPhone::Instagram]-> var hashResult = Memory.alloc(16)
[Apple iPhone::Instagram]-> var s = Memory.allocUtf8String("12345678")
[Apple iPhone::Instagram]-> cc_md5(s, 8, hashResult)
"0x10ea2df40"
[Apple iPhone::Instagram]-> hashResult
"0x10ea2df40"
[Apple iPhone::Instagram]-> hashResult.readByteArray(16)
0000  25 d5 5a d2 83 aa 40 0a f4 64 c7 6d 71 3c 07 ad   %.Z...@..d.mq<..
[Apple iPhone::Instagram]->
```

Cellebrite

Frida Tools

Making shortcuts

Cellebrite

frida-trace

Main tool to create hooks

Cellebrite

frida-trace

Main tool to create hooks

-U USB

-i Function(s) to trace

target Process to trace

tons of other options

-f Spawn application

```
C:\Users\orb
λ frida-trace --help
Usage: frida-trace [options] target

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -D ID, --device=ID    connect to device with the given ID
  -U, --usb             connect to USB device
  -R, --remote          connect to remote frida-server
  -H HOST, --host=HOST  connect to remote frida-server on HOST
  -f FILE, --file=FILE  spawn FILE
  -F, --attach-frontmost
                        attach to frontmost application
  -n NAME, --attach-name=NAME
                        attach to NAME
  -p PID, --attach-pid=PID
                        attach to PID
  --stdio=inherit|pipe  stdio behavior when spawning (defaults to "inherit")
  --runtime=duk|v8      script runtime to use (defaults to "duk")
  --debug               enable the Node.js compatible script debugger
  -O FILE, --options-file=FILE
                        text file containing additional command line options
  -I MODULE, --include-module=MODULE
                        include MODULE
  -X MODULE, --exclude-module=MODULE
                        exclude MODULE
  -i FUNCTION, --include=FUNCTION
                        include FUNCTION
  -x FUNCTION, --exclude=FUNCTION
                        exclude FUNCTION
  -a MODULE!OFFSET, --add=MODULE!OFFSET
                        add MODULE!OFFSET
  -T, --include-imports
                        include program's imports
  -t MODULE, --include-module-imports=MODULE
                        include MODULE imports
  -m OBJC_METHOD, --include-objc-method=OBJC_METHOD
                        include OBJC_METHOD
  -M OBJC_METHOD, --exclude-objc-method=OBJC_METHOD
                        exclude OBJC_METHOD
  -s DEBUG_SYMBOL, --include-debug-symbol=DEBUG_SYMBOL
                        include DEBUG_SYMBOL
  -q, --quiet           do not format output messages
  -d, --decorate        add module name to generated onEnter log statement
  -S PATH, --init-session=PATH
                        path to JavaScript file used to initialize the session
  -P PARAMETERS_JSON, --parameters=PARAMETERS_JSON
                        parameters as JSON, exposed as a global named
                        'parameters'
  -o OUTPUT, --output=OUTPUT
                        dump messages to file
```

Cellebrite

frida-trace

Main tool to create hooks

-U USB

-i Function(s) to trace

target Process to trace

```
C:\Users\orb
λ frida-trace -U Instagram -i "*MD5*"
Instrumenting functions...
CC_MD5_Update: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5_Update.js"
CC_MD5_Final: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5_Final.js"
CC_MD5_Init: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5_Init.js"
CC_MD5: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5.js"
MD5Final: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\MD5Final.js"
_ZN3WTF3MD58checksumERNSt3__15arrayIhLm16EEE: Loaded handler at "C:\\Users\\orb\\__handlers__\\JavaScriptCore\\_ZN3WTF3MD58checksumERNSt3__15ar_d02d754c.js"
_ZN3WTF3MD58addBytesEPKhm: Loaded handler at "C:\\Users\\orb\\__handlers__\\JavaS

...

EncryptedMD5CheckProgressInit: Loaded handler at "C:\\Users\\orb\\__handlers__\\MediaToolbox\\EncryptedMD5CheckProgressInit.js"
EncryptedMD5CheckProgressDestroy: Loaded handler at "C:\\Users\\orb\\__handlers__\\MediaToolbox\\EncryptedMD5CheckProgressDestroy.js"
GEOMD5StringForURLString: Loaded handler at "C:\\Users\\orb\\__handlers__\\GeoServices\\GEOMD5StringForURLString.js"
MD5DigestWithStartBytes: Loaded handler at "C:\\Users\\orb\\__handlers__\\ImageCapture\\MD5DigestWithStartBytes.js"
ML3AudioUtilitiesMD5AudioPacketData: Loaded handler at "C:\\Users\\orb\\__handlers__\\MusicLibrary\\ML3AudioUtilitiesMD5AudioPacketData.js"
ISMD5StringForBytes: Loaded handler at "C:\\Users\\orb\\__handlers__\\iTunesStore\\ISMD5StringForBytes.js"
ColorSyncCodeFragmentGetMD5: Loaded handler at "C:\\Users\\orb\\__handlers__\\ColorSync\\ColorSyncCodeFragmentGetMD5.js"
ColorSyncProfileGetMD5: Loaded handler at "C:\\Users\\orb\\__handlers__\\ColorSync\\ColorSyncProfileGetMD5.js"
Started tracing 43 functions. Press Ctrl+C to stop.
```

Cellebrite

frida-trace

Main tool to create hooks

-U USB

-i Function(s) to trace

target Process to trace

```
C:\Users\orb
λ frida-trace -U Instagram -i "*MD5*"
Instrumenting functions...
CC_MD5_Update: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5_Update.js"
CC_MD5_Final: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5_Final.js"
CC_MD5_Init: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5_Init.js"
CC_MD5: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5.js"
MD5Final: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\MD5Final.js"
_ZN3WTF3MD58checksumERNSt3__15arrayIhLm16EEE: Loaded handler at "C:\\Users\\orb\\__handlers__\\JavaScriptCore\\_ZN3WTF3MD58checksumERNSt3__15ar_d02d754c.js"
_ZN3WTF3MD58addBytesEPKhm: Loaded handler at "C:\\Users\\orb\\__handlers__\\JavaS

...

EncryptedMD5CheckProgressInit: Loaded handler at "C:\\Users\\orb\\__handlers__\\MediaToolbox\\EncryptedMD5CheckProgressInit.js"
EncryptedMD5CheckProgressDestroy: Loaded handler at "C:\\Users\\orb\\__handlers__\\MediaToolbox\\EncryptedMD5CheckProgressDestroy.js"
GEOMD5StringForURLString: Loaded handler at "C:\\Users\\orb\\__handlers__\\GeoServices\\GEOMD5StringForURLString.js"
MD5DigestWithStartBytes: Loaded handler at "C:\\Users\\orb\\__handlers__\\ImageCapture\\MD5DigestWithStartBytes.js"
ML3AudioUtilitiesMD5AudioPacketData: Loaded handler at "C:\\Users\\orb\\__handlers__\\MusicLibrary\\ML3AudioUtilitiesMD5AudioPacketData.js"
ISMD5StringForBytes: Loaded handler at "C:\\Users\\orb\\__handlers__\\iTunesStore\\ISMD5StringForBytes.js"
ColorSyncCodeFragmentGetMD5: Loaded handler at "C:\\Users\\orb\\__handlers__\\ColorSync\\ColorSyncCodeFragmentGetMD5.js"
ColorSyncProfileGetMD5: Loaded handler at "C:\\Users\\orb\\__handlers__\\ColorSync\\ColorSyncProfileGetMD5.js"
Started tracing 43 functions. Press Ctrl+C to stop.
```

Cellebrite

frida-trace

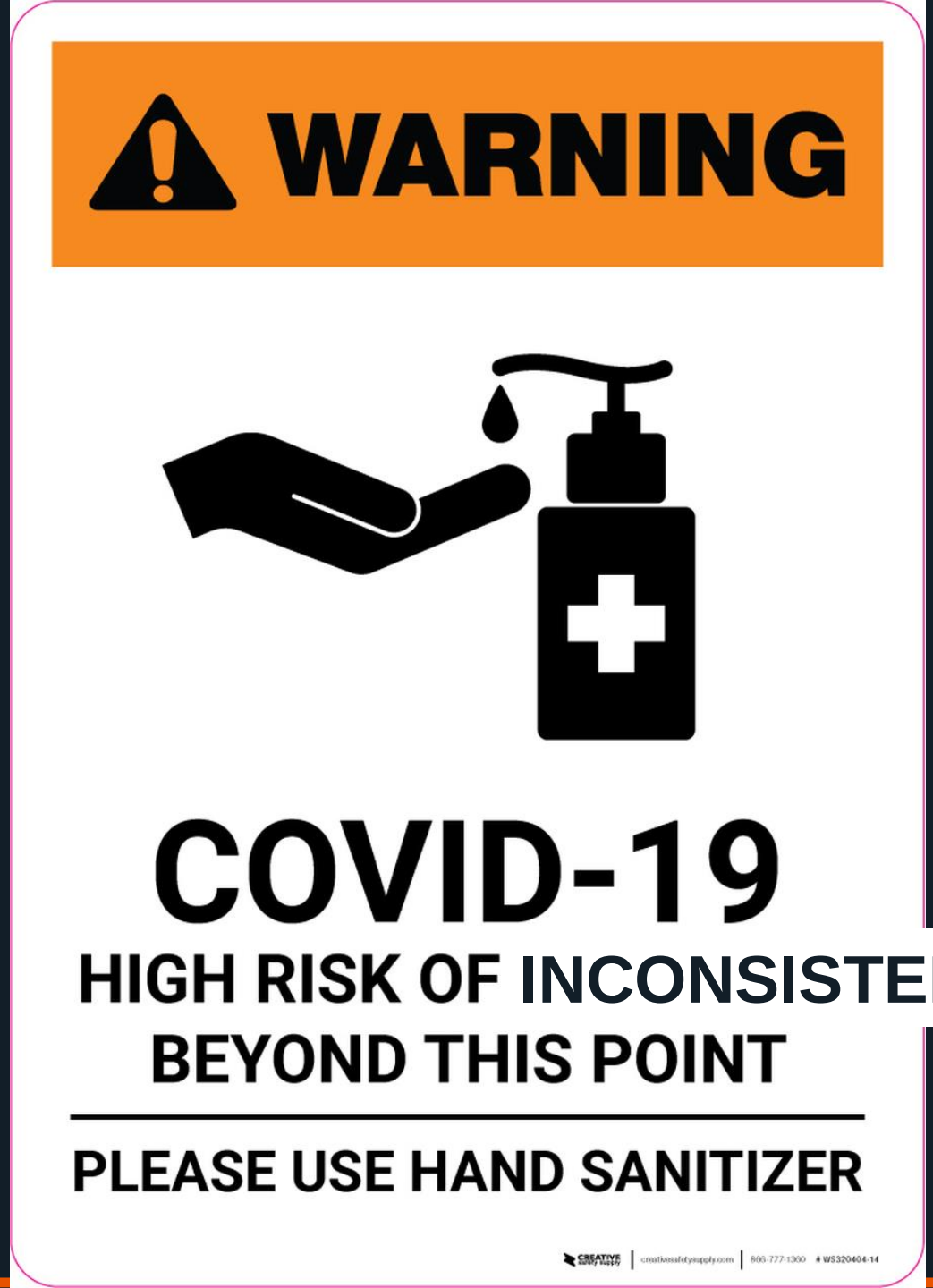
Main tool to create hooks

-U USB

-i Function(s) to trace

target Process to trace

```
C:\Users\orb
λ frida-trace -U Instagram -i "*MD5*"
Instrumenting functions...
CC_MD5_Update: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5_Update.js"
CC_MD5_Final: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5_Final.js"
CC_MD5_Init: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5_Init.js"
CC_MD5: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\CC_MD5.js"
MD5Final: Loaded handler at "C:\\Users\\orb\\__handlers__\\libcommonCrypto.dylib\\MD5Final.js"
_ZN3WTF3MD58checksumERNSt3__15arrayIhLm16EEE: Loaded handler at "C:\\Users\\orb\\__handlers__\\JavaScriptCore\\_ZN3WTF3MD58checksumERNSt3__15ar_d02d754c.js"
_ZN3WTF3MD58addBytesEPKhm: Loaded handler at "C:\\Users\\orb\\__handlers__\\JavaS

...

EncryptedMD5CheckProgressInit: Loaded handler at "C:\\Users\\orb\\__handlers__\\MediaToolbox\\EncryptedMD5CheckProgressInit.js"
EncryptedMD5CheckProgressDestroy: Loaded handler at "C:\\Users\\orb\\__handlers__\\MediaToolbox\\EncryptedMD5CheckProgressDestroy.js"
GEOMD5StringForURLString: Loaded handler at "C:\\Users\\orb\\__handlers__\\GeoServices\\GEOMD5StringForURLString.js"
MD5DigestWithStartBytes: Loaded handler at "C:\\Users\\orb\\__handlers__\\ImageCapture\\MD5DigestWithStartBytes.js"
ML3AudioUtilitiesMD5AudioPacketData: Loaded handler at "C:\\Users\\orb\\__handlers__\\MusicLibrary\\ML3AudioUtilitiesMD5AudioPacketData.js"
ISMD5StringForBytes: Loaded handler at "C:\\Users\\orb\\__handlers__\\iTunesStore\\ISMD5StringForBytes.js"
ColorSyncCodeFragmentGetMD5: Loaded handler at "C:\\Users\\orb\\__handlers__\\ColorSync\\ColorSyncCodeFragmentGetMD5.js"
ColorSyncProfileGetMD5: Loaded handler at "C:\\Users\\orb\\__handlers__\\ColorSync\\ColorSyncProfileGetMD5.js"
Started tracing 43 functions. Press Ctrl+C to stop.
```


WARNING
COVID-19
HIGH RISK OF INCONSISTENCY
BEYOND THIS POINT
PLEASE USE HAND SANITIZER

Cellebrite

frida-trace

Tracing CC_MD5

```
C:\Users\orb
λ frida-trace -U Instagram -i CC_MD5
Instrumenting functions...
CC_MD5: Loaded handler at "C:\\Orb\\Progs\\cmder\\__handlers__\\Pegasus\\CC_MD5.js"
Started tracing 1 function. Press Ctrl+C to stop.
```

Cellebrite

frida-trace

Tracing CC_MD5

Playing with the device

```
C:\Users\orb
λ frida-trace -U Instagram -i CC_MD5
Instrumenting functions...
CC_MD5: Loaded handler at "C:\\Orb\\Progs\\cmder\\__handlers__\\Pegasus\\CC_MD5.js"
Started tracing 1 function. Press Ctrl+C to stop.
           /* TID 0x111cf */
  3483 ms  CC_MD5()
           /* TID 0xce6b */
  3582 ms  CC_MD5()
  3730 ms  CC_MD5()
  3779 ms  CC_MD5()
           /* TID 0xcfaf */
  4200 ms  CC_MD5()
  4248 ms  CC_MD5()
           /* TID 0x56a7 */
  4448 ms  CC_MD5()
           /* TID 0x111cf */
  4495 ms  CC_MD5()
           /* TID 0x56a7 */
  4883 ms  CC_MD5()
           /* TID 0x624f */
  4922 ms  CC_MD5()
  5734 ms  CC_MD5()
  5779 ms  CC_MD5()
           /* TID 0xce6b */
  6901 ms  CC_MD5()
           /* TID 0xcfaf */
  6952 ms  CC_MD5()
  7194 ms  CC_MD5()
           /* TID 0x624f */
  7243 ms  CC_MD5()
           /* TID 0x56a7 */
  7650 ms  CC_MD5()
  7704 ms  CC_MD5()
           /* TID 0x111cf */
  7978 ms  CC_MD5()
  8036 ms  CC_MD5()
```

Cellebrite

Actual calls to CC_MD5 with timestamps

frida-trace

Tracing CC_MD5 while playing with the device

```
C:\Users\orb
λ frida-trace -U Instagram -i CC_MD5
Instrumenting functions...
CC_MD5: Loaded handler at "C:\\Orb\\Progs\\cmder\\__handlers__\\Pegasus\\CC_MD5.js"
Started tracing 1 function. Press Ctrl+C to stop.
           /* TID 0x111cf */
  3483 ms  CC_MD5()
           /* TID 0xce6b */
  3582 ms  CC_MD5()
  3730 ms  CC_MD5()
  3779 ms  CC_MD5()
           /* TID 0xcfaf */
  4200 ms  CC_MD5()
  4248 ms  CC_MD5()
           /* TID 0x56a7 */
  4448 ms  CC_MD5()
           /* TID 0x111cf */
  4495 ms  CC_MD5()
           /* TID 0x56a7 */
  4883 ms  CC_MD5()
           /* TID 0x624f */
  4922 ms  CC_MD5()
  5734 ms  CC_MD5()
  5779 ms  CC_MD5()
           /* TID 0xce6b */
  6901 ms  CC_MD5()
           /* TID 0xcfaf */
  6952 ms  CC_MD5()
  7194 ms  CC_MD5()
           /* TID 0x624f */
  7243 ms  CC_MD5()
           /* TID 0x56a7 */
  7650 ms  CC_MD5()
  7704 ms  CC_MD5()
           /* TID 0x111cf */
  7978 ms  CC_MD5()
  8036 ms  CC_MD5()
```

Cellebrite

frida-trace handlers

JavaScript templates for implementing hooks

```
/*
 * Auto-generated by Frida. Please modify to match the signature of CC_MD5.
 * This stub is currently auto-generated from manpages when available.
 *
 * For full API reference, see: http://www.frida.re/docs/javascript-api/
 */

{
  /**
   * Called synchronously when about to call CC_MD5.
   *
   * @this {object} - Object allowing you to store state for use in onLeave.
   * @param {function} log - Call this function with a string to be presented to the user.
   * @param {array} args - Function arguments represented as an array of NativePointer objects.
   * For example use args[0].readUtf8String() if the first argument is a pointer to a C string encoded as UTF-8.
   * It is also possible to modify arguments by assigning a NativePointer object to an element of this array.
   * @param {object} state - Object allowing you to keep state across function calls.
   * Only one JavaScript function will execute at a time, so do not worry about race-conditions.
   * However, do not use this to store function arguments across onEnter/onLeave, but instead
   * use "this" which is an object for keeping state local to an invocation.
   */
  onEnter: function (log, args, state) {
    log('CC_MD5()');
  },

  /**
   * Called synchronously when about to return from CC_MD5.
   *
   * See onEnter for details.
   *
   * @this {object} - Object allowing you to access state stored in onEnter.
   * @param {function} log - Call this function with a string to be presented to the user.
   * @param {NativePointer} retval - Return value represented as a NativePointer object.
   * @param {object} state - Object allowing you to keep state across function calls.
   */
  onLeave: function (log, retval, state) {
  }
}
```

Cellebrite

frida-trace handler

JavaScript templates for implementing hooks

```
onEnter: function (log, args, state) {
    log('CC_MD5()');



}
```

```
onLeave: function (log, retval, state) {



}
```

Cellebrite

frida-trace handler

JavaScript templates for implementing hooks

implementing `onEnter`

```
onEnter: function (log, args, state) {
    log('CC_MD5()');
    log("Hashing data: ");


}
```

```
onLeave: function (log, retval, state) {


}
```

Cellebrite

frida-trace handler

JavaScript templates for implementing hooks

implementing `onEnter`

```
onEnter: function (log, args, state) {
    log('CC_MD5()');
    log("Hashing data: " + args[0]);


}
```

```
onLeave: function (log, retval, state) {


}
```

Cellebrite

frida-trace handler

JavaScript templates for implementing hooks

implementing `onEnter`

```
onEnter: function (log, args, state) {
    log('CC_MD5()');
    log("Hashing data: " + args[0].readCString());


}
```

```
onLeave: function (log, retval, state) {


}
```

Cellebrite

frida-trace handler

JavaScript templates for implementing hooks

implementing `onEnter`

```
onEnter: function (log, args, state) {
    log('CC_MD5()');
    log("Hashing data: " +
        args[0].readCString(args[1].toInt32()));

}
```

```
onLeave: function (log, retval, state) {


}
```

Cellebrite

frida-trace

Tracing CC_MD5

Playing with the device

Logging input

```
c:\Users\orb
λ frida-trace -U Instagram -i CC_MD5
Instrumenting functions...
CC_MD5: Loaded handler at "c:\\Users\\orb\\__handlers__\\Pegasus\\CC_MD5.js"
Started tracing 1 function. Press Ctrl+C to stop.
           /* TID 0x8213 */
  1086 ms  CC_MD5()
  1086 ms  Hashing data: MGCopyAnswerProductVersion
  1149 ms  CC_MD5()
  1149 ms  Hashing data: YW5vbnltb3VzX3Byb2ZpbGVfcGlj.2
  1176 ms  CC_MD5()
  1176 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/95140556_594026277870211_4156802974091313152_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=eyyPZZqBQv0AX97kpro&oh=680b34ff827d65ae4d8dd818ff991a9d&oe=5EE69064
  1181 ms  CC_MD5()
  1181 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/59381178_2348911458724961_5863612957363011584_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=cZY-YnIHFegAX-35LKM&oh=5e8d141f9edfae0e050dc89827d1f79f&oe=5EE760A3
  1184 ms  CC_MD5()
  1184 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/96141680_258597662175001_3551320530768363520_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=VvJCow_WtnMAX-AU4Sy&oh=cf0e6f633a85590c64430e41ffbebe17&oe=5EE6F838
  1187 ms  CC_MD5()
  1187 ms  Hashing data: YW5vbnltb3VzX3Byb2ZpbGVfcGlj.2
           /* TID 0xa07 */
  4203 ms  CC_MD5()
  4203 ms  Hashing data: MGCopyAnswerHasThinBezel
           /* TID 0x8213 */
  4270 ms  CC_MD5()
  4270 ms  Hashing data: YW5vbnltb3VzX3Byb2ZpbGVfcGlj.2
  5821 ms  CC_MD5()
  5821 ms  Hashing data: MGCopyAnswermultitasking
  5869 ms  CC_MD5()
  5869 ms  Hashing data: MjMwODI3MDQ0NzgxNTc1MDY3Mg==.2-w640h1136
  9712 ms  CC_MD5()
  9712 ms  Hashing data: MjMwODI3MDQ0NzgxNTc1MDY3Mg==.2-w640h1136
           /* TID 0xa07 */
```

Cellebrite

frida-trace

Tracing CC_MD5

Playing with the device

Logging input

```
c:\Users\orb
λ frida-trace -U Instagram -i CC_MD5
Instrumenting functions...
CC_MD5: Loaded handler at "c:\\Users\\orb\\__handlers__\\Pegasus\\CC_MD5.js"
Started tracing 1 function. Press Ctrl+C to stop.
           /* TID 0x8213 */
  1086 ms  CC_MD5()
  1086 ms  Hashing data: MGCopyAnswerProductVersion
  1149 ms  CC_MD5()
  1149 ms  Hashing data: YW5vbnltb3VzX3Byb2ZpbGVfcGlj.2
  1176 ms  CC_MD5()
  1176 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/95140556_5940
2627787870211_4156802974091313152_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=eyyPZZqBQv0AX97kpr
o&oh=680b34ff827d65ae4d8dd818ff991a9d&oe=5EE69064
  1181 ms  CC_MD5()
  1181 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/59381178_2348
911458724961_5863612957363011584_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=cZY-YnIHFegAX-35L
KM&oh=5e8d141f9edfae0e050dc89827d1f79f&oe=5EE760A3
  1184 ms  CC_MD5()
  1184 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/96141680_2585
97662175001_3551320530768363520_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=VvJCow_WtnMAX-AU4S
y&oh=cf0e6f633a85590c64430e41ffbebe17&oe=5EE6F838
  1187 ms  CC_MD5()
  1187 ms  Hashing data: YW5vbnltb3VzX3Byb2ZpbGVfcGlj.2
           /* TID 0xa07 */
  4203 ms  CC_MD5()
  4203 ms  Hashing data: MGCopyAnswerHasThinBezel
           /* TID 0x8213 */
  4270 ms  CC_MD5()
  4270 ms  Hashing data: YW5vbnltb3VzX3Byb2ZpbGVfcGlj.2
  5821 ms  CC_MD5()
  5821 ms  Hashing data: MGCopyAnswermultitasking
  5869 ms  CC_MD5()
  5869 ms  Hashing data: MjMwODI3MDQ0NzgxNTc1MDY3Mg==.2-w640h1136
  9712 ms  CC_MD5()
  9712 ms  Hashing data: MjMwODI3MDQ0NzgxNTc1MDY3Mg==.2-w640h1136
           /* TID 0xa07 */
```

Cellebrite

frida-trace handler

JavaScript templates for implementing hooks

implementing `onEnter`

implementing `onLeave`

```
onEnter: function (log, args, state) {
    log('CC_MD5()');
    log("Hashing data: " +
        args[0].readCString(args[1].toInt32()));

}
```

```
onLeave: function (log, retval, state) {
    log("CC_MD5 returned");

}
```

Cellebrite

frida-trace

Tracing CC_MD5

Playing with the device

Logging input and returns

```
c:\Users\orb
λ frida-trace -U Instagram -i CC_MD5
Instrumenting functions...
CC_MD5: Loaded handler at "c:\\Users\\orb\\__handlers__\\Pegasus\\CC_MD5.js"
Started tracing 1 function. Press Ctrl+C to stop.
           /* TID 0xa027 */
  4663 ms  CC_MD5()
  4663 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/79160523_43273
3534276483_3524882799448293376_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=sZXLiNS0W34AX9-1e8
n&oh=2c2ffdd0021805cc1c15f14717c5be9f&oe=5EE634AC
  4664 ms  CC_MD5 returned
  4767 ms  CC_MD5()
  4767 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/79160523_43273
3534276483_3524882799448293376_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=sZXLiNS0W34AX9-1e8
n&oh=2c2ffdd0021805cc1c15f14717c5be9f&oe=5EE634AC
  4767 ms  CC_MD5 returned
  4776 ms  CC_MD5()
  4776 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/68712457_35987
7354893479_2289928823942152192_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=2K0rQhswLuMAX9hqFI
7&oh=9f83281412008144b26dca71caeaf2a5&oe=5EE721D0
  4776 ms  CC_MD5 returned
           /* TID 0xcd5f */
  4810 ms  CC_MD5()
  4810 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/68712457_35987
7354893479_2289928823942152192_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=2K0rQhswLuMAX9hqFI
7&oh=9f83281412008144b26dca71caeaf2a5&oe=5EE721D0
  4810 ms  CC_MD5 returned
  5813 ms  CC_MD5()
  5813 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/46411694_22765
32852582777_7202222863296757760_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=fWmUX0cLVN0AX8fAZ
8B&oh=aad920fb02c35593845d5add43bedd28&oe=5EE6959B
  5813 ms  CC_MD5 returned
```

Cellebrite

frida-trace handler

JavaScript templates for implementing hooks

implementing `onEnter`

implementing `onLeave`

```
onEnter: function (log, args, state) {
    log('CC_MD5()');
    log("Hashing data: " +
        args[0].readCString(args[1].toInt32()));

}
```

```
onLeave: function (log, retval, state) {
    log("CC_MD5 returned: ");
    log(hexdump(retval, {length: 16}));
}
```

Cellebrite

frida-trace

Tracing CC_MD5

Playing with the device

Logging input and return values

```
c:\Users\orb
λ frida-trace -U Instagram -i CC_MD5
Instrumenting functions...
CC_MD5: Loaded handler at "c:\\Users\\orb\\__handlers__\\Pegasus\\CC_MD5.js"
Started tracing 1 function. Press Ctrl+C to stop.
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16ef76a18  c4 b7 3b 7b 23 b0 77 e4 a5 47 cb 46 f2 56 ab d2  ..;{#.w..G.F.V..
           /* TID 0xf3bf */
  6894 ms  CC_MD5()
  6894 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/79825544_470124460599150_1224692418167701504_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=1gKx794ef6gAX8_ZgtS&oh=ffa766accf4f28311a7570664f26ab05&oe=5EE7F5EC
  6894 ms  CC_MD5 returned
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16ef76a88  c4 b7 3b 7b 23 b0 77 e4 a5 47 cb 46 f2 56 ab d2  ..;{#.w..G.F.V..
  6968 ms  CC_MD5()
  6968 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/79825544_470124460599150_1224692418167701504_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=1gKx794ef6gAX8_ZgtS&oh=ffa766accf4f28311a7570664f26ab05&oe=5EE7F5EC
  6968 ms  CC_MD5 returned
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16eeeaa18  99 90 b7 85 08 57 d1 cd 3a f0 dd e6 f5 af 77 91  .....W..:.....w.
           /* TID 0x10c3b */
  7936 ms  CC_MD5()
  7936 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/67310557_649773548849427_4130659181743046656_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=rauUAF2krrIAX_7olUv&oh=10b7a81fd8b1508f323e056d6421deba&oe=5EE8A0D3
  7937 ms  CC_MD5 returned
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16eeeaa88  99 90 b7 85 08 57 d1 cd 3a f0 dd e6 f5 af 77 91  .....W..:.....w.
  8001 ms  CC_MD5()
  8001 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/67310557_649773548849427_4130659181743046656_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=rauUAF2krrIAX_7olUv&oh=10b7a81fd8b1508f323e056d6421deba&oe=5EE8A0D3
  8001 ms  CC_MD5 returned
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16eeeaa18  07 b1 f3 21 b5 00 60 df 76 32 b4 a6 b2 a7 38 55  ...!..`.v2....8U
```

Cellebrite

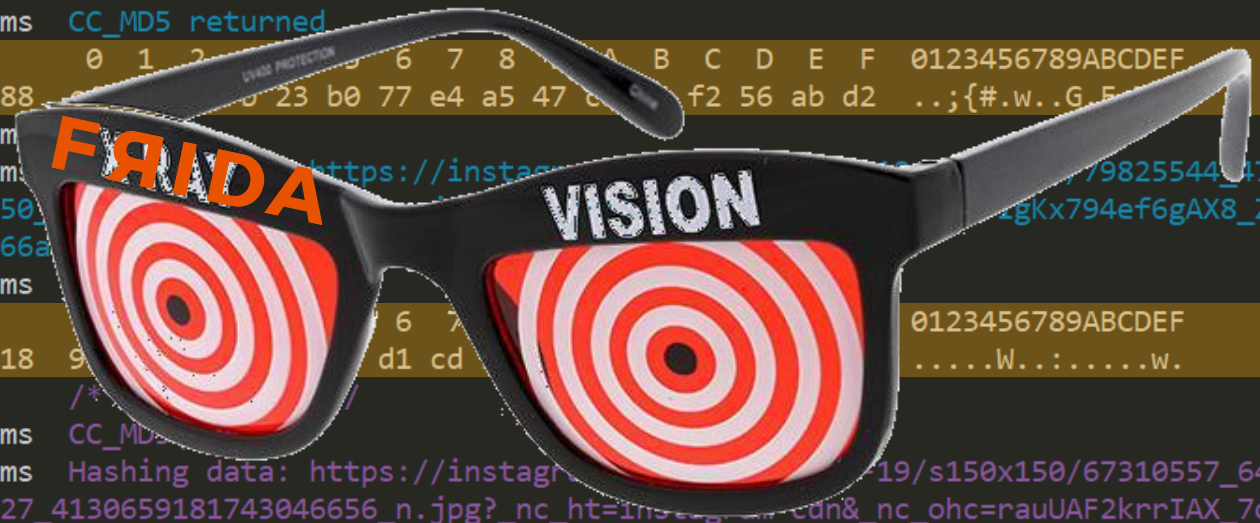
frida-trace

Tracing CC_MD5

Playing with the device

Logging input and return values

```
c:\Users\orb
λ frida-trace -U Instagram -i CC_MD5
Instrumenting functions...
CC_MD5: Loaded handler at "c:\\Users\\orb\\__handlers__\\Pegasus\\CC_MD5.js"
Started tracing 1 function. Press Ctrl+C to stop.
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16ef76a18  c4 b7 3b 7b 23 b0 77 e4 a5 47 cb 46 f2 56 ab d2  ..;{#.w..G.F.V..
           /* TID 0xf3bf */
  6894 ms  CC_MD5()
  6894 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/79825544_470124
460599150_1224692418167701504_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=1gKx794ef6gAX8_ZgtS&
oh=ffa766accf4f28311a7570664f26ab05&oe=5EE7F5EC
  6894 ms  CC_MD5 returned
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16ef76a88  c4 b7 3b 7b 23 b0 77 e4 a5 47 cb 46 f2 56 ab d2  ..;{#.w..G.F.V..
  6968 ms
  6968 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/79825544_470124
460599150_1224692418167701504_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=1gKx794ef6gAX8_ZgtS&
oh=ffa766a
  6968 ms
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16eeeaa18  99 90 b7 85 08 57 d1 cd 3a f0 dd e6 f5 af 77 91  .....W..:.....w.
           /*
  7936 ms  CC_MD5()
  7936 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/67310557_649773
548849427_4130659181743046656_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=rauUAF2krrIAX_7olUv&
oh=10b7a81fd8b1508f323e056d6421deba&oe=5EE8A0D3
  7937 ms  CC_MD5 returned
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16eeeaa88  99 90 b7 85 08 57 d1 cd 3a f0 dd e6 f5 af 77 91  .....W..:.....w.
  8001 ms  CC_MD5()
  8001 ms  Hashing data: https://instagram-cdn/v/t51.2885-19/s150x150/67310557_649773
548849427_4130659181743046656_n.jpg?_nc_ht=instagram-cdn&_nc_ohc=rauUAF2krrIAX_7olUv&
oh=10b7a81fd8b1508f323e056d6421deba&oe=5EE8A0D3
  8001 ms  CC_MD5 returned
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16eeeaa18  07 b1 f3 21 b5 00 60 df 76 32 b4 a6 b2 a7 38 55  ...!..`.v2....8U
```



Recap

What we have so far

Message record

Image metadata, URLs

Image file

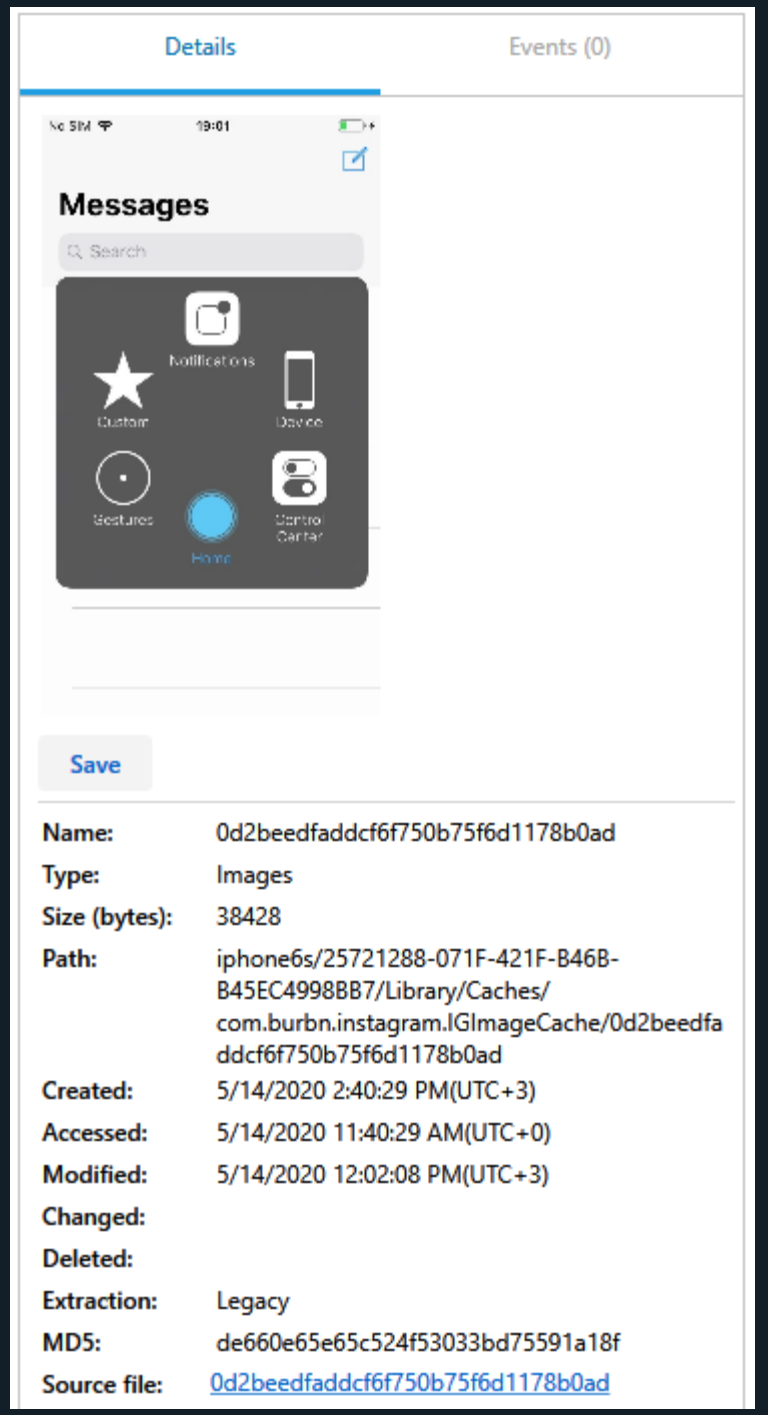
Hashed filename

Traced MD5 function

Input string -> output filename

Image file

0D2BEEDFADDCF6F750B75F6D1178B0AD



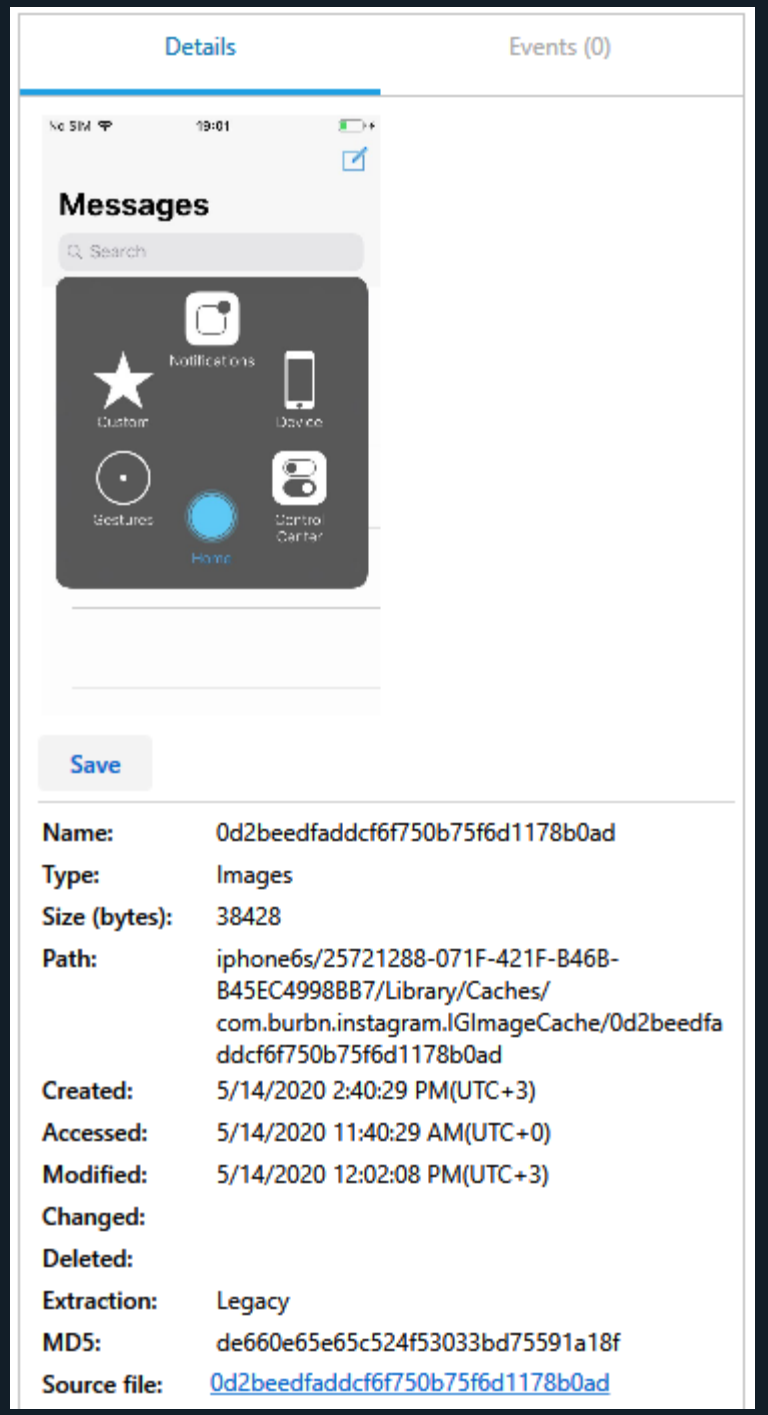
Cellebrite

Image file

0D2BEEDFADDCF6F750B75F6D1178B0AD

frida-trace output

```
CC_MD5()
Hashing data:
MjMwODI3MDQ0NzgxNTc1MDY3Mg==.2-w640h1136

CC_MD5 returned
 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0d 2b ee df ad dc f6 f7 50 b7 5f 6d 11 78 b0 ad
```



Cellebrite

Message record

```
IGImageURL {
    NS.relative =
        "https://instagram.ftlv2-1.fna.fbcdn.net/v/t51.2885-
        15/e35/96829986_140831820890044_3094139032359062170_
        n.jpg?_nc_ht=instagram.ftlv2-
        1.fna.fbcdn.net&_nc_cat=104&_nc_ohc=WO1CpDRZQVkAX93F
        M0n&oh=9b9ec5bf640f98a389fc267470cf756c&oe=5EE56316&
        ig_cache_key=MjMwODI3MDQ0NzgxNTc1MDY3Mg==.2",
    width = 640,
    height = 1136
}
```

Message record

MD5 input:
MjMwODI3MDQ0NzgxNTc1MDY3Mg==.2-w640h1136

```
IGImageURL {
    NS.relative =
        "https://instagram.ftlv2-1.fna.fbcdn.net/v/t51.2885-
        15/e35/96829986_140831820890044_3094139032359062170_
        n.jpg?_nc_ht=instagram.ftlv2-
        1.fna.fbcdn.net&_nc_cat=104&_nc_ohc=WO1CpDRZQVkAX93F
        M0n&oh=9b9ec5bf640f98a389fc267470cf756c&oe=5EE56316&
        ig_cache_key=MjMwODI3MDQ0NzgxNTc1MDY3Mg==.2",
    width = 640,
    height = 1136
}
```

Message record

MD5 input:
MjMwODI3MDQ0NzgxNTc1MDY3Mg==.2-w640h1136

```
IGImageURL {
    NS.relative =
        "https://instagram.ftlv2-1.fna.fbcdn.net/v/t51.2885-
        15/e35/96829986_140831820890044_3094139032359062170_
        n.jpg?_nc_ht=instagram.ftlv2-
        1.fna.fbcdn.net&_nc_cat=104&_nc_ohc=WO1CpDRZQVkAX93F
        M0n&oh=9b9ec5bf640f98a389fc267470cf756c&oe=5EE56316&
        ig_cache_key=MjMwODI3MDQ0NzgxNTc1MDY3Mg==.2",
    width = 640,
    height = 1136
}
```

Message record

MD5 input:
MjMwODI3MDQ0NzgxNTc1MDY3Mg==.2-w640h1136

```
IGImageURL {
    NS.relati
        "https                                    et/v/t51.2885-
        15/e35                                    9032359062170_
        n.jpg?
        1.fna.                                    1CpDRZQVkAX93F
        M0n&oh                                    c&oe=5EE56316&
        ig_cac                                    g==.2",
    width = 6
    height = 1136
}
```

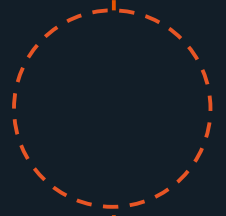
Agenda

Overview of mobile app decoding

And some technical challenges in it

Introduction to dynamic instrumentation

And how it can solve those challenges



Hands-on work with FRIDA

Solving more hard problems

Cellebrite

Decrypting Wickr

Hands-on exercise

Cellebrite

Wickr.Me

Multi-platform secure messaging application

Cellebrite

Wickr Security Features

- Military grade end-to-end encryption — Wickr runs the only triple encrypted zero trust SaaS platform on the planet
- User-defined burn-on-read settings so you can decide what is done with your communications once they are sent
- Wickr Open Access ‘Smart VPN’ provides a secure connection even in less-than-secure environments
- Screenshot detection available
- Data shredder and key verification ensure you know who you are talking to and who has your files. Control your Data!

https://wickr.com/why-wickr/

Cellebrite

Does Wickr protect my data from forensic tools?

Another feature of forensic tools is to recover and report on any application data found on a device. In this regard, these tools don't necessarily do anything special beyond reading and displaying data in such a way that can be presented in a court of law, so Wickr apps don't do anything to thwart them specifically. Rather, we rely on strong application security controls like encrypted app storage and authentication to protect against the generic threat of an attacker accessing your device and viewing your data - with or without a forensic tool.

What you should always remember is that with your password, or with access to your app which you've conveniently configured for autologin, forensic tools are just as capable as hackers or anyone else of accessing your app data. So, keep your password strong and use things like autologin wisely.

https://support.wickr.com/hc/en-us/articles/360025596733

Plan

Install Wickr.Me

And generate some data

Do some forensics

Find where Wickr stores data

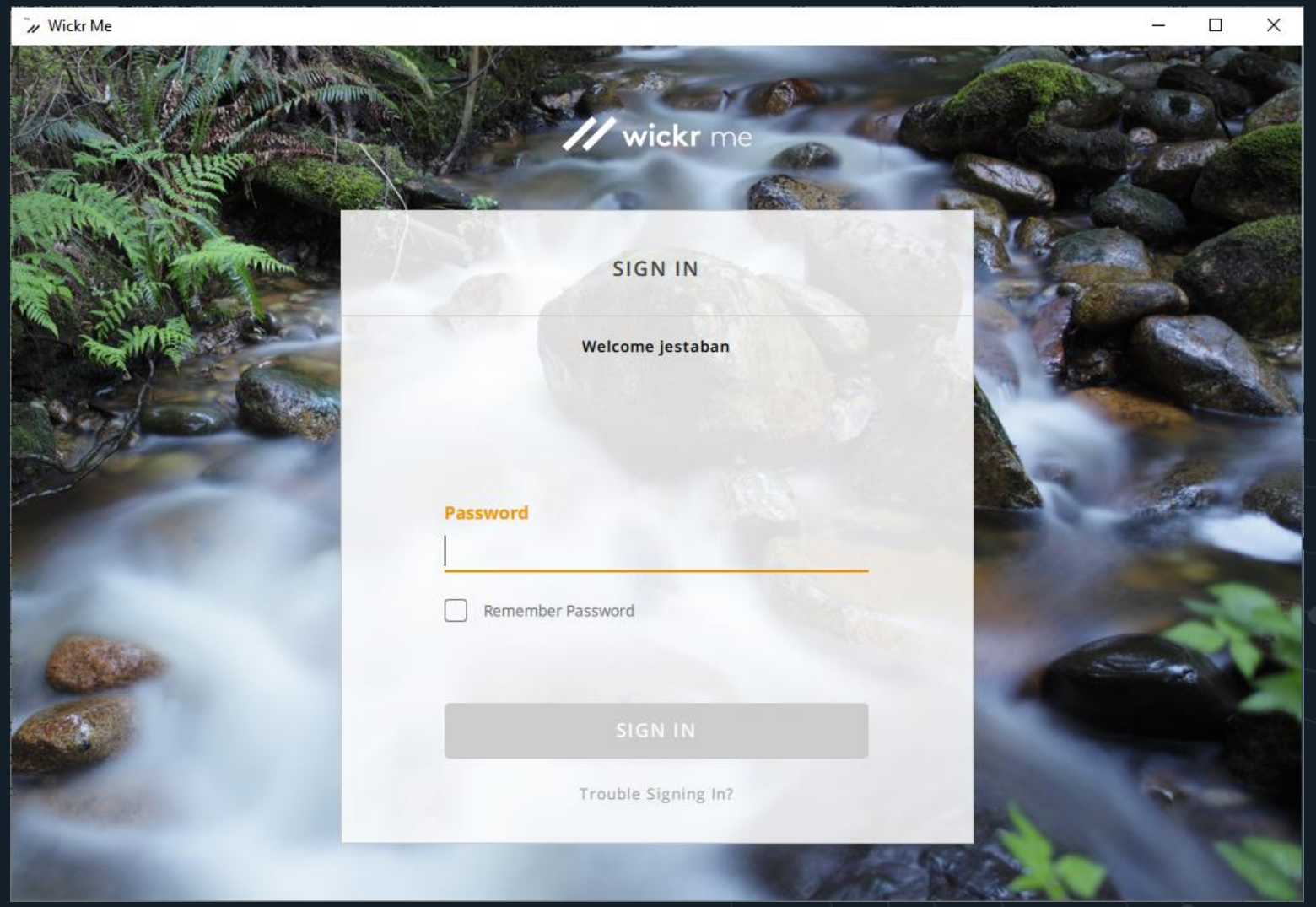
Do some tracing with FRIDA

And find the key to the encrypted database

Cellebrite

Log-in screen

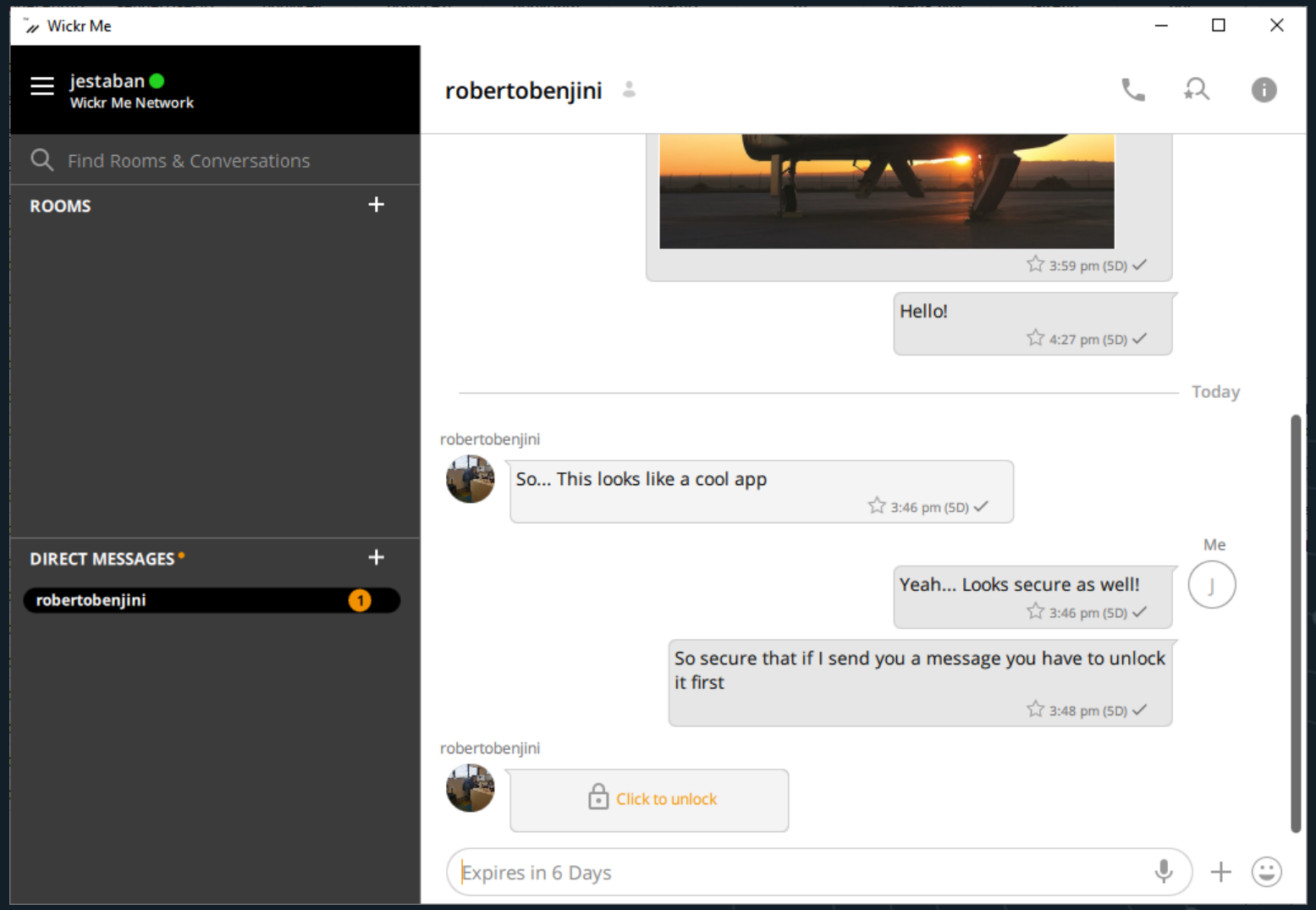
Password is required to use the application



Cellebrite

Chats

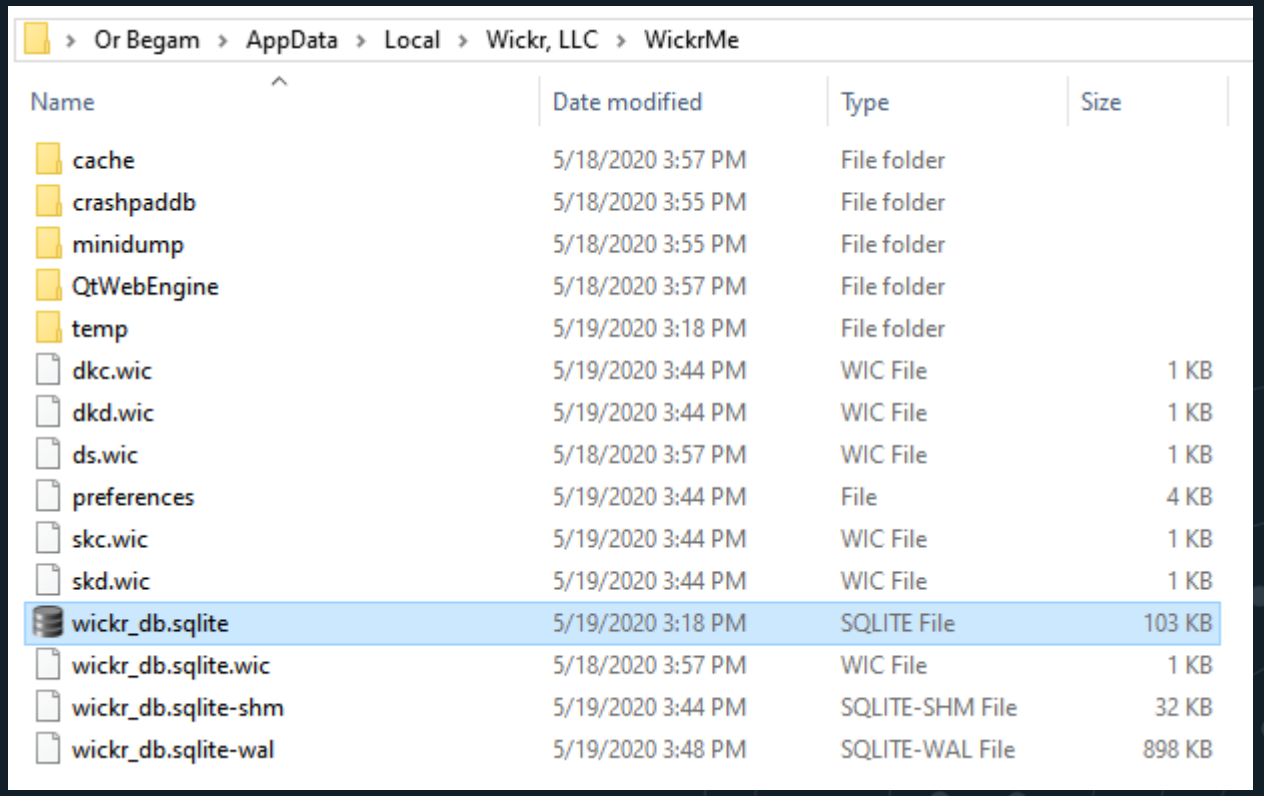
Message arrive “locked” and need user action to “unlock”



Cellebrite

Data storage

wickr_db.sqlite found in AppData folder

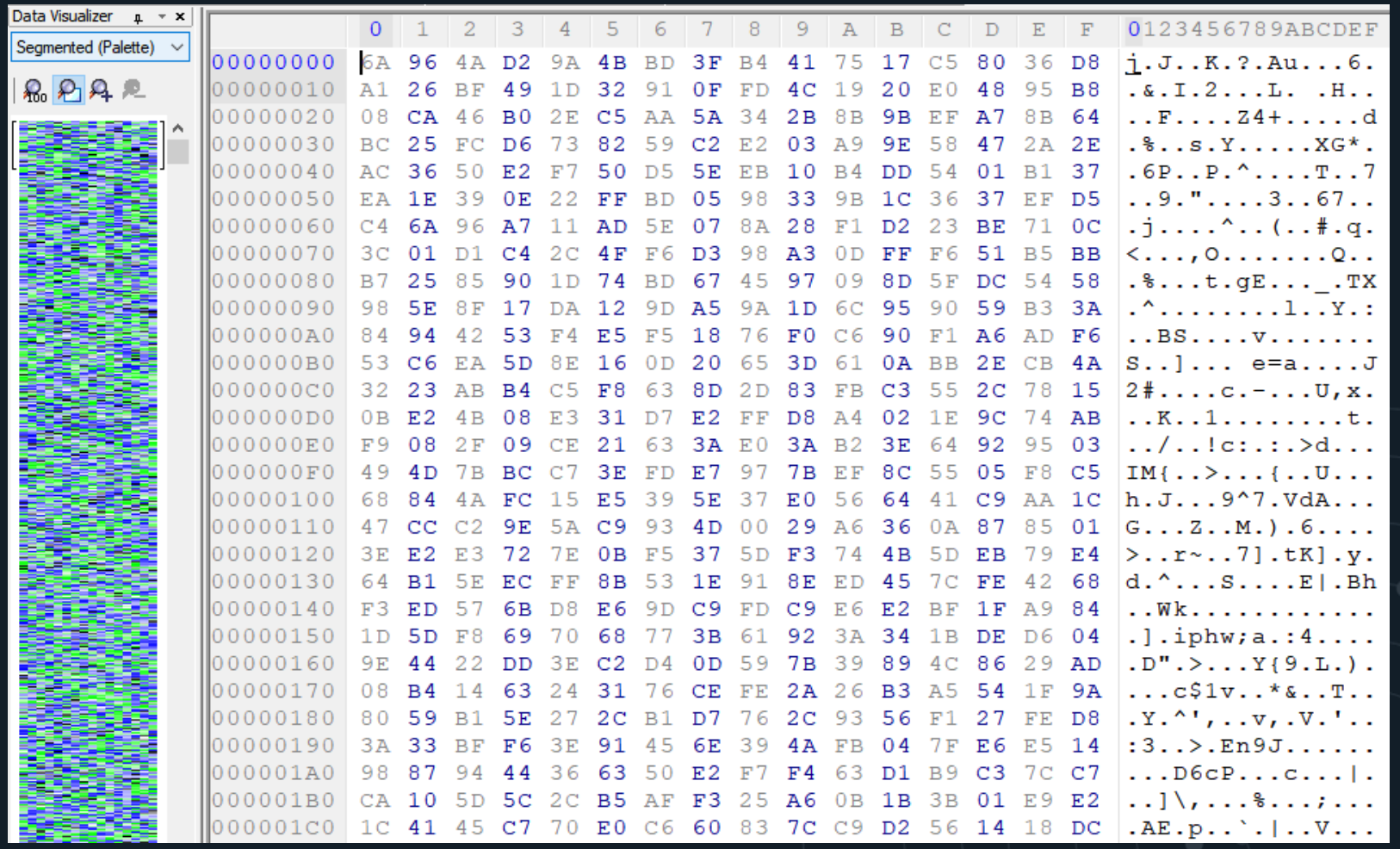


Cellebrite

Data storage

wickr_db.sqlite found in AppData folder

Looks encrypted :/



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00000000	6A	96	4A	D2	9A	4B	BD	3F	B4	41	75	17	C5	80	36	D8	j..J..K.?.Au...6.
00000010	A1	26	BF	49	1D	32	91	0F	FD	4C	19	20	E0	48	95	B8	.&.I.2...L. .H..
00000020	08	CA	46	B0	2E	C5	AA	5A	34	2B	8B	9B	EF	A7	8B	64	..F....Z4+.....d
00000030	BC	25	FC	D6	73	82	59	C2	E2	03	A9	9E	58	47	2A	2E	.%..s.Y.....XG*.
00000040	AC	36	50	E2	F7	50	D5	5E	EB	10	B4	DD	54	01	B1	37	.6P..P.^....T..7
00000050	EA	1E	39	0E	22	FF	BD	05	98	33	9B	1C	36	37	EF	D5	..9."....3..67..
00000060	C4	6A	96	A7	11	AD	5E	07	8A	28	F1	D2	23	BE	71	0C	.j....^..(..#.q.
00000070	3C	01	D1	C4	2C	4F	F6	D3	98	A3	0D	FF	F6	51	B5	BB	<...,O.......Q..
00000080	B7	25	85	90	1D	74	BD	67	45	97	09	8D	5F	DC	54	58	.%...t.gE..._.TX
00000090	98	5E	8F	17	DA	12	9D	A5	9A	1D	6C	95	90	59	B3	3A	.^.......l..Y.:
000000A0	84	94	42	53	F4	E5	F5	18	76	F0	C6	90	F1	A6	AD	F6	..BS....v.......
000000B0	53	C6	EA	5D	8E	16	0D	20	65	3D	61	0A	BB	2E	CB	4A	S..]... e=a....J
000000C0	32	23	AB	B4	C5	F8	63	8D	2D	83	FB	C3	55	2C	78	15	2#....c.-...U,x.
000000D0	0B	E2	4B	08	E3	31	D7	E2	FF	D8	A4	02	1E	9C	74	AB	..K..1........t.
000000E0	F9	08	2F	09	CE	21	63	3A	E0	3A	B2	3E	64	92	95	03	../..!c:.:.>d...
000000F0	49	4D	7B	BC	C7	3E	FD	E7	97	7B	EF	8C	55	05	F8	C5	IM{..>...{..U...
00000100	68	84	4A	FC	15	E5	39	5E	37	E0	56	64	41	C9	AA	1C	h.J...9^7.VdA...
00000110	47	CC	C2	9E	5A	C9	93	4D	00	29	A6	36	0A	87	85	01	G...Z..M.).6....
00000120	3E	E2	E3	72	7E	0B	F5	37	5D	F3	74	4B	5D	EB	79	E4	>..r~..7].tK].y.
00000130	64	B1	5E	EC	FF	8B	53	1E	91	8E	ED	45	7C	FE	42	68	d.^...S....E|.Bh
00000140	F3	ED	57	6B	D8	E6	9D	C9	FD	C9	E6	E2	BF	1F	A9	84	..Wk............
00000150	1D	5D	F8	69	70	68	77	3B	61	92	3A	34	1B	DE	D6	04	.].iphw;a.:4....
00000160	9E	44	22	DD	3E	C2	D4	0D	59	7B	39	89	4C	86	29	AD	.D".>...Y{9.L.).
00000170	08	B4	14	63	24	31	76	CE	FE	2A	26	B3	A5	54	1F	9A	...c$1v..*&..T..
00000180	80	59	B1	5E	27	2C	B1	D7	76	2C	93	56	F1	27	FE	D8	.Y.^',..v,.V.'..
00000190	3A	33	BF	F6	3E	91	45	6E	39	4A	FB	04	7F	E6	E5	14	:3..>.En9J......
000001A0	98	87	94	44	36	63	50	E2	F7	F4	63	D1	B9	C3	7C	C7	...D6cP...c...|.
000001B0	CA	10	5D	5C	2C	B5	AF	F3	25	A6	0B	1B	3B	01	E9	E2	..]\,...%...;...
000001C0	1C	41	45	C7	70	E0	C6	60	83	7C	C9	D2	56	14	18	DC	.AE.p..`.|..V...

Cellebrite

SQLite Encryption

SQLCipher is the de-facto standard

Open-source:

https://github.com/sqlcipher/sqlcipher

```
% hexdump -C unencrypted-sqlite.db
00000000  53 51 4c 69 74 65 20 66  6f 72 6d 61 74 20 33 00  |SQLite format 3.|
00000010  04 00 01 01 00 40 20 20  00 00 00 02 00 00 00 03  |.....@  ........|
000003b0  00 00 00 00 00 00 00 00  00 00 00 00 00 41 01 06  |.............A..|
000003c0  17 1b 1b 01 5b 74 61 62  6c 65 73 65 63 72 65 74  |....[tablesecret|
000003d0  73 73 65 63 72 65 74 73  03 43 52 45 41 54 45 20  |ssecrets.CREATE |
000003e0  54 41 42 4c 45 20 73 65  63 72 65 74 73 28 69 64  |TABLE secrets(id|
000003f0  2c 20 70 61 73 73 77 6f  72 64 2c 20 6b 65 79 29  |, password, key)|
00000bd0  00 00 00 00 00 00 00 00  00 00 00 00 00 21 01 04  |.............!..|
00000be0  25 1d 1f 4c 61 75 6e 63  68 20 43 6f 64 65 73 70  |%..Launch Codesp|
00000bf0  61 24 24 77 6f 72 64 70  72 6f 6a 65 74 69 6c 65  |a$$wordprojetile|
```

```
% hexdump -C encrypted-sqlcipher.db
00000000  de ab bc 3a 40 2b 5d 00  b0 d2 9e 3b 75 91 76 73  |...:@+]....;u.vs|
00000010  bc 41 70 0c 8c ab a0 7a  37 eb a2 a8 a9 27 a5 0a  |.Ap....z7....'..|
00000020  38 c9 0b 9c 06 57 78 96  67 a2 e5 78 f8 8c 58 f3  |8....Wx.g..x..X.|
00000030  ea 7c c6 23 14 8a 75 33  d0 a5 2c 30 2e e1 a4 96  |.|.#..u3..,0....|
00000040  b1 c6 5a 21 67 0a 31 bb  3b de a2 d4 80 b4 60 e3  |..Z!g.1.;.....`.|
00000050  05 b0 75 04 f2 26 66 ed  c7 4e 7e 9c ac 2e ec 1d  |..u..&f..N~.....|
00000060  2d fc 31 b4 32 ce 24 0a  d0 23 71 b0 1f 21 12 2c  |-.1.2.$..#q..!.,|
00000070  92 af 8e d9 de ac 76 e6  20 62 56 c6 f5 05 f5 b3  |......v. bV.....|
00000080  53 d0 5f 4c 5e ec 5b 8a  be e7 d1 46 f0 d9 dc b9  |S._L^.[....F....|
00000090  a3 59 d6 63 a4 ae cf d8  e4 82 29 83 dd c7 86 13  |.Y.c......).....|
```

https://www.zetetic.net/sqlcipher/

Cellebrite

SQLCipher

2 ways to open a database:

SQL statement (PRAGMA)

Native function call

Setting the Key

PRAGMA key	Set the key for use with the database.
sqlite3_key()	C function providing an alternative to `PRAGMA key`.
sqlite3_key_v2()	C function providing an alternative to `PRAGMA key`.

https://www.zetetic.net/sqlcipher/sqlcipher-api/

Cellebrite

SQLCipher >
Setting the key >

PRAGMA key

Execute SQL statement to set the key

Encrypting a database

To specify an encryption passphrase for the database via the SQL interface you use a PRAGMA. The passphrase you enter is passed through PBKDF2 key derivation to obtain the encryption key for the database

```
PRAGMA key = 'passphrase';
```

Alternately, you can specify an exact byte sequence using a blob literal. If you use this method it is your responsibility to ensure that the data you provide is a 64 character hex string, which will be converted directly to 32 bytes (256 bits) of key data without key derivation.

```
PRAGMA key = "x'2DD29CA851E7B56E4697B0E1F08507293D761A05CE4D1B628663F411A8086D99'";
```

To encrypt a database programmatically you can use the `sqlite3_key` function. The data provided in `pKey` is converted to an encryption key according to the same rules as `PRAGMA key`.

```
int sqlite3_key(sqlite3 *db, const void *pKey, int nKey);
```

`PRAGMA key` or `sqlite3_key` should be called as the first operation when a database is open.

https://github.com/sqlcipher/sqlcipher

Cellebrite

SQLCipher >
Setting the key >

PRAGMA key

Execute SQL statement to set the key

Encrypting a database

To specify an encryption passphrase for the database via the SQL interface you use a PRAGMA. The passphrase you enter is passed through PBKDF2 key derivation to obtain the encryption key for the database

```
PRAGMA key = 'passphrase';
```

Alternately, you can specify an exact byte sequence using a blob literal. If you use this method it is your responsibility to ensure that the data you provide is a 64 character hex string, which will be converted directly to 32 bytes (256 bits) of key data without key derivation.

```
PRAGMA key = "x'2DD29CA851E7B56E4697B0E1F08507293D761A05CE4D1B628663F411A8086D99'";
```

To encrypt a database programmatically you can use the `sqlite3_key` function. The data provided in `pKey` is converted to an encryption key according to the same rules as `PRAGMA key`.

```
int sqlite3_key(sqlite3 *db, const void *pKey, int nKey);
```

`PRAGMA key` or `sqlite3_key` should be called as the first operation when a database is open.

https://github.com/sqlcipher/sqlcipher

```
PRAGMA key =
"x'2DD29CA851E7B56E4697B0E1F08507293D761A05CE4D1B628663F411A8086D99'";
```

Cellebrite

SQLCipher >
Setting the key >

sqlite3_key()

Call a function to set the key

Encrypting a database

To specify an encryption passphrase for the database via the SQL interface you use a PRAGMA. The passphrase you enter is passed through PBKDF2 key derivation to obtain the encryption key for the database

```
PRAGMA key = 'passphrase';
```

Alternately, you can specify an exact byte sequence using a blob literal. If you use this method it is your responsibility to ensure that the data you provide is a 64 character hex string, which will be converted directly to 32 bytes (256 bits) of key data without key derivation.

```
PRAGMA key = "x'2DD29CA851E7B56E4697B0E1F08507293D761A05CE4D1B628663F411A8086D99'";
```

To encrypt a database programmatically you can use the `sqlite3_key` function. The data provided in `pKey` is converted to an encryption key according to the same rules as `PRAGMA key`.

```
int sqlite3_key(sqlite3 *db, const void *pKey, int nKey);
```

`PRAGMA key` or `sqlite3_key` should be called as the first operation when a database is open.

https://github.com/sqlcipher/sqlcipher

Cellebrite

SQLCipher >
Setting the key >

sqlite3_key()

Call a function to set the key

For example:

Encrypting a database

To specify an encryption passphrase for the database via the SQL interface you use a PRAGMA. The passphrase you enter is passed through PBKDF2 key derivation to obtain the encryption key for the database

```
PRAGMA key = 'passphrase';
```

Alternately, you can specify an exact byte sequence using a blob literal. If you use this method it is your responsibility to ensure that the data you provide is a 64 character hex string, which will be converted directly to 32 bytes (256 bits) of key data without key derivation.

```
PRAGMA key = "x'2DD29CA851E7B56E4697B0E1F08507293D761A05CE4D1B628663F411A8086D99'";
```

To encrypt a database programmatically you can use the `sqlite3_key` function. The data provided in `pKey` is converted to an encryption key according to the same rules as `PRAGMA key`.

```
int sqlite3_key(sqlite3 *db, const void *pKey, int nKey);
```

`PRAGMA key` or `sqlite3_key` should be called as the first operation when a database is open.

https://github.com/sqlcipher/sqlcipher

```
char * key = "very strong passphrase";
sqlite3_key(my_database, key, strlen(key));
```

Cellebrite

SQLCipher >

Setting the key

Important note

Encrypting a database

To specify an encryption passphrase for the database via the SQL interface you use a PRAGMA. The passphrase you enter is passed through PBKDF2 key derivation to obtain the encryption key for the database

```
PRAGMA key = 'passphrase';
```

Alternately, you can specify an exact byte sequence using a blob literal. If you use this method it is your responsibility to ensure that the data you provide is a 64 character hex string, which will be converted directly to 32 bytes (256 bits) of key data without key derivation.

```
PRAGMA key = "x'2DD29CA851E7B56E4697B0E1F08507293D761A05CE4D1B628663F411A8086D99'";
```

To encrypt a database programmatically you can use the `sqlite3_key` function. The data provided in `pKey` is converted to an encryption key according to the same rules as `PRAGMA key`.

```
int sqlite3_key(sqlite3 *db, const void *pKey, int nKey);
```

`PRAGMA key` or `sqlite3_key` should be called as the first operation when a database is open.

https://github.com/sqlcipher/sqlcipher

`PRAGMA key` or `sqlite3_key` should be called as the first operation when a database is open.

Cellebrite

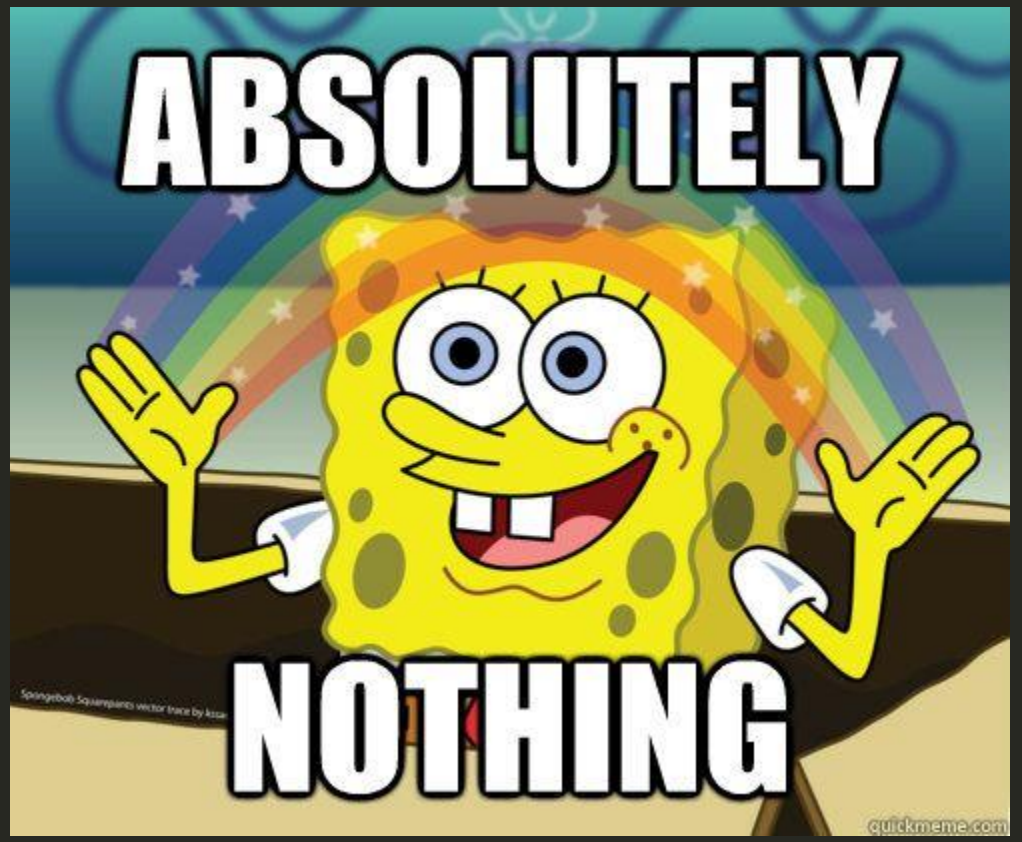
Tracing sqlite3_key()

```
c:\Users\orb
λ frida-ps | grep Wickr
23588  WickrMe.exe

c:\Users\orb
λ frida-trace WickrMe.exe -i sqlite3_key
Started tracing 0 functions. Press Ctrl+C to stop.
```

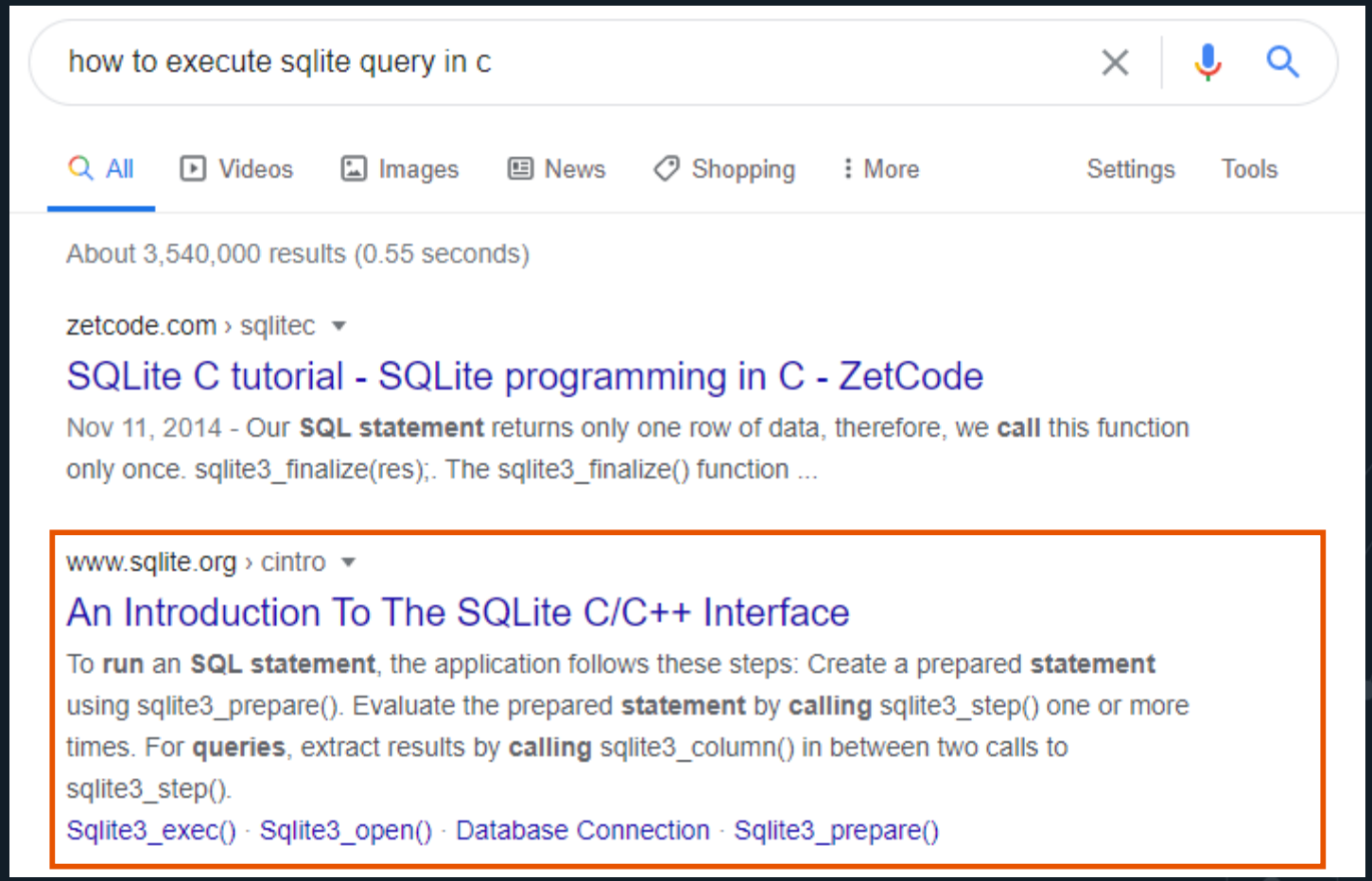
Cellebrite

Tracing sqlite3_key()

```
c:\Users\orb
λ frida-ps | grep Wickr
23588  WickrMe.exe

c:\Users\orb
λ frida-trace WickrMe.exe -i sqlite3_key
Started tracing 0 functions. Press Ctrl+C to stop.
```



Cellebrite


how to execute sqlite query in c
All
Videos
Images
News
Shopping
More
Settings
Tools
About 3,540,000 results (0.55 seconds)
zetcode.com › sqlitec
SQLite C tutorial - SQLite programming in C - ZetCode
Nov 11, 2014 - Our SQL statement returns only one row of data, therefore, we call this function only once. sqlite3_finalize(res);. The sqlite3_finalize() function ...
www.sqlite.org › cintro
An Introduction To The SQLite C/C++ Interface
To run an SQL statement, the application follows these steps: Create a prepared statement using sqlite3_prepare(). Evaluate the prepared statement by calling sqlite3_step() one or more times. For queries, extract results by calling sqlite3_column() in between two calls to sqlite3_step().
Sqlite3_exec() · Sqlite3_open() · Database Connection · Sqlite3_prepare()

Cellebrite

Tracing sqlite3_exec()

“A wrapper function that does sqlite3_prepare(), sqlite3_step(), sqlite3_column(), sqlite3_finalize() for a string of one or more SQL statements.”

https://www.sqlite.org/cintro.html

```
c:\Users\orb
λ frida-trace WickrMe.exe -i sqlite3_exec
Instrumenting functions...
sqlite3_exec: Loaded handler at "c:\\Users\\orb\\__handlers__\\StateRepository.Core.dll\\sqlite3_exec.js"
Started tracing 1 function. Press Ctrl+C to stop.
```

Cellebrite

Tracing sqlite3_exec()

“A wrapper function that does sqlite3_prepare(), sqlite3_step(), sqlite3_column(), sqlite3_finalize() for a string of one or more SQL statements.”

https://www.sqlite.org/cintro.html

```
c:\Users\orb
λ frida-trace WickrMe.exe -i sqlite3_exec
Instrumenting functions...
sqlite3_exec: Loaded handler at "c:\\Users\\orb\\__handlers__\\StateRepository.Core.dll\\sqlite3_exec.js"
Started tracing 1 function. Press Ctrl+C to stop.
```

Cellebrite

Tracing sqlite3_step()

“Advance an sqlite3_stmt to the next result row or to completion.”

https://www.sqlite.org/cintro.html

```
c:\Users\orb
λ frida-trace WickrMe.exe -i sqlite3_exec
Instrumenting functions...
sqlite3_exec: Loaded handler at "c:\\Users\\orb\\__handlers__\\StateRepository.Core.dll\\sqlite3_exec.js"
Started tracing 1 function. Press Ctrl+C to stop.

c:\Users\orb
λ frida-trace WickrMe.exe -i sqlite3_step
Instrumenting functions...
sqlite3_step: Loaded handler at "c:\\Users\\orb\\__handlers__\\StateRepository.Core.dll\\sqlite3_step.js"
Started tracing 1 function. Press Ctrl+C to stop.
```

Cellebrite

Tracing sqlite3_step()

“Advance an sqlite3_stmt to the next result row or to completion.”

https://www.sqlite.org/cintro.html

```
c:\Users\orb
λ frida-trace WickrMe.exe -i sqlite3_exec
Instrumenting functions...
sqlite3_exec: Loaded handler at "c:\\Users\\orb\\__handlers__\\StateRepository.Core.dll\\sqlite3_exec.js"
Started tracing 1 function. Press Ctrl+C to stop.

c:\Users\orb
λ frida-trace WickrMe.exe -i sqlite3_step
Instrumenting functions...
sqlite3_step: Loaded handler at "c:\\Users\\orb\\__handlers__\\StateRepository.Core.dll\\sqlite3_step.js"
Started tracing 1 function. Press Ctrl+C to stop.
```

Cellebrite

Brute force?

Trace everything SQL related

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*sql*" -i "*Sql*" > frida_sql.log
```

frida_sql.log

```
Started tracing 710 functions. Press Ctrl+C to stop.
           /* TID 0x4f94 */
  5214 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
           /* TID 0x2108 */
  5215 ms  ?isNull@QSqlField@@QEBA_NXZ()
  5215 ms  ?isNull@QSqlField@@QEBA_NXZ()
  5698 ms  ?isNull@QSqlField@@QEBA_NXZ()
  5698 ms  ?isNull@QSqlField@@QEBA_NXZ()
           /* TID 0x4f94 */
 65014 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65015 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65015 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65016 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65016 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65017 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65017 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65018 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65018 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65018 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65018 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65019 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65019 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65020 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65021 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65022 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65022 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65023 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65023 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65024 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65024 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65024 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65025 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65025 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
```

Cellebrite

Brute force?

Trace everything SQL related

Lots of garbage to filter…

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*sql*" -i "*Sql*" > frida_sql.log
```

frida_sql.log

```
Started tracing 710 functions. Press Ctrl+C to stop.
           /* TID 0x4f94 */
  5214 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
           /* TID 0x2108 */
  5215 ms  ?isNull@QSqlField@@QEBA_NXZ()
  5215 ms  ?isNull@QSqlField@@QEBA_NXZ()
  5698 ms  ?isNull@QSqlField@@QEBA_NXZ()
  5698 ms  ?isNull@QSqlField@@QEBA_NXZ()
           /* TID 0x4f94 */
 65014 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65015 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65015 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65016 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65016 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65017 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65017 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65018 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65018 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65018 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65018 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65019 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65019 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65020 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65021 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65022 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65022 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65023 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65023 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65024 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65024 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65024 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65025 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
 65025 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@QMetaObject@@HPEAPEAX@Z()
```

Cellebrite

Brute force.

What happens right after we log in?

```
 /* TID 0x4cd4 */
 11805 ms  ??0QSqlDatabase@@QEAA@XZ()
 11805 ms  ?addDatabase@QSqlDatabase@@SA?AV1@AEBVQString@@0@Z()
 11808 ms     | ??0QSqlDriverPlugin@@QEAA@PEAVQObject@@@Z()
 11808 ms     | ??0QSqlError@@QEAA@AEBVQString@@...()
 11808 ms     | ??0QSqlDriver@@IEAA@AEAVQSqlDriverPrivate@@...()
 11808 ms  ??4QSqlDatabase@@QEAAAEAV0@AEBV0@@Z()
 11808 ms  ??1QSqlDatabase@@QEAA@XZ()
 11808 ms  ?setDatabaseName@QSqlDatabase@@QEAAXAEBVQString@@@Z()
 11808 ms  ?open@QSqlDatabase@@QEAA_NXZ()
 11808 ms     | ?isOpen@QSqlDriver@@UEBA_NXZ()
/* TID 0x3794 */
 11808 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@...()
/* TID 0x4cd4 */
 11809 ms     | ?setOpen@QSqlDriver@@MEAAX_N@Z()
 11809 ms     | ?setOpenError@QSqlDriver@@MEAAX_N@Z()
 11809 ms  ??0QSqlDatabase@@QEAA@AEBV0@@Z()
 11809 ms  ??0QSqlQuery@@QEAA@VQSqlDatabase@@@Z()
 11809 ms     | ??0QSqlDriver@@QEAA@PEAVQObject@@@Z()
 11809 ms     |    | ??0QSqlError@@QEAA@AEBVQString@@...()
 11809 ms     | ??0QSqlError@@QEAA@AEBVQString@@0W4ErrorType@0@0@Z()
 ...
 11809 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
 11809 ms     | ?clear@QSqlResult@@IEAAXXZ()
 11809 ms     |    | ?clearIndex@QSqlResultPrivate@@QEAAXXZ()
 11810 ms     | ?setActive@QSqlResult@@MEAAX_N@Z()
 11810 ms     | ??0QSqlError@@QEAA@AEBVQString@@0W4ErrorType@0@0@Z()
 11810 ms     | ?setLastError@QSqlResult@@MEAAXAEBVQSqlError@@@Z()
 11810 ms     |    | ??4QSqlError@@QEAAAEAV0@AEBV0@@Z()
 11810 ms     | ??1QSqlError@@QEAA@XZ()
 11810 ms     | ?setAt@QSqlResult@@MEAAXH@Z()
```

Cellebrite

Brute force.

What happens right after we log in?

Needles in the haystack

```
 /* TID 0x4cd4 */
 11805 ms  ??0QSqlDatabase@@QEAA@XZ()
 11805 ms  ?addDatabase@QSqlDatabase@@SA?AV1@AEBVQString@@0@Z()
 11808 ms     | ??0QSqlDriverPlugin@@QEAA@PEAVQObject@@@Z()
 11808 ms     | ??0QSqlError@@QEAA@AEBVQString@@...()
 11808 ms     | ??0QSqlDriver@@IEAA@AEAVQSqlDriverPrivate@@...()
 11808 ms  ??4QSqlDatabase@@QEAAAEAV0@AEBV0@@Z()
 11808 ms  ??1QSqlDatabase@@QEAA@XZ()
 11808 ms  ?setDatabaseName@QSqlDatabase@@QEAAXAEBVQString@@@Z()
 11808 ms  ?open@QSqlDatabase@@QEAA_NXZ()
 11808 ms     | ?isOpen@QSqlDriver@@UEBA_NXZ()
/* TID 0x3794 */
 11808 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@...()
/* TID 0x4cd4 */
 11809 ms     | ?setOpen@QSqlDriver@@MEAAX_N@Z()
 11809 ms     | ?setOpenError@QSqlDriver@@MEAAX_N@Z()
 11809 ms  ??0QSqlDatabase@@QEAA@AEBV0@@Z()
 11809 ms  ??0QSqlQuery@@QEAA@VQSqlDatabase@@@Z()
 11809 ms     | ??0QSqlDriver@@QEAA@PEAVQObject@@@Z()
 11809 ms     |    | ??0QSqlError@@QEAA@AEBVQString@@...()
 11809 ms     | ??0QSqlError@@QEAA@AEBVQString@@0W4ErrorType@0@0@Z()
 ...
 11809 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
 11809 ms     | ?clear@QSqlResult@@IEAAXXZ()
 11809 ms     |    | ?clearIndex@QSqlResultPrivate@@QEAAXXZ()
 11810 ms     | ?setActive@QSqlResult@@MEAAX_N@Z()
 11810 ms     | ??0QSqlError@@QEAA@AEBVQString@@0W4ErrorType@0@0@Z()
 11810 ms     | ?setLastError@QSqlResult@@MEAAXAEBVQSqlError@@@Z()
 11810 ms     |    | ??4QSqlError@@QEAAAEAV0@AEBV0@@Z()
 11810 ms     | ??1QSqlError@@QEAA@XZ()
 11810 ms     | ?setAt@QSqlResult@@MEAAXH@Z()
```

Cellebrite

Internal function calls

Brute force.

What happens right after we log in?

Needles in the haystack

```
/* TID 0x4cd4 */
 11805 ms  ??0QSqlDatabase@@QEAA@XZ()
 11805 ms  ?addDatabase@QSqlDatabase@@SA?AV1@AEBVQString@@0@Z()
 11808 ms     | ??0QSqlDriverPlugin@@QEAA@PEAVQObject@@@Z()
 11808 ms     | ??0QSqlError@@QEAA@AEBVQString@@...()
 11808 ms     | ??0QSqlDriver@@IEAA@AEAVQSqlDriverPrivate@@...()
 11808 ms  ??4QSqlDatabase@@QEAAAEAV0@AEBV0@@Z()
 11808 ms  ??1QSqlDatabase@@QEAA@XZ()
 11808 ms  ?setDatabaseName@QSqlDatabase@@QEAAXAEBVQString@@@Z()
 11808 ms  ?open@QSqlDatabase@@QEAA_NXZ()
 11808 ms     | ?isOpen@QSqlDriver@@UEBA_NXZ()
/* TID 0x3794 */
 11808 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@...()
/* TID 0x4cd4 */
 11809 ms     | ?setOpen@QSqlDriver@@MEAAX_N@Z()
 11809 ms     | ?setOpenError@QSqlDriver@@MEAAX_N@Z()
 11809 ms  ??0QSqlDatabase@@QEAA@AEBV0@@Z()
 11809 ms  ??0QSqlQuery@@QEAA@VQSqlDatabase@@@Z()
 11809 ms     | ??0QSqlDriver@@QEAA@PEAVQObject@@@Z()
 11809 ms     |    | ??0QSqlError@@QEAA@AEBVQString@@...()
 11809 ms     | ??0QSqlError@@QEAA@AEBVQString@@0W4ErrorType@0@0@Z()
 ...
 11809 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
 11809 ms     | ?clear@QSqlResult@@IEAAXXZ()
 11809 ms     |    | ?clearIndex@QSqlResultPrivate@@QEAAXXZ()
 11810 ms     | ?setActive@QSqlResult@@MEAAX_N@Z()
 11810 ms     | ??0QSqlError@@QEAA@AEBVQString@@0W4ErrorType@0@0@Z()
 11810 ms     | ?setLastError@QSqlResult@@MEAAXAEBVQSqlError@@@Z()
 11810 ms     |    | ??4QSqlError@@QEAAAEAV0@AEBV0@@Z()
 11810 ms     | ??1QSqlError@@QEAA@XZ()
 11810 ms     | ?setAt@QSqlResult@@MEAAXH@Z()
```

Cellebrite

SqlDatabase addDatabase setDatabaseName

All Images Videos Maps News More Settings Tools

About 8,860 results (0.61 seconds)

Did you mean: ***Sql Database add Database*** setDatabaseName

doc.qt.io › qsqldatabase

QSqlDatabase Class | Qt SQL 5.14.2 - Qt Documentation

QSqlDatabase, **addDatabase**(QSqlDriver *driver, const QString ... set the connection parameters with **setDatabaseName**(), setUserName(), setPassword(), setHostName(), ... See also **SQL Database** Drivers, registerSqlDriver(), and drivers().

QSqlQuery · QSqlDriver · QSqlError

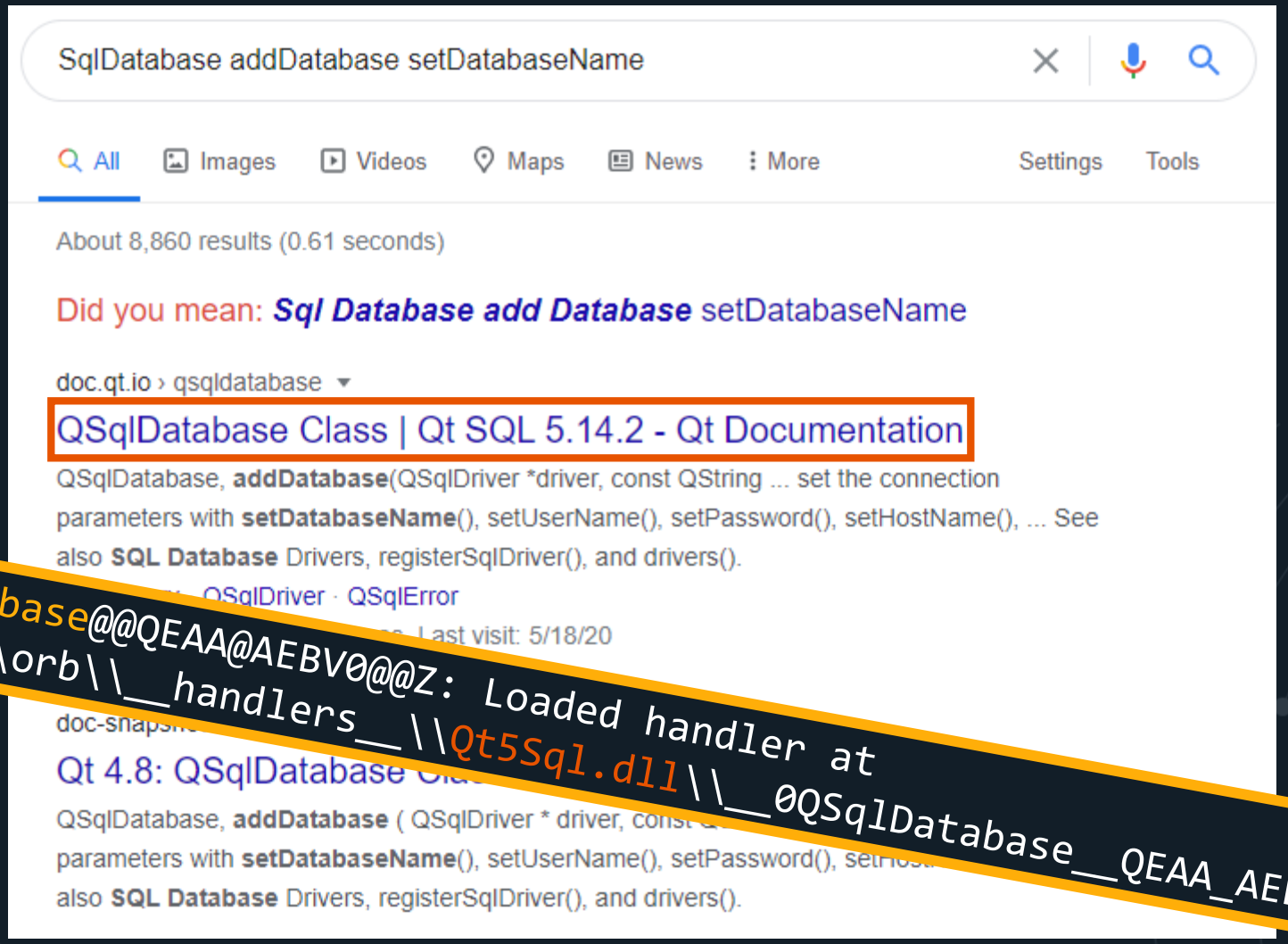
You've visited this page 2 times. Last visit: 5/18/20

doc-snapshots.qt.io › qsqldatabase

Qt 4.8: QSqlDatabase Class Reference

QSqlDatabase, **addDatabase** (QSqlDriver * driver, const QString ... set the connection parameters with **setDatabaseName**(), setUserName(), setPassword(), setHostName(), ... See also **SQL Database** Drivers, registerSqlDriver(), and drivers().

Cellebrite




```
??0QSqlDatabase@@QEAA@AEBV0@@Z: Loaded handler at
"c:\\Users\\orb\\__handlers__\\Qt5Sql.dll\\__0QSqlDatabase__QEAA_AEBV0__Z.js"
```

Cellebrite

QSqlDatabase Class

The QSqlDatabase class handles a connection to a database. More...

Static Public Members

QSqlDatabase	**addDatabase**(const QString &*type*, const QString &*connectionName* = QLatin1String(defaultConnection))
QSqlDatabase	**addDatabase**(QSqlDriver **driver*, const QString &*connectionName* = QLatin1String(defaultConnection))
QSqlDatabase	**cloneDatabase**(const QSqlDatabase &*other*, const QString &*connectionName*)

```
QSqlDatabase db = QSqlDatabase::addDatabase("QPSQL");
db.setHostName("acidalia");
db.setDatabaseName("customdb");
db.setUserName("mojito");
db.setPassword("J0a1m8");
bool ok = db.open();
```

https://doc.qt.io/qt-5/qsqldatabase.html

Cellebrite

Log-in process

`QSqlDatabase` object with functions:

`addDatabase`

`setDatabaseName`

`open`

```
 /* TID 0x4cd4 */
 11805 ms  ??0QSqlDatabase@@QEAA@XZ()
 11805 ms  ?addDatabase@QSqlDatabase@@SA?AV1@AEBVQString@@0@Z()
 11808 ms     | ??0QSqlDriverPlugin@@QEAA@PEAVQObject@@@Z()
 11808 ms     | ??0QSqlError@@QEAA@AEBVQString@@...()
 11808 ms     | ??0QSqlDriver@@IEAA@AEAVQSqlDriverPrivate@@...()
 11808 ms  ??4QSqlDatabase@@QEAAAEAV0@AEBV0@@Z()
 11808 ms  ??1QSqlDatabase@@QEAA@XZ()
 11808 ms  ?setDatabaseName@QSqlDatabase@@QEAAXAEBVQString@@@Z()
 11808 ms  ?open@QSqlDatabase@@QEAA_NXZ()
 11808 ms     | ?isOpen@QSqlDriver@@UEBA_NXZ()
/* TID 0x3794 */
 11808 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@...()
/* TID 0x4cd4 */
 11809 ms     | ?setOpen@QSqlDriver@@MEAAX_N@Z()
 11809 ms     | ?setOpenError@QSqlDriver@@MEAAX_N@Z()
 11809 ms  ??0QSqlDatabase@@QEAA@AEBV0@@Z()
 11809 ms  ??0QSqlQuery@@QEAA@VQSqlDatabase@@@Z()
 11809 ms     | ??0QSqlDriver@@QEAA@PEAVQObject@@@Z()
 11809 ms     |    | ??0QSqlError@@QEAA@AEBVQString@@...()
 11809 ms     | ??0QSqlError@@QEAA@AEBVQString@@0W4ErrorType@0@0@Z()
 ...
 11809 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
 11809 ms     | ?clear@QSqlResult@@IEAAXXZ()
 11809 ms     |    | ?clearIndex@QSqlResultPrivate@@QEAAXXZ()
 11810 ms     | ?setActive@QSqlResult@@MEAAX_N@Z()
 11810 ms     | ??0QSqlError@@QEAA@AEBVQString@@0W4ErrorType@0@0@Z()
 11810 ms     | ?setLastError@QSqlResult@@MEAAXAEBVQSqlError@@@Z()
 11810 ms     |    | ??4QSqlError@@QEAAAEAV0@AEBV0@@Z()
 11810 ms     | ??1QSqlError@@QEAA@XZ()
 11810 ms     | ?setAt@QSqlResult@@MEAAXH@Z()
```

Cellebrite

Log-in process

What about `QSqlQuery`?

```
 /* TID 0x4cd4 */
 11805 ms  ??0QSqlDatabase@@QEAA@XZ()
 11805 ms  ?addDatabase@QSqlDatabase@@SA?AV1@AEBVQString@@0@Z()
 11808 ms     | ??0QSqlDriverPlugin@@QEAA@PEAVQObject@@@Z()
 11808 ms     | ??0QSqlError@@QEAA@AEBVQString@@...()
 11808 ms     | ??0QSqlDriver@@IEAA@AEAVQSqlDriverPrivate@@...()
 11808 ms  ??4QSqlDatabase@@QEAAAEAV0@AEBV0@@Z()
 11808 ms  ??1QSqlDatabase@@QEAA@XZ()
 11808 ms  ?setDatabaseName@QSqlDatabase@@QEAAXAEBVQString@@@Z()
 11808 ms  ?open@QSqlDatabase@@QEAA_NXZ()
 11808 ms     | ?isOpen@QSqlDriver@@UEBA_NXZ()
/* TID 0x3794 */
 11808 ms  ?qt_metacall@QSqlDriverPlugin@@UEAAHW4Call@...()
/* TID 0x4cd4 */
 11809 ms     | ?setOpen@QSqlDriver@@MEAAX_N@Z()
 11809 ms     | ?setOpenError@QSqlDriver@@MEAAX_N@Z()
 11809 ms  ??0QSqlDatabase@@QEAA@AEBV0@@Z()
 11809 ms  ??0QSqlQuery@@QEAA@VQSqlDatabase@@@Z()
 11809 ms     | ??0QSqlDriver@@QEAA@PEAVQObject@@@Z()
 11809 ms     |    | ??0QSqlError@@QEAA@AEBVQString@@...()
 11809 ms     | ??0QSqlError@@QEAA@AEBVQString@@0W4ErrorType@0@0@Z()
 ...
 11809 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
 11809 ms     | ?clear@QSqlResult@@IEAAXXZ()
 11809 ms     |    | ?clearIndex@QSqlResultPrivate@@QEAAXXZ()
 11810 ms     | ?setActive@QSqlResult@@MEAAX_N@Z()
 11810 ms     | ??0QSqlError@@QEAA@AEBVQString@@0W4ErrorType@0@0@Z()
 11810 ms     | ?setLastError@QSqlResult@@MEAAXAEBVQSqlError@@@Z()
 11810 ms     |    | ??4QSqlError@@QEAAAEAV0@AEBV0@@Z()
 11810 ms     | ??1QSqlError@@QEAA@XZ()
 11810 ms     | ?setAt@QSqlResult@@MEAAXH@Z()
```

Cellebrite

qsqlquery

All News Videos Images Maps More Settings Tools

About 64,400 results (0.58 seconds)

doc.qt.io › qsqlquery

QSqlQuery Class | Qt SQL 5.14.2 - Qt Documentation

bool **QSqlQuery**::next(). Retrieves the next record in the result, if available, and positions the query on the retrieved record. Note that the result must be in the ...

You've visited this page 2 times. Last visit: 5/18/20

QSqlQuery

The QSqlQuery class provides a means of executing and ...

More results from qt.io »

doc.qt.io › sql-sqlstatements

Executing SQL Statements | Qt SQL 5.14.2 - Qt Documentation

The **QSqlQuery** class provides an interface for executing SQL statements and navigating through the result set of a query. The QSqlQueryModel and ...

Cellebrite

Executing SQL Statements

The QSqlQuery class provides an interface for executing SQL statements and navigating through the result set of a query.

The QSqlQueryModel and QSqlTableModel classes described in the next section provide a higher-level interface for accessing databases. If you are unfamiliar with SQL, you might want to skip directly to the next section (Using the SQL Model Classes).

Executing a Query

To execute an SQL statement, simply create a QSqlQuery object and call QSqlQuery::exec() like this:

```
QSqlQuery query;
query.exec("SELECT name, salary FROM employee WHERE salary > 50000");
```

The QSqlQuery constructor accepts an optional QSqlDatabase object that specifies which database connection to use. In the example above, we don't specify any connection, so the default connection is used.

https://doc.qt.io/qt-5/sql-sqlstatements.html

Cellebrite

Log-in process

Cleaning up

```
11805 ms  ??0QSqlDatabase@@QEAA@XZ()
11805 ms  ?addDatabase@QSqlDatabase@@SA?AV1@AEBVQString@@0@Z()
11808 ms  ??4QSqlDatabase@@QEAAAEAV0@AEBV0@@Z()
11808 ms  ??1QSqlDatabase@@QEAA@XZ()
11808 ms  ?setDatabaseName@QSqlDatabase@@QEAAXAEBVQString@@@Z()
11808 ms  ?open@QSqlDatabase@@QEAA_NXZ()
11809 ms  ??0QSqlDatabase@@QEAA@AEBV0@@Z()

11809 ms  ??0QSqlQuery@@QEAA@VQSqlDatabase@@@Z()
11809 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
```

Cellebrite

Log-in process

Cleaning up

```
11805 ms  ??0QSqlDatabase@@QEAA@XZ()
11805 ms  ?addDatabase@QSqlDatabase@@SA?AV1@AEBVQString@@0@Z()
11808 ms  ??4QSqlDatabase@@QEAAAEAV0@AEBV0@@Z()
11808 ms  ??1QSqlDatabase@@QEAA@XZ()
11808 ms  ?setDatabaseName@QSqlDatabase@@QEAAXAEBVQString@@@Z()
11808 ms  ?open@QSqlDatabase@@QEAA_NXZ()
11809 ms  ??0QSqlDatabase@@QEAA@AEBV0@@Z()

11809 ms  ??0QSqlQuery@@QEAA@VQSqlDatabase@@@Z()
11809 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
```

Cellebrite

Log-in process

Cleaning up

```
11805 ms  ??0QSqlDatabase@@QEAA@XZ()
11805 ms  ?addDatabase@QSqlDatabase@@SA?AV1@AEBVQString@@0@Z()
11808 ms  ??4QSqlDatabase@@QEAAAEAV0@AEBV0@@Z()
11808 ms  ??1QSqlDatabase@@QEAA@XZ()
11808 ms  ?setDatabaseName@QSqlDatabase@@QEAAXAEBVQString@@@Z()
11808 ms  ?open@QSqlDatabase@@QEAA_NXZ()
11809 ms  ??0QSqlDatabase@@QEAA@AEBV0@@Z()

11809 ms  ??0QSqlQuery@@QEAA@VQSqlDatabase@@@Z()
11809 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
```

```
11805 ms  QSqlDatabase()
11805 ms  QSqlDatabase.addDatabase(QString)
11808 ms  QSqlDatabase.setDatabaseName(QString)
11808 ms  QSqlDatabase.open()

11809 ms  QSqlQuery(QSqlDatabase)
11809 ms  QSqlQuery.exec(QString)
```

Cellebrite

Log-in proce

Cleaning up

```
11805 ms  ??0QSqlDatabase@@QEAA@XZ()
11805 ms  ?addDatabase@QSqlDatabase@@SA?AV1@AEBVQString@@0@Z()
11808 ms  ??4QSqlDatabase@@QEAAAEAV0@AEBV0@@Z()
11808 ms  ??1QSqlDatabase@@QEAA@XZ()
                                    @@QEAAXAEBVQString@@@Z()
                                    )
                                    Z()

                                    ase@@@Z()
                                    tring@@@Z()

                                    ing)
                                    QString)
11809 ms  QSqlQuery(QSqlDatabase)
11809 ms  QSqlQuery.exec(QString)
```

Cellebrite

Tracing QSqlQuery.exec()

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

Cellebrite

Tracing QSqlQuery.exec()

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log

 /* TID 0x751c */
  2399 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2405 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2405 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2405 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2490 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2490 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2490 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2490 ms  ?exec@QSqlQuery@@QEAA_NXZ()
  2490 ms  ?exec@QSqlQuery@@QEAA_NXZ()
  2490 ms  ?exec@QSqlQuery@@QEAA_NXZ()
  2490 ms  ?exec@QSqlQuery@@QEAA_NXZ()
  2492 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2513 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2514 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2514 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2514 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2514 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2514 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
```

Cellebrite

Tracing `QSqlQuery.exec()`

Note which function overload gets called first

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log

 /* TID 0x751c */
  2399 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2405 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2405 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2405 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2490 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2490 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2490 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2490 ms  ?exec@QSqlQuery@@QEAA_NXZ()
  2490 ms  ?exec@QSqlQuery@@QEAA_NXZ()
  2490 ms  ?exec@QSqlQuery@@QEAA_NXZ()
  2490 ms  ?exec@QSqlQuery@@QEAA_NXZ()
  2492 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2513 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2514 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2514 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2514 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2514 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
  2514 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
```

Cellebrite

Tracing QSqlQuery.exec()

Editing the handler

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log

onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');



},
```

Cellebrite

Tracing QSqlQuery.exec()

- Editing the handler
- Adding logging
- Guessing the number of arguments

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        for (var i = 0; i < 5; i++) {
                log(args[i]);
        }
},
```

Cellebrite

Tracing QSqlQuery.exec()

- Editing the handler
- Adding logging
- Guessing the number of arguments

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        for (var i = 0; i < 5; i++) {
                log(args[i]);
        }
},
```

```
5744 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
5744 ms  0x15c6fb960
5744 ms  0x15c6fb948
5744 ms  0x1582e5212c0
5744 ms  0x1
5744 ms  0x1584df6a9e0

5744 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
5744 ms  0x15c6fb950
5744 ms  0x15c6fba28
5744 ms  0x165
5744 ms  0x40
5744 ms  0x158497ab098

5744 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
5744 ms  0x15c6fb940
5744 ms  0x15c6fb930
5744 ms  0x175
5744 ms  0x40
5744 ms  0x15849741bf8
```

Cellebrite

Tracing QSqlQuery.exec()

Adding logging

Guessing the number of arguments

```
bool QSqlQuery::exec(const QString &query)
```

These look like memory addresses

These don't

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        for (var i = 0; i < 5; i++) {
                log(args[i]);
        }
},
```

```
5744 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
5744 ms  0x15c6fb960
5744 ms  0x15c6fb948
5744 ms  0x1582e5212c0
5744 ms  0x1
5744 ms  0x1584df6a9e0

5744 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
5744 ms  0x15c6fb950
5744 ms  0x15c6fba28
5744 ms  0x165
5744 ms  0x40
5744 ms  0x158497ab098

5744 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
5744 ms  0x15c6fb940
5744 ms  0x15c6fb930
5744 ms  0x175
5744 ms  0x40
5744 ms  0x15849741bf8
```

Cellebrite

Tracing QSqlQuery.exec()

Digging into the first 2 arguments

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[0], {length: 64}));
        log(hexdump(args[1], {length: 64}));
},
```

Cellebrite

Tracing QSqlQuery.exec()

Digging into the first 2 arguments

No SQL queries :(

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[0], {length: 64}));
        log(hexdump(args[1], {length: 64}));
},
```

```
 5656 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()

            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
15c6fb960  60 0f c6 40 58 01 00 00 80 91 ed 2b fa 7f 00 00  `..@X......+....
15c6fb970  40 ad 75 4e 58 01 00 00 fe ff ff ff ff ff ff ff  @.uNX...........
15c6fb980  00 00 00 00 00 00 00 00 a5 0f a6 2b fa 7f 00 00  ...........+....
15c6fb990  00 00 f0 0f 00 00 00 00 95 f8 96 2b fa 7f 00 00  ...........+....

            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
15c6fb948  b0 66 5c 4e 58 01 00 00 b0 de a8 4a 58 01 00 00  .f\NX......JX...
15c6fb958  f0 da a8 4a 58 01 00 00 60 0f c6 40 58 01 00 00  ...JX...`..@X...
15c6fb968  80 91 ed 2b fa 7f 00 00 40 ad 75 4e 58 01 00 00  ...+....@.uNX...
15c6fb978  fe ff ff ff ff ff ff ff 00 00 00 00 00 00 00 00  ................
```

Cellebrite

Tracing QSqlQuery.exec()

Digging into the first 2 arguments

No SQL queries :(

More pointers?

0x15840c60f60

0x15840c60f60

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[0], {length: 64}));
        log(hexdump(args[1], {length: 64}));
},
```

```
 5656 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()

            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
15c6fb960  60 0f c6 40 58 01 00 00 80 91 ed 2b fa 7f 00 00  `..@X......+....
15c6fb970  40 ad 75 4e 58 01 00 00 fe ff ff ff ff ff ff ff  @.uNX...........
15c6fb980  00 00 00 00 00 00 00 00 a5 0f a6 2b fa 7f 00 00  ...........+....
15c6fb990  00 00 f0 0f 00 00 00 00 95 f8 96 2b fa 7f 00 00  ...........+....

            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
15c6fb948  b0 66 5c 4e 58 01 00 00 b0 de a8 4a 58 01 00 00  .f\NX......JX...
15c6fb958  f0 da a8 4a 58 01 00 00 60 0f c6 40 58 01 00 00  ...JX...`..@X...
15c6fb968  80 91 ed 2b fa 7f 00 00 40 ad 75 4e 58 01 00 00  ...+....@.uNX...
15c6fb978  fe ff ff ff ff ff ff ff 00 00 00 00 00 00 00 00  ................
```

Cellebrite

Tracing QSqlQuery.exe

Digging into the first arguments

No SQL queries :(

More pointers?

0x15840c60f60

0x15840c60f60

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log

onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[0], {length: 64}));
        log(hexdump(args[1], {length: 64}));
```


```
tring@@@Z()

           9  A  B  C  D  E  F  0123456789ABCDEF
1 ed 2b fa 7f 00 00  `..@X......+....
f ff ff ff ff ff ff  @.uNX...........
f a6 2b fa 7f 00 00  ..........+....
8 96 2b fa 7f 00 00  ..........+....

           9  A  B  C  D  E  F  0123456789ABCDEF
e a8 4a 58 01 00 00  .f\NX......JX...
f c6 40 58 01 00 00  ...JX...`..@X...
15c6fb968  80 91 ed 2b fa 7f 00 00 40 ad 75 4e 58 01 00 00  ...+....@.uNX...
15c6fb978  fe ff ff ff ff ff ff ff 00 00 00 00 00 00 00 00  ................
```

Cellebrite

Tracing QSqlQuery.exec()

Digging 2 pointers down into the first 2 arguments

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[0].readPointer(),
                    {length: 64}));
        log(hexdump(args[1] readPointer(),
                    {length: 64}));
},
```

Cellebrite

Tracing QSqlQuery.exec()

Digging 2 pointers down into the first 2 arguments

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[0].readPointer(),
                    {length: 64}));
        log(hexdump(args[1] readPointer(),
                    {length: 64}));
},
```

```
 5256 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()

              0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
15849b6c440  01 00 00 00 00 00 00 00 20 c8 b6 49 58 01 00 00  ........ ..IX...
15849b6c450  00 00 00 00 00 00 00 00 36 9d f1 69 00 16 05 88  ........6..i....
15849b6c460  53 63 6f 74 74 73 64 61 60 c4 b6 49 58 01 00 00  Scottsda`..IX...
15849b6c470  00 00 00 00 00 00 00 80 34 9d f3 69 00 17 05 80  ........4..i....

              0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
1584e4d68d0  01 00 00 00 54 00 00 00 74 00 00 00 01 00 00 00  ....T...t.......
1584e4d68e0  18 00 00 00 00 00 00 00 50 00 52 00 41 00 47 00  ........P.R.A.G.
1584e4d68f0  4d 00 41 00 20 00 6b 00 65 00 79 00 20 00 3d 00  M.A. .k.e.y. .=.
1584e4d6900  20 00 22 00 78 00 27 00 30 00 30 00 65 00 62 00   .".x.'.0.0.e.b.
```

Cellebrite

Tracing QSqlQuery.exec()

Digging 2 pointers down into the first 2 arguments

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[0].readPointer(),
                    {length: 64}));
        log(hexdump(args[1] readPointer(),
                    {length: 64}));
},
```

```
 5256 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()

              0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
15849b6c440  01 00 00 00 00 00 00 00 20 c8 b6 49 58 01 00 00  ........ ..IX...
15849b6c450  00 00 00 00 00 00 00 00 36 9d f1 69 00 16 05 88  ........6..i....
15849b6c460  53 63 6f 74 74 73 64 61 60 c4 b6 49 58 01 00 00  Scottsda`..IX...
15849b6c470  00 00 00 00 00 00 00 80 34 9d f3 69 00 17 05 80  ........4..i....

              0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
1584e4d68d0  01 00 00 00 54 00 00 00 74 00 00 00 01 00 00 00  ....T...t.......
1584e4d68e0  18 00 00 00 00 00 00 00 50 00 52 00 41 00 47 00  ........P.R.A.G.
1584e4d68f0  4d 00 41 00 20 00 6b 00 65 00 79 00 20 00 3d 00  M.A. .k.e.y. .=.
1584e4d6900  20 00 22 00 78 00 27 00 30 00 30 00 65 00 62 00   .".x.'.0.0.e.b.
```

Cellebrite

Tracing QSqlQuery.exec()

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[0].readPointer(),
                    {length: 64}));
        log(hexdump(args[1] readPointer(),
                    {length: 64}));
},
```

```
 5256 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
```

$$\left(\left(32\frac{bytes}{key}\cdot 2\frac{chars}{hex\ byte}\right)+18\frac{chars}{PRAGMA\ key = "x''"}\right)\cdot 2\frac{bytes}{UTF\ 16\ char}+0x18\frac{bytes}{header}=188\ bytes$$

```
                 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
1584e4d68d0   01 00 00 00 54 00 00 00 74 00 00 00 01 00 00 00  ....T...t.......
1584e4d68e0   18 00 00 00 00 00 00 00 50 00 52 00 41 00 47 00  ........P.R.A.G.
1584e4d68f0   4d 00 41 00 20 00 6b 00 65 00 79 00 20 00 3d 00  M.A. .k.e.y. .=.
1584e4d6900   20 00 22 00 78 00 27 00 30 00 30 00 65 00 62 00   .".x.'.0.0.e.b.
```

```
CDEF
X...
....
X...
....
```

Cellebrite

Tracing QSqlQuery.exec()

Cleaning up

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[1].readPointer(),
                    {length: 256}));
},
```

Cellebrite

Tracing QSqlQuery.exec()

Cleaning up

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

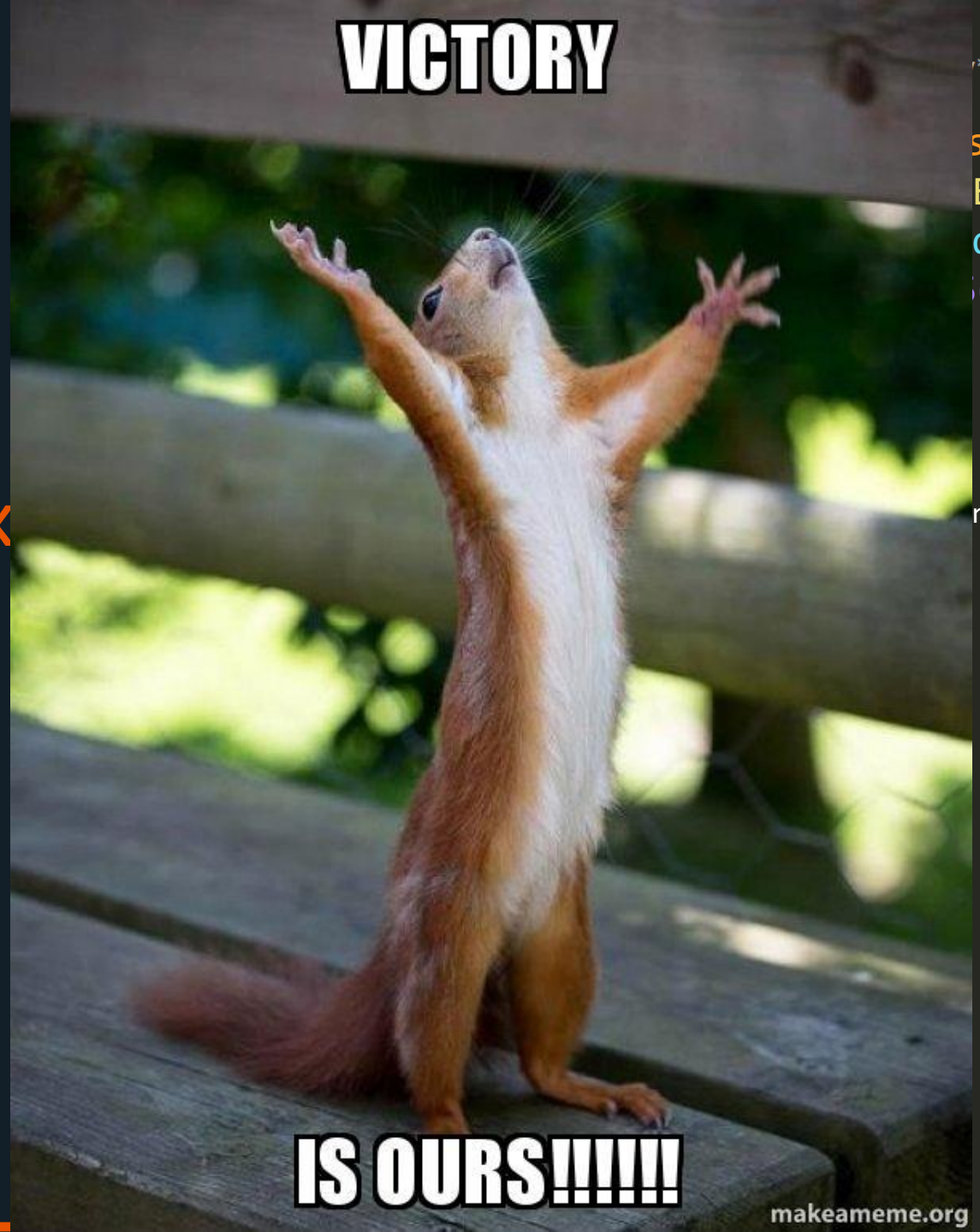
```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[1].readPointer(),
                    {length: 256}));
},
```

```
 7675 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
             0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
1584df38180  01 00 00 00 54 00 00 00 74 00 00 00 00 00 00 00  ....T...t.......
1584df38190  18 00 00 00 00 00 00 00 50 00 52 00 41 00 47 00  ........P.R.A.G.
1584df381a0  4d 00 41 00 20 00 6b 00 65 00 79 00 20 00 3d 00  M.A. .k.e.y. .=.
1584df381b0  20 00 22 00 78 00 27 00 30 00 30 00 65 00 62 00   .".x.'.0.0.e.b.
1584df381c0  36 00 32 00 65 00 36 00 33 00 38 00 39 00 36 00  6.2.e.6.3.8.9.6.
1584df381d0  39 00 33 00 37 00 62 00 37 00 33 00 30 00 31 00  9.3.7.b.7.3.0.1.
1584df381e0  62 00 30 00 64 00 61 00 31 00 66 00 61 00 63 00  b.0.d.a.1.f.a.c.
1584df381f0  66 00 38 00 61 00 31 00 32 00 31 00 38 00 36 00  f.8.a.1.2.1.8.6.
1584df38200  65 00 64 00 30 00 33 00 65 00 61 00 39 00 38 00  e.d.0.3.e.a.9.8.
1584df38210  62 00 32 00 61 00 34 00 63 00 62 00 31 00 36 00  b.2.a.4.c.b.1.6.
1584df38220  63 00 36 00 38 00 66 00 34 00 35 00 39 00 33 00  c.6.8.f.4.5.9.3.
1584df38230  39 00 33 00 65 00 31 00 66 00 63 00 27 00 22 00  9.3.e.1.f.c.'.".
1584df38240  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584df38250  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584df38260  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584df38270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Cellebrite

Tracing QSqlQuery.ex

Cleaning up




```
*" > frida_exec.log

state) {
EAA_NAEBVQString@@@Z()');
dPointer(),
6}));
```

```
ng@@@Z()
   A  B  C  D  E  F  0123456789ABCDEF
 00 00 00 00 00 00  ....T...t.......
 52 00 41 00 47 00  ........P.R.A.G.
 79 00 20 00 3d 00  M.A. .k.e.y. .=.
 30 00 65 00 62 00   ."x.'.0.0.e.b.
 38 00 39 00 36 00  6.2.e.6.3.8.9.6.
 33 00 30 00 31 00  9.3.7.b.7.3.0.1.
 66 00 61 00 63 00  b.0.d.a.1.f.a.c.
 31 00 38 00 36 00  f.8.a.1.2.1.8.6.
 61 00 39 00 38 00  e.d.0.3.e.a.9.8.
 62 00 31 00 36 00  b.2.a.4.c.b.1.6.
 35 00 39 00 33 00  c.6.8.f.4.5.9.3.
 63 00 27 00 22 00  9.3.e.1.f.c.'.".
 00 00 00 00 00 00  ................
 00 00 00 00 00 00  ................
 00 00 00 00 00 00  ................
 00 00 00 00 00 00  ................
```

Cellebrite

Tracing QSqlQuery.exec()

Cleaning up even more

Total string length is 84 (0x54) characters

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[1].readPointer(),
                    {length: 256}));
},
```

```
 7675 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
             0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
1584df38180  01 00 00 00 54 00 00 00 74 00 00 00 00 00 00 00  ....T...t.......
1584df38190  18 00 00 00 00 00 00 00 50 00 52 00 41 00 47 00  ........P.R.A.G.
1584df381a0  4d 00 41 00 20 00 6b 00 65 00 79 00 20 00 3d 00  M.A. .k.e.y. .=.
1584df381b0  20 00 22 00 78 00 27 00 30 00 30 00 65 00 62 00   .".x.'.0.0.e.b.
1584df381c0  36 00 32 00 65 00 36 00 33 00 38 00 39 00 36 00  6.2.e.6.3.8.9.6.
1584df381d0  39 00 33 00 37 00 62 00 37 00 33 00 30 00 31 00  9.3.7.b.7.3.0.1.
1584df381e0  62 00 30 00 64 00 61 00 31 00 66 00 61 00 63 00  b.0.d.a.1.f.a.c.
1584df381f0  66 00 38 00 61 00 31 00 32 00 31 00 38 00 36 00  f.8.a.1.2.1.8.6.
1584df38200  65 00 64 00 30 00 33 00 65 00 61 00 39 00 38 00  e.d.0.3.e.a.9.8.
1584df38210  62 00 32 00 61 00 34 00 63 00 62 00 31 00 36 00  b.2.a.4.c.b.1.6.
1584df38220  63 00 36 00 38 00 66 00 34 00 35 00 39 00 33 00  c.6.8.f.4.5.9.3.
1584df38230  39 00 33 00 65 00 31 00 66 00 63 00 27 00 22 00  9.3.e.1.f.c.'.".
1584df38240  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584df38250  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584df38260  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584df38270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Cellebrite

Tracing QSqlQuery.exec()

Cleaning up even more

Total string length is 84 (0x54) characters

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[1].readPointer(),
                    {length: 256}));
},
```

```
 7675 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
             0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
1584df38180  01 00 00 00 54 00 00 00 74 00 00 00 00 00 00 00  ....T...t.......
1584df38190  18 00 00 00 00 00 00 00 50 00 52 00 41 00 47 00  ........P.R.A.G.
1584df381a0  4d 00 41 00 20 00 6b 00 65 00 79 00 20 00 3d 00  M.A. .k.e.y. .=.
1584df381b0  20 00 22 00 78 00 27 00 30 00 30 00 65 00 62 00   .".x.'.0.0.e.b.
1584df381c0  36 00 32 00 65 00 36 00 33 00 38 00 39 00 36 00  6.2.e.6.3.8.9.6.
1584df381d0  39 00 33 00 37 00 62 00 37 00 33 00 30 00 31 00  9.3.7.b.7.3.0.1.
1584df381e0  62 00 30 00 64 00 61 00 31 00 66 00 61 00 63 00  b.0.d.a.1.f.a.c.
1584df381f0  66 00 38 00 61 00 31 00 32 00 31 00 38 00 36 00  f.8.a.1.2.1.8.6.
1584df38200  65 00 64 00 30 00 33 00 65 00 61 00 39 00 38 00  e.d.0.3.e.a.9.8.
1584df38210  62 00 32 00 61 00 34 00 63 00 62 00 31 00 36 00  b.2.a.4.c.b.1.6.
1584df38220  63 00 36 00 38 00 66 00 34 00 35 00 39 00 33 00  c.6.8.f.4.5.9.3.
1584df38230  39 00 33 00 65 00 31 00 66 00 63 00 27 00 22 00  9.3.e.1.f.c.'.".
1584df38240  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584df38250  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584df38260  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584df38270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Cellebrite

Tracing QSqlQuery.exec()

- Cleaning up even more
- Total string length is 84 (0x54) characters
- We can verify this string from other traces

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()');
        log(hexdump(args[1].readPointer(),
                    {length: 256}));
},
```

```
 7675 ms  ?exec@QSqlQuery@@QEAA_NAEBVQString@@@Z()
             0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
1584aa7cac0  01 00 00 00 15 00 00 00 16 00 00 00 00 00 00 00  ................
1584aa7cad0  18 00 00 00 00 00 00 00 50 00 52 00 41 00 47 00  ........P.R.A.G.
1584aa7cae0  4d 00 41 00 20 00 63 00 69 00 70 00 68 00 65 00  M.A. .c.i.p.h.e.
1584aa7caf0  72 00 5f 00 76 00 65 00 72 00 73 00 69 00 6f 00  r._.v.e.r.s.i.o.
1584aa7cb00  6e 00 00 00 50 01 00 00 c3 8d 2c 6d 00 ec 02 88  n...P.....,m....
1584aa7cb10  06 00 00 00 06 00 00 00 01 00 00 00 00 00 00 00  ................
1584aa7cb20  90 6e db 30 fa 7f 00 00 00 00 00 00 00 00 00 00  .n.0............
1584aa7cb30  00 00 00 00 00 00 00 00 20 2d 56 4e 58 01 00 00  ........ -VNX...
1584aa7cb40  01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584aa7cb50  c0 d5 f5 4d 58 01 00 00 c6 8d 33 6d 00 ed 02 80  ...MX.....3m....
1584aa7cb60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584aa7cb70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584aa7cb80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584aa7cb90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1584aa7cba0  00 00 00 00 00 00 00 00 c9 8d 36 6d 00 ee 02 88  ..........6m....
1584aa7cbb0  00 00 00 00 06 00 00 00 80 91 ed 2b fa 7f 00 00  ...........+....
```

Cellebrite

Tracing QSqlQuery.exec()

REALLY cleaning up

Read pointer to pointer

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log

onEnter: function (log, args, state) {
        log('Executing SQL:');

        qstring_addr = args[1].readPointer();



},
```

Cellebrite

Tracing QSqlQuery.exec()

REALLY cleaning up

Read pointer to pointer

Read length

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('Executing SQL:');

        qstring_addr = args[1].readPointer();
        qstring_len = qstring_addr.add(4).readInt();




},
```

Cellebrite

Tracing QSqlQuery.exec()

REALLY cleaning up

Read pointer to pointer

Read length

Validate length

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('Executing SQL:');

        qstring_addr = args[1].readPointer();
        qstring_len = qstring_addr.add(4).readInt();

        if (qstring_len == 0) {
                return;
        }



},
```

Cellebrite

Tracing QSqlQuery.exec()

- REALLY cleaning up
- Read pointer to pointer
- Read length
- Validate length
- Read and log value

```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

```
onEnter: function (log, args, state) {
        log('Executing SQL:');

        qstring_addr = args[1].readPointer();
        qstring_len = qstring_addr.add(4).readInt();

        if (qstring_len == 0) {
                return;
        }

        qstring_value = string_addr.add(0x18)
                                .readUtf16String(qstring_len);
        log(qstring_value);
},
```

Cellebrite

Tracing QSqlQuery.exec()

REALLY cleaning up

Read pointer to pointer

Read length

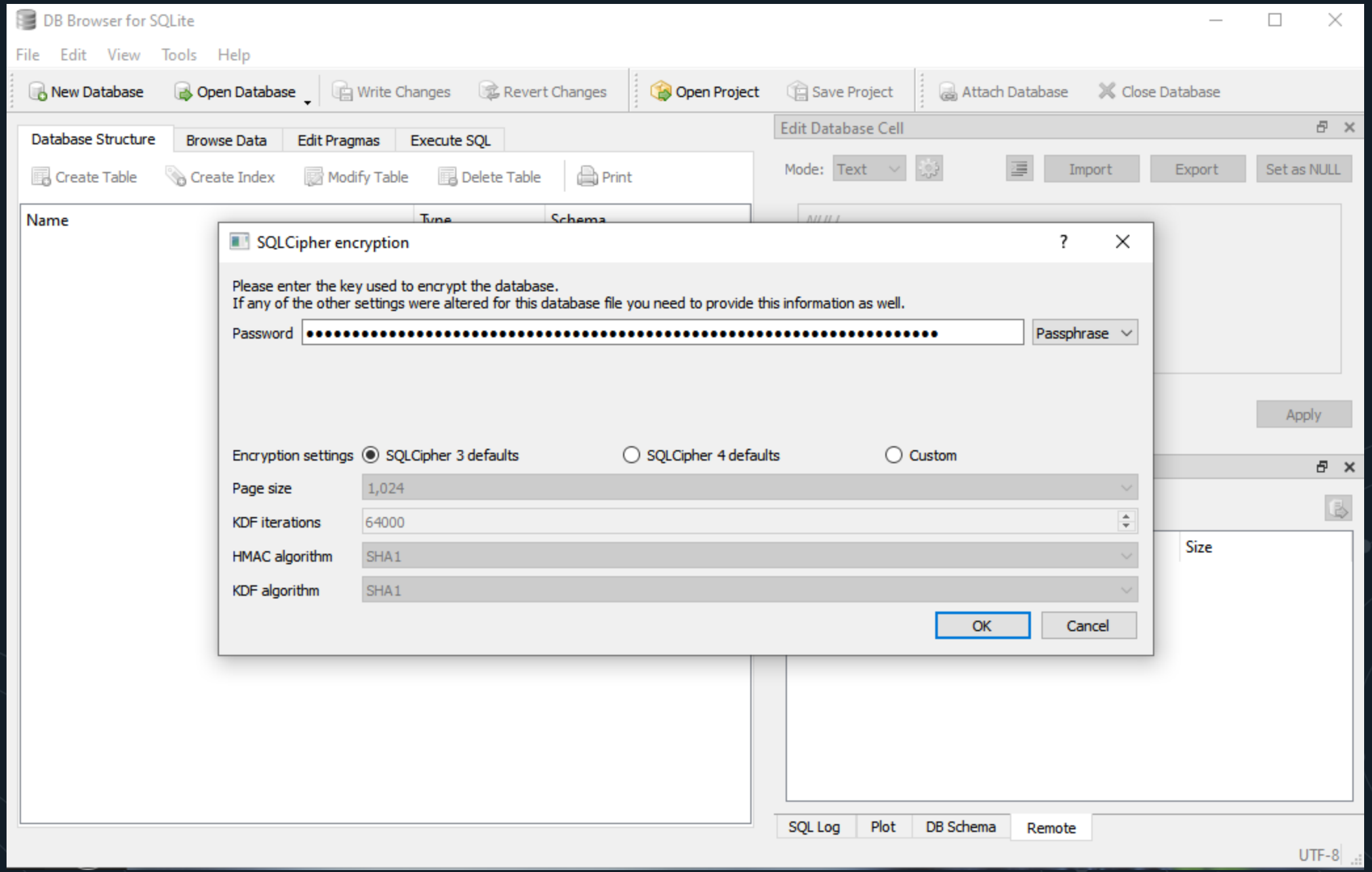
Validate length

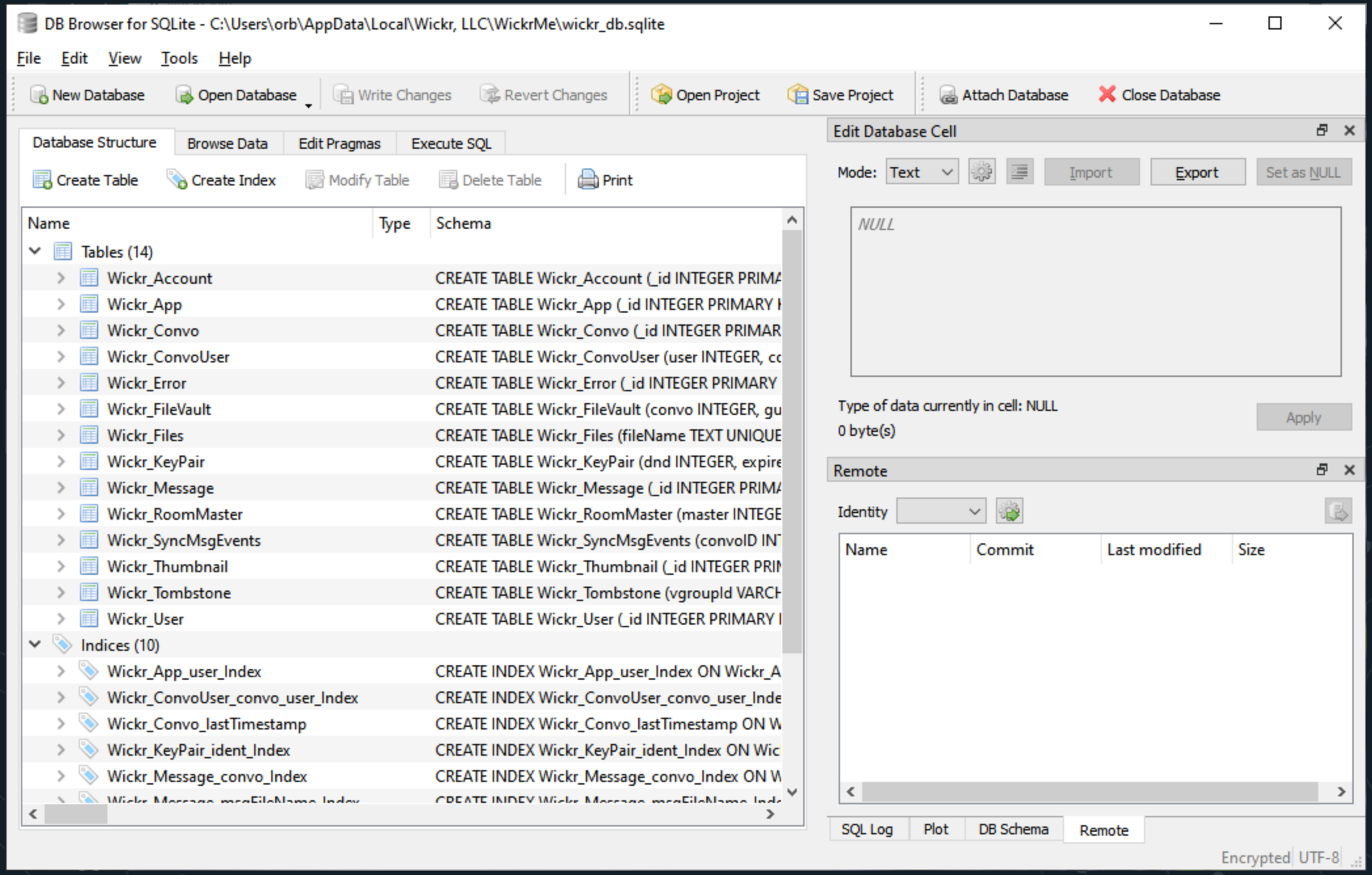
Read and log value

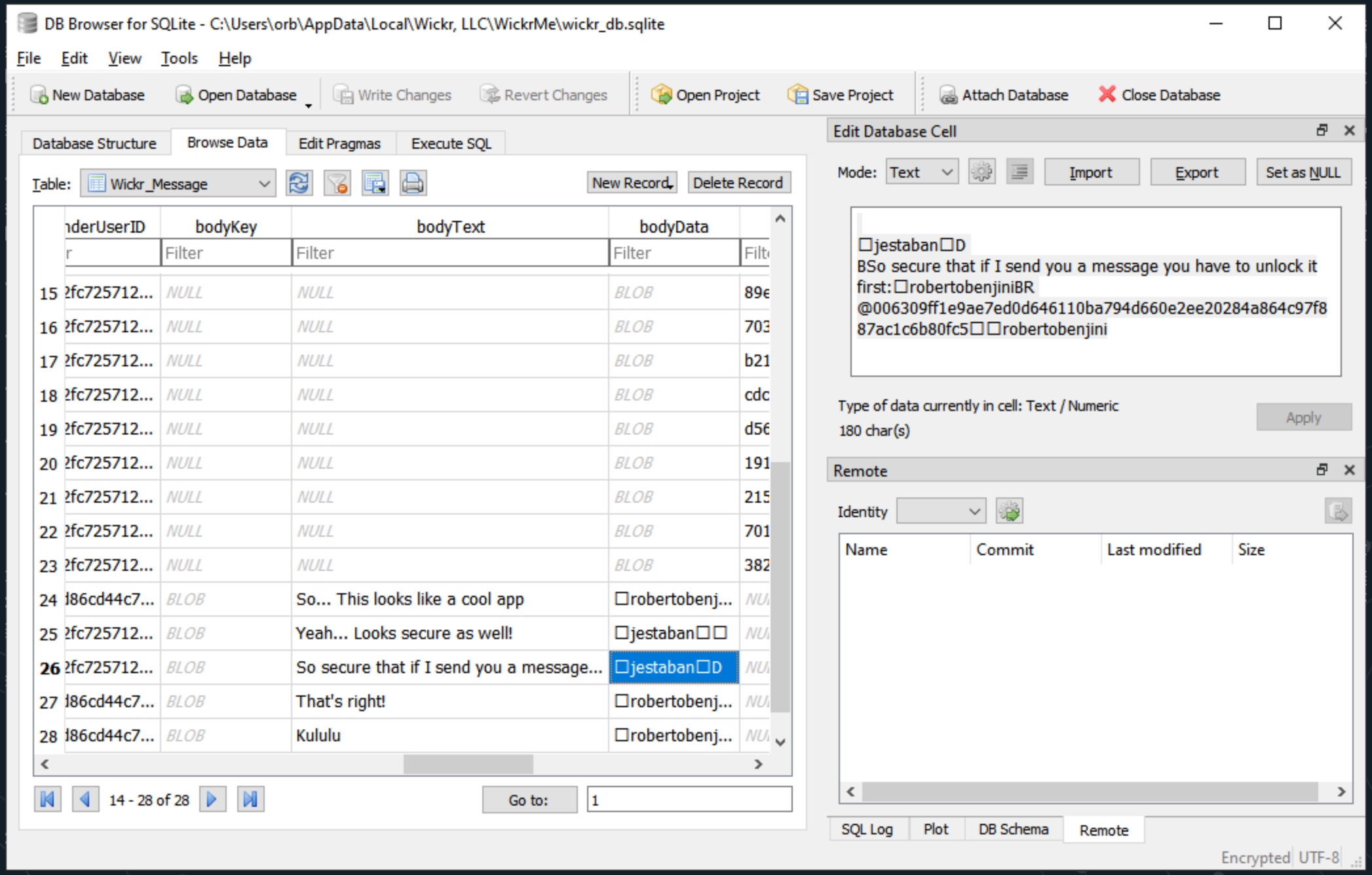
```
c:\Users\orb
λ frida-trace WickrMe.exe -i "*exec*@QSqlQuery*" > frida_exec.log
```

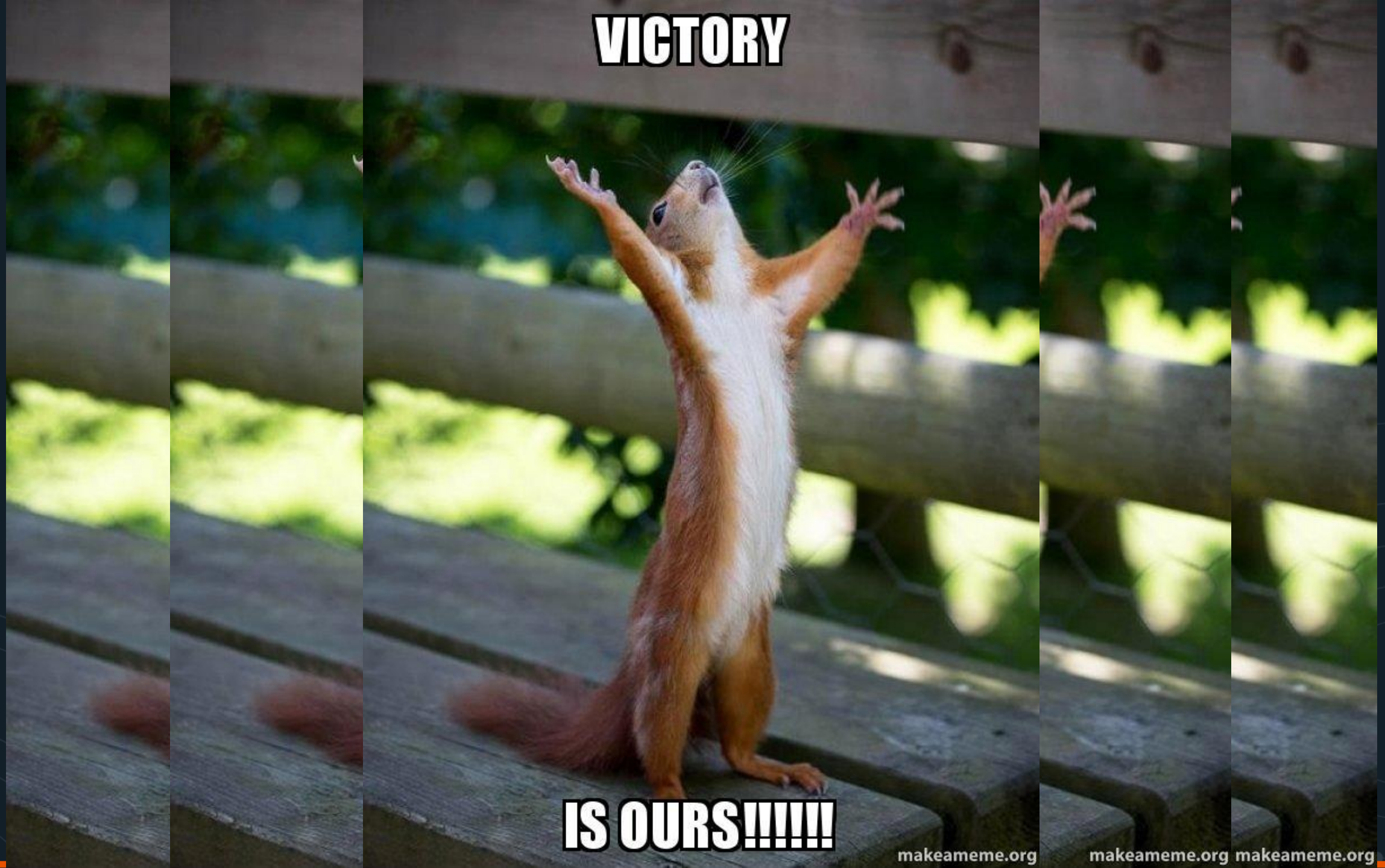
```
onEnter: function (log, args, state) {
        log('Executing SQL:');

        qstring_addr = args[1].readPointer();
        qstring_len = qstring_addr.add(4).readInt();

        if (qstring_len == 0) {
                return;
        }

        qstring_value = string_addr.add(0x18)
                                .readUtf16String(qstring_len);
        log(qstring_value);
},
```

```
6017 ms  Executing SQL:
6017 ms  PRAGMA key =
"x'00eb62e63896937b7301b0da1facf8a12186ed03ea98b2a4cb16c68f459393e1fc'"
6017 ms  Executing SQL:
6017 ms  PRAGMA cipher_version
6017 ms  Executing SQL:
6017 ms  PRAGMA cipher_compatibility = 3;
6017 ms  Executing SQL:
6017 ms  PRAGMA journal_mode = WAL
```


DB Browser for SQLite
File Edit View Tools Help
New Database
Open Database
Write Changes
Revert Changes
Open Project
Save Project
Attach Database
Close Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Create Table
Create Index
Modify Table
Delete Table
Print
Name
Edit Database Cell
Mode: Text
Import
Export
Set as NULL
Apply
SQLCipher encryption
Please enter the key used to encrypt the database.
If any of the other settings were altered for this database file you need to provide this information as well.
Password
Passphrase
Encryption settings
SQLCipher 3 defaults
SQLCipher 4 defaults
Custom
Page size
1,024
KDF iterations
64000
HMAC algorithm
SHA1
KDF algorithm
SHA1
OK
Cancel
Size
SQL Log
Plot
DB Schema
Remote
UTF-8


DB Browser for SQLite - C:\Users\orb\AppData\Local\Wickr, LLC\WickrMe\wickr_db.sqlite
File Edit View Tools Help
New Database
Open Database
Write Changes
Revert Changes
Open Project
Save Project
Attach Database
Close Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Create Table
Create Index
Modify Table
Delete Table
Print
Name
Type
Schema
Tables (14)
Wickr_Account
Wickr_App
Wickr_Convo
Wickr_ConvoUser
Wickr_Error
Wickr_FileVault
Wickr_Files
Wickr_KeyPair
Wickr_Message
Wickr_RoomMaster
Wickr_SyncMsgEvents
Wickr_Thumbnail
Wickr_Tombstone
Wickr_User
Indices (10)
Wickr_App_user_Index
Wickr_ConvoUser_convo_user_Index
Wickr_Convo_lastTimestamp
Wickr_KeyPair_ident_Index
Wickr_Message_convo_Index
Edit Database Cell
Mode: Text
Import
Export
Set as NULL
NULL
Type of data currently in cell: NULL
0 byte(s)
Apply
Remote
Identity
Name
Commit
Last modified
Size
SQL Log
Plot
DB Schema
Remote
Encrypted
UTF-8


DB Browser for SQLite - C:\Users\orb\AppData\Local\Wickr, LLC\WickrMe\wickr_db.sqlite
File Edit View Tools Help
New Database
Open Database
Write Changes
Revert Changes
Open Project
Save Project
Attach Database
Close Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table: Wickr_Message
New Record
Delete Record
nderUserID
bodyKey
bodyText
bodyData
Filter
15 2fc725712... NULL NULL BLOB 89e
16 2fc725712... NULL NULL BLOB 703
17 2fc725712... NULL NULL BLOB b21
18 2fc725712... NULL NULL BLOB cdc
19 2fc725712... NULL NULL BLOB d56
20 2fc725712... NULL NULL BLOB 191
21 2fc725712... NULL NULL BLOB 215
22 2fc725712... NULL NULL BLOB 701
23 2fc725712... NULL NULL BLOB 382
24 l86cd44c7... BLOB So... This looks like a cool app □robertobenj...
25 2fc725712... BLOB Yeah... Looks secure as well! □jestaban□□
26 2fc725712... BLOB So secure that if I send you a message... □jestaban□D
27 l86cd44c7... BLOB That's right! □robertobenj...
28 l86cd44c7... BLOB Kululu □robertobenj...
14 - 28 of 28
Go to: 1
Edit Database Cell
Mode: Text
Import
Export
Set as NULL
□jestaban□D
BSo secure that if I send you a message you have to unlock it first:□robertobenjiniBR
@006309ff1e9ae7ed0d646110ba794d660e2ee20284a864c97f887ac1c6b80fc5□□robertobenjini
Type of data currently in cell: Text / Numeric
180 char(s)
Apply
Remote
Identity
Name
Commit
Last modified
Size
SQL Log
Plot
DB Schema
Remote
Encrypted
UTF-8


VICTORY
IS OURS!!!!!!
makeameme.org
makeameme.org
makeameme.org

Resources

Official Frida documentation

frida.re/docs/

Script repositories

codeshare.frida.re/

github.com/dweinstein/awesome-Frida

github.com/iddoeldor/frida-snippets

google.com/search?q=frida+snippets


Cellebrite
Thank you
@Shloophen
Or.Begam@cellebrite.com