



# Mobile Reverse Engineering with r2frida

A beginners workshop...

[@enovella](#) && [@as0ler](#) && [@hexploitable](#)

Sep 2020

# Disclaimer/Expectations

1. This is a beginners workshop and today's walkthrough exercises are aimed at users new to r2frida, frida or r2.
2. You will learn how to install and set up r2frida, run basic commands, analyse an app at runtime and leverage the power of r2 and frida in unison.
3. Even if you have used r2/frida/r2frida before, stick around - you might learn something or obtain ideas for your own research/reversing.

# \$ whoami



**ÁLEX SOLER**

SECURITY RESEARCHER  
ATTACKIQ  
[@as0ler](#)



**EDU NOVELLA**

MOBILE SECURITY  
RESEARCHER  
NOWSECURE  
[@enovella\\_](#)



**GRANT DOUGLAS**

MOBILE SECURITY  
RESEARCHER  
NOWSECURE  
[@hexploitable](#)

# Agenda

1. Introduction to r2frida and some basic commands (~15 min)
2. **iOS walkthrough 1**: Intro to iOS commands and anti-jailbreak (1h)
3. 15 minute break
4. **iOS walkthrough 2**: Reversing r2con Crackme (1h)
5. 15 minute break
6. **Android walkthrough**: Solving CyberTruck Challenge crackme (1h)
7. Questions? (~15 min)



# Introduction to r2frida



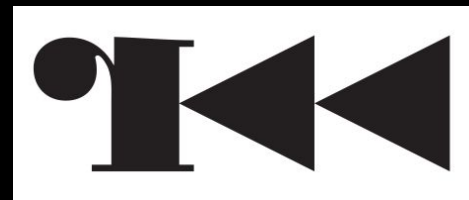
# What is radare2?



- Advanced free/open/libre hexadecimal editor with disassembler, debugger, ...
- Multi-platform, multi-architecture, works on any POSIX system and Windows
- Provides libraries, apis, bindings and scripting to use all the features
- Command-line interface (with visual and embedded web server interfaces)
- Each module can be extended with plugins
- r2pipe is the recommended way to script r2 from ANY language
- Easy to integrate with existing tools

## Bonus

- Cutter is the official graphical user interface
- r2pm is the package manager
- Great community and r2con conference



# What is Frida?



- A world-class dynamic instrumentation toolkit
  - Debug live processes
- Scriptable
  - Execute your own debug scripts inside another process
- Multi-platform
  - Windows, macOS, Linux, iOS, Android, QNX
- Highly modular, interacted with via JavaScript API
- Open Source

# FRIDA

# Why not leverage the power of both?



## R2Frida

- Radare2 IO plugin to use Frida from r2 shell
- Combines the power of static (Radare2) & dynamic analysis (Frida)
- Attach to (or spawn) any local or remote process (USB, TCP)
- Ability to read and write memory from target process
- Frida information loaded into r2 flags
  - maps, symbols, imports, class, methods, fields, ...
- Flexibility to build your own r2frida plugins
- Written in C/JavaScript (Open Source)



## Bonus

- Easy install: `r2pm -i r2frida`



# R2frida design



- Radare2 tool on top of the rest
- r2frida IO plugin
- Frida provides JS APIs to interact with target
- Target process instrumented by Frida and JS

```
$ r2 frida://attach/usb//re.target.app  
[0x00000000]> \?
```

# References

## Radare2

- 15yo project by pancake
- <https://rada.re>
- <https://twitter.com/radareorg>
- <https://github.com/radare/radare2>
- <https://t.me/radare>

## Frida

- 12yo project by Ole Andre
- <https://frida.re>
- <https://twitter.com/fridadotre>
- <https://github.com/frida/frida>
- <https://t.me/fridadotre>

## r2frida

- 5yo **NowSecure** project by pancake, mrmacete, ole
- <https://github.com/nowsecure/r2frida>

# Walkthroughs - setup

## Pre-requisites

- Radare2
- Frida
- R2frida plugin

[https://github.com/hexploitable/r2con2020\\_r2frida](https://github.com/hexploitable/r2con2020_r2frida)

## Challenges

- OWASP CrackMe 2 (iOS)
- Anti-Jailbreak app (iOS)
- R2con 2020 crackme (iOS)
- CyberTruck 2019 crackme (Android)

## Devices (preferably rooted)

- iPhone
- Android



# r2frida cmds: install

- From the pkg manager:

```
$ r2pm -ci r2frida
```

- From the sources:

```
$ # install deps  
$ git clone https://github.com/nowsecure/r2frida.git  
$ cd r2frida  
$ make  
$ sudo make install
```

- Radare2 plugin listing:

```
$ r2 -L| grep frida  
rw_ frida    frida:// io plugin (MIT)
```

# r2frida cmds: starting up

- Listing devices:
  - `r2 frida://usb/`
- Listing processes:
  - `r2 frida://usb/<device>/`
  - `r2 frida://usb//`
- Spawning a process:
  - `r2 frida://spawn/usb/<device>/<app id>`
  - `r2 frida://spawn/usb//<app id>`
- Attaching to a process:
  - `r2 frida://attach/usb/<device>/<process id>`
  - `r2 frida://attach/usb/<device>/<process name>`
  - `r2 frida://attach/usb//<process id>`
- Apps are suspended at startup:
  - Resume the process with `\dc`
  - iOS apps are killed by OS if you don't resume within 20s
    - **Hacky** workaround:  
<https://tinyurl.com/spawndelay>
    - try not to use this unless necessary

- Via the network (default port === 27042)
  - `r2 frida://connect/ip:port/<process id>`
  - `r2 frida://connect/ip:port/<process name>`
  - `r2 frida://spawn/connect/ip:port/<app id>`

# r2frida cmds: engine selection

- Choosing the JavaScript engine:
  - Choices:
    - V8 (default in **r2frida**)
    - Duktape (default in **frida**)
      - Required for using Stalker
      - Required for using Gadget

```
> r2 frida://spawn/usb//r2con.2020.CrackMe
-- I accidentally the kernel with radare2.
[0x00000000]> \eval Script.runtime
V8
```

```
> R2FRIDA_DISABLE_V8=1 r2 frida://spawn/usb//r2con.2020.CrackMe
-- This is an unregistered copy.
[0x00000000]> \eval Script.runtime
DUK
```

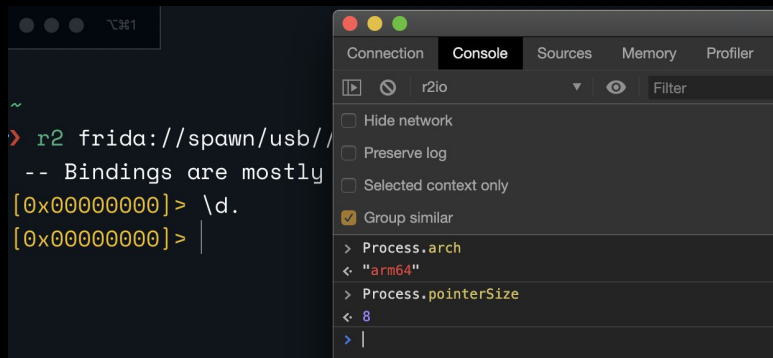
- **R2FRIDA\_DISABLE\_V8=1** r2 frida://\*
- **R2FRIDA\_DISABLE\_V8=0** r2 frida://usb//Gadget

# r2frida cmds: common

- Asking for help:
  - `\?`
- Grep the output of commands
  - `\?~+keyword`
- Check Frida Version:
  - `\?V`
- Connecting to chrome dev tools:
  - `\d.`
  - Start devtools in Chrome and select the green NodeJS icon.

```
[0x00000000]> \?V  
{"version": "12.11.12"}
```

```
[0x00000000]> \d.
```



# r2frida cmds: common

Process info:

- \i

```
[0x00000000]> \i
arch           arm
bits           64
os             darwin
pid            454
uid            501
objc           true
runtime        V8
java           false
cylang         true
pageSize       16384
pointerSize    8
codeSigningPolicy
optional
isDebuggerAttached false
cwd            /
```

Running Frida code:

- \eval
- \<space>

```
[0x00000000]> \eval
console.log('hello: ' +
Process.arch)

hello: arm64
```

Restart Session:

- oo

```
[0x00000000]> oo
error: The connection is closed
[0x00000000]> DetachReason:
FRIDA_SESSION_DETACH_REASON_PROCE
SS_TERMINATED
[0x00000000]> \i~arch
arch           arm
```



# r2frida cmds: information (\?~^i)

Library info:

- \il
- \il\*
- \il\*

```
[0x00000000]> \il
0x000000001046f0000 CrackMe
0x00000000104730000
pspawn_payload-stg2.dylib
0x000000001bcae4000 Foundation
0x000000001bb365000 libobjc.A.dylib
0x000000001bb2fa000 libSystem.B.dylib
0x000000001bc079000 CoreFoundation
...
```

Export info:

- \iE
- \iE name
- \iE\* name

```
[0x1046f0000]> \iE
0x1046f0000 v _mh_execute_header
0x1046fba44 f main
0x1047122f0 v
OBJC_CLASS_$_SecondViewController
0x104712368 v OBJC_CLASS_$_AppDelegate
OBJC_CLASS_$_FirstViewController
...
```

Import info:

- \ii
- \ii name
- \ii\* name
- \ii\* name

```
[0x1046f0000]> \ii
0x1f533de60 v OBJC_METACLASS_$_NSObject
/usr/lib/libobjc.A.dylib
0x1f533de60 v OBJC_METACLASS_$_NSObject
/usr/lib/libobjc.A.dylib
0x1f533de60 v OBJC_METACLASS_$_NSObject
/usr/lib/libobjc.A.dylib
...
```

Symbol info:

- \is
- \is name
- \is\* name
- \is\* name

```
[0x1046f0000]> \is
0x1046f7e38 s -[SecondViewController
viewDidLoad]
0x1046fb190 s __35-[SecondViewController
viewDidLoad]_block_invoke
0x1046fb1c4 s __35-[SecondViewController
viewDidLoad]_block_invoke_2
...
```

# r2frida cmds: information (\?~^i)

- Class info:

- \ic

```
[0x00000000]> \ic
NSLeafProxy
Object
__NSGenericDeallocHandler
__NSZombie_
__NSMessageBuilder
JSEExport
NSProxy
...
```

```
[0x00000000]> \ic~+activity
com.android.internal.telephony.PhoneInternalI
nterface$DataActivityState
android.app.ActivityThread$ServiceArgsData
android.content.pm.ActivityInfo$WindowLayout
android.app.ActivityManager$TaskDescription$1
...
```

- Method info:

- \ic **Classname**

```
[0x00000000]> \ic FirstViewController
0x000000001046fbb14 + load
0x0000000010470e05c - challenge1TabItem
0x000000001046fbb00 - bonus_flag_agf
0x0000000010470e090 - setChallenge1TabItem:
0x0000000010470e0e0 - .cxx_destruct
0x00000000104704d54 - viewDidLoad
...
```

# r2frida cmds: tracing code (\?~dt)

- Functions:

- \dtf

```
[0x00000000]> \dtf?
Usage: dtf [format] || dtf [addr] [fmt]
^ = trace onEnter instead of onExit
+ = show backtrace on trace
p/x = show pointer in hexadecimal
c = show value as a string (char)
i = show decimal argument
z = show pointer to string
s = show string in place
O = show pointer to ObjC object
Undocumented: Z, S
dtf trace format
[0x00000000]> \dtf fopen z^;
true

[0x00000000]> [TRACE] dtf fopen (0:
"/proc/self/cmdline") 0x7f0f24e04c
libcutils.so 0xc04c
...
```

- Registers:

- \dtr

```
[0x00000000]> \dtr?
dtr trace regs
[0x00000000]> \dtr `isa fopen` x0

[0x00000000]> [TRACE] dtr 0x7f0f6bdcc0
(x0: "/proc/self/cmdline") 0x7f0f24e04c
libcutils.so 0xc04c
0x7f0f24de78 libcutils.so
atrace_set_debuggable+0x6c
0x72745c18 boot-framework.oat
0x3e3c18
```

# r2frida cmds: debugging (\?~^db)

- Set breakpoints
  - `\db <address>`
- Unset breakpoints:
  - `\db- <address>`
- List breakpoints:
  - `\db`
- List threads
  - `\dpt`
- Dump registers:
  - `\dr`
- Clear all breakpoints:
  - `\db-*`

```
[0x00000000]> \db `isa fopen`
```

```
[0x00000000]> \drj~{[1].context}  
{ "lr": "0x1bbca456c", "fp": "0x16e215730", "x28": "0x102351ea0", "x27": "0x102351ea8", "x26": "0x1035dbcd8", "x25": "0x10273a274", "x24": "0x1035e9ab3", "x23": "0x102351e18", "x22": "0x0", "x21": "0x1035d9d50", "x20": "0x16e215750", "x19": "0x16e215740", "x18": "0x0", "x17": "0x0", "x16": "0x14e", "x15": "0x0", "x14": "0x0", "x13": "0x4", "x12": "0x3", "x11": "0x0", "x10": "0x11", "x9": "0x23", "x8": "0x1f53439e0", "x7": "0x1", "x6": "0x0", "x5": "0x0", "x4": "0x1", "x3": "0x1", "x2": "0x1", "x1": "0x0", "x0": "0x1403", "sp": "0x16e215700", "pc": "0x1bbd2e400" }
```

```
[0x00000000]> \drj~{[1].context[pc]}  
0x1bbd2e400
```

# r2frida cmds: memory (\?~^dm)

- Show memory ranges

- \dm
- \dm.

```
[0x00000000]> \dm~nodebug~r-x
0x0000000010215c000 - 0x00000000102164000 r-x
/private/var/containers/Bundle/Application/
6902ADEE-1A7F-455E-B521-48E359A382C2/nodebu
g.app/nodebug
```

- Show memory maps

- \dmm
- \dmm.

```
[0x00000000]> \dmm
0x000000001020c4000 - 0x0000000010212c000 rwx
/usr/lib/pspawn_payload-stg2.dylib
0x00000000102478000 - 0x00000000102538000 rwx
/usr/sbin/frida-server
0x00000000103bcc000 - 0x00000000103bf8000
...
```

- List mallocs

- Ranges: \dmh
- Maps: \dmhm

- Using the heap

- Allocate memory: \dma <size>
- Allocate String: \dmas <string>
- Clone bytes: \dmad <ptr> <size>
- Deallocate: \dma- <addr>

# r2frida cmds: search (\?~^/)

```
$ r2 frida://0
-- There is no F5 key in radare2 yet
[0x00000000]> \/?
/      search
/j      search json
/v1
/v1j
/v2
/v2j
/v4
/v4j
/v8
/v8j
/w      search wide
/wj     search wide json
/x      search hex
/xj     search hex json
```

```
$ r2 frida://0
-- Good morning, pal *<:-)
[0x00000000]> s 0x7f094d4d0000
[0x7f094d4d0000]> .\e/
e search.in=raw
e search.from=0x7f094d4d0000
e search.to=0x7f094d4d7000
e anal.in=raw
e anal.from=0x7f094d4d0000
e anal.to=0x7f094d4d7000
```

```
$ r2 frida://0
-- See you in shell
[0x00000000]> \e
e java.wait=false
e patch.code=true
e search.in=perm:r--
e search.quiet=false
e stalker.event=compile
e stalker.timeout=300
e stalker.in=raw
e hook.backtrace=true
e hook.verbose=true
e symbols.unredact=false
```

# r2frida cmds: mount

## Mount remote filesystem

```
[0x00000000]> m / io 0
Mounted io on / at 0x0
[0x00000000]> md /system
d media
f build.prop
d bin
d usr
d framework
d fake-libs
d priv-app
d product
d lib64
f compatibility_matrix.xml
l vendor
d etc
[0x00000000]> mc /data/local/tmp/remotefile
You are reading this file from the Android
device :)
```

```
[0x00000000]> m?
m /mnt ext2 0      Mount ext2 fs at /mnt with delta 0 on IO
m /mnt             Mount fs at /mnt with autodetect fs and
                   current offset

m                  List all mountpoints in human readable
format

m*                Same as above, but in r2 commands
m-/              Umount given path (/)
mL               List filesystem plugins (Same as Lm)
mc [file]         Cat: Show the contents of the given file
md /             List directory contents for path
mf[?] [o|n]       Search files for given filename or for
offset

mg /foo           Get fs file/dir and dump it to disk
mi /foo/bar       Get offset and size of given file
mj               List mounted filesystems in JSON
mo /foo/bar       Open given file into a malloc://
mp msdos 0        Show partitions in msdos format at offset 0
mp               List all supported partition types
ms /mnt           Open filesystem prompt at /mnt
mw [file] [data]  Write data into file
my               Yank contents of file into clipboard
```

# r2frida cmds: others

- Auto set for r2 evaluable vars and r2frida commands

```
[0x00000000]> \init
```

```
e dbg.backend =io  
e anal.autoname=true  
e cmd.fcn.new=aan  
.=!ie*  
.=!il*  
m /r2f io 0  
s entry0
```

```
[0x00000000]> .\init
```

```
Mounted io on /r2f at 0x0
```

```
[0x104acd790]>
```





# iOS Walkthrough 1

# Agenda iOS Walkthrough

1. Intro to objc using r2frida commands
2. iOS Challenge
  - a. Bypass debug checks.
  - b. Bypass Jailbreak detection.



\db \$\$ - breaktime



# iOS Walkthrough 2



\db \$\$ - breaktime



# Android Walkthrough


# Agenda Android Walkthrough

## 1. Intro to Android CrackMe:

- a. CyberTruck Challenge 2019
  - i. Solving Challenge 1
  - ii. Solving Challenge 2
  - iii. Solving Challenge 3
  - iv. Bypassing TamperProof

# Credits

This is where we give credit to the ones who help to this workshop

- @pancake for the Radare2 project and all his contributors.
- @oleavr for the Frida project and all his contributors.
- @mrmacete for r2frida support and all his contributors.
-  NowSecure™ for open-sourcing & supporting the tools.





# References

- Official r2frida source: <https://github.com/nowsecure/r2frida>
- Official r2frida Gitbook (WIP): <https://github.com/nowsecure/r2frida-book>
- Unofficial r2frida-wiki: <https://github.com/enovella/r2frida-wiki>
- Pass the Salt 2018: R2frida Better Together: <https://2018.pass-the-salt.org/files/talks/02-r2frida.pdf>
- r2con 2016: the ultimate static analysis on dynamic steroids:  
<https://github.com/radareorg/r2con2016/blob/03f27eba5410e9edaa517af3be8c7a4cc06cedb1/talks/08-r2frida/r2frida.pdf>
- r2con 2017: r2frida:  
<https://github.com/radareorg/r2con2017/blob/master/talks/r2frida/r2frida-r2con-2017.pdf>
- Reversing a Hardened Android Video-Game:  
[https://slides.com/eduardonovella/nowsecure\\_connect19\\_enovella\\_noname\\_packer#/](https://slides.com/eduardonovella/nowsecure_connect19_enovella_noname_packer#/)

# Videos

- r2con 2016: the ultimate static analysis on dynamic steroids - <https://www.youtube.com/watch?v=ivCucgeVeZI>
- r2con 2017: r2frida - <https://www.youtube.com/watch?v=URyd4bcV-lk>
- Pass the Salt 2018: R2frida Better Together - <https://passthesalt.ubicast.tv/videos/r2frida-better-together>
- NowSecure Webinar: OSS Tools: Creating a Reverse Engineering Plug-in for r2frida - <https://www.brighttalk.com/webcast/15139/353221/oss-tools-creating-a-reverse-engineering-plug-in-for-r2frida>

# Questions?



@as0ler



@enovella\_



@hexploitable

FYI -- we'll be offering a 2-days advanced workshop in the near future.  
Stay tuned!