r2frida

Francesco Tamagni

- mrmacete / @bezjaje
- reverse / forward engineer
- minor contributor of r2 and Frida
- working @ NowSecure research team
- focus on mac/iOS
- i sleep too much to be a hacker

r2frida is a way to integrate r2 and Frida

what's Frida

- open source http://frida.re
- created by @oleavr, it has a nice growing community
- injects a thread in the target application
- from there it spawns a **js engine** in the app
- and an "**agent**" to communicate back and forth with your tool
- the target app may be on a remote device or on the host

what's Frida (2)

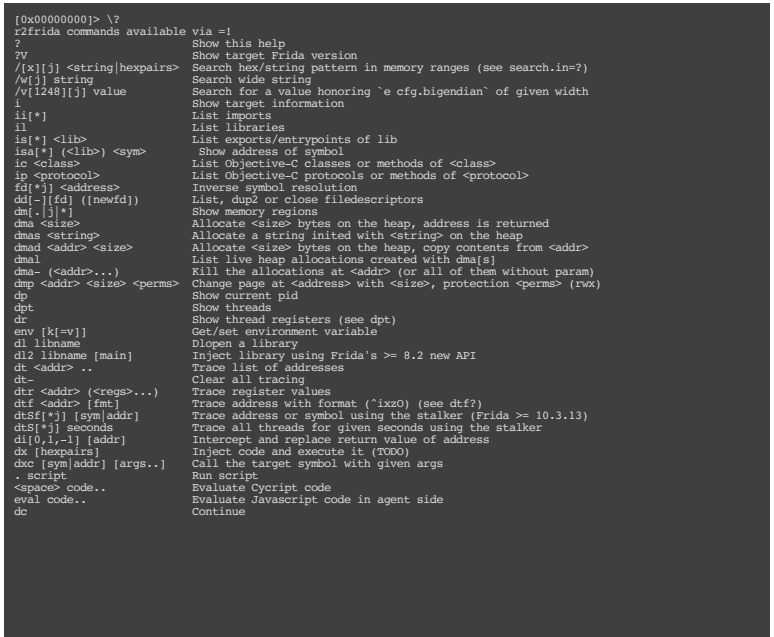
- through the agent you can:
  - read / write process memory
  - access imports, exports and debug symbols
  - call / implement native functions
  - Interceptor: place probes (at function or instruction level)
  - manipulate ObjC / Java runtime
  - Stalker: dynamic recompile with transformation
  - all of the above scriptable in JS
- you can **build tools on top** of Frida
- bindings in many languages

what's Frida (3)

- works on all major platforms
  - Windows
  - Linux
  - mac/iOS
  - Android
  - QNX
- on these architectures
  - arm / arm64
  - ia32 / x64
  - mips

r2frida

- **io plugin** based on frida
- source at https://github.com/nowsecure/r2frida
- "opens" the memory of a process on host or mobile devices (can spawn it too)
- **"debuggerless"** debugging
- has commands which integrate Frida features in r2



install

- using r2pm
  - r2pm -i r2frida
- from git
  - git clone https://github.com/nowsecure/r2frida.git
  - cd r2frida
  - make && make install
- check installation

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

start it up

- **attach to a running process**
  - on the host
    - r2 frida://Twitter
    - r2 frida://<pid>
  - on a device
    - r2 frida://<device_id>/Twitter
    - r2 frida://<device_id/<pid>
- **spawn a process**
  - on the host
    - r2 frida:///usr/bin/ls
    - r2 "frida:///usr/bin/ls -al"
  - on a device
    - r2 frida://<device_id>//your.package.name

spawn a process

- spawn in suspended state
- must be resumed with **\dc**

```
$ r2 "frida:///bin/ls /"
[0x00000000]> \dtf malloc i
true
[0x00000000]> \dc
[0x00000000]> malloc 3312 = 0x7fadd4020800
malloc 4096 = 0x7fadd400d800
malloc 231 = 0x7fadd3c19070
malloc 32768 = 0x7fadd403dc00
malloc 16384 = 0x7fadd600a200
[0x00000000]>
```

inspection commands

...but let's do a demo first

i[*]

- show information about the process
- **.\i*** sets the correct arch, bits and endianess to get a proper disassembly in r2

```
[0x18eab4ffc]> \i~arch
arch    arm
[0x18eab4ffc]> pd 5
        └─< 0x18eab4ffc      e923bb6df8     jmp 0x187190b24
            0x18eab5001      5f             pop rdi
            0x18eab5002      01a9f65702a9   add dword [rcx - 0x56fda80a], ebp
            0x18eab5008      f4             hlt
            0x18eab5009      4f03a9fd7b04.  add r13, qword [r9 - 0x56fb8403]
[0x18eab4ffc]> .\i*
[0x18eab4ffc]> pd 5
            0x18eab4ffc      e923bb6d       stp d9, d8, [sp, -0x50]!
            0x18eab5000      f85f01a9       stp x24, x23, [sp, 0x10]
            0x18eab5004      f65702a9       stp x22, x21, [sp, 0x20]
            0x18eab5008      f44f03a9       stp x20, x19, [sp, 0x30]
            0x18eab500c      fd7b04a9       stp x29, x30, [sp, 0x40]
```

il

- list libs, in order of linking
- get module names to use in other commands
- **il.** where am i?

```
[0x00000000]> \il
0x0000000107f70000 Twitter
0x00007fffa5aeb000 libxml2.2.dylib
0x00007fffa5c05000 libz.1.dylib
0x00007fffa4e52000 libicucore.A.dylib
0x00000001084b4000 Growl
0x00007fff8ff64000 Cocoa
0x00007fff9609f000 QuartzCore
0x00007fffa589a000 libsqlite3.dylib
0x00007fff96806000 Security
0x00007fff8faac000 Carbon
0x00007fff988c7000 WebKit
0x00007fff9125b000 CoreLocation
0x00007fff91f72000 Foundation
0x00007fffa5406000 libobjc.A.dylib
0x00007fffa4757000 libSystem.B.dylib

[...]
```

ii[*]

- without arguments show all imports of all modules
- with one argument you can specify the module
- with two arguments you'll get the specific symbol imported by the given module, if present
- **.\ii*** sets flags

```
[0x18eab4ffc]> \ii libsystem_c.dylib
0x1990f4b74 f dyld_stub_binder /usr/lib/system/libdyld.dylib
0x19fb820e8 v _os_alloc_once_table /usr/lib/system/libsystem_kernel.dylib
0x19fb82098 v bootstrap_port /usr/lib/system/libsystem_kernel.dylib
0x19fb8209c v mach_task_self_ /usr/lib/system/libsystem_kernel.dylib
0x199240178 f free /usr/lib/system/libsystem_malloc.dylib
0x1992d5908 f _sigtramp /usr/lib/system/libsystem_platform.dylib
0x19fb800fc v __unix_conforming /usr/lib/system/libsystem_pthread.dylib
0x19fb85618 v _NSConcreteGlobalBlock /usr/lib/system/libsystem_blocks.dylib
0x19fb85218 v _NSConcreteStackBlock /usr/lib/system/libsystem_blocks.dylib
0x1990f557c f dladdr /usr/lib/system/libdyld.dylib
0x1990f5664 f dlopen /usr/lib/system/libdyld.dylib
0x1990f572c f dlsym /usr/lib/system/libdyld.dylib
[...]

[0x18eab4ffc]> \ii* libsystem_c.dylib dlopen
f sym.imp.dlopen = 0x1990f5664
```

is[*j] <lib>

- use **.\is*** to import all exported symbols of a library as flags

```
[0x7fffa5e2fa34]> pd 3
            0x7fffa5e2fa34      b805000002      mov eax, 0x2000005
            0x7fffa5e2fa39      4989ca          mov r10, rcx
            0x7fffa5e2fa3c      0f05            syscall
[0x7fffa5e2fa34]> .\is* libsystem_kernel.dylib
[0x7fffa5e2fa34]> pd 3
            ;-- sym.fun.__open:
            ;-- sym.fun.open:
            0x7fffa5e2fa34      b805000002      mov eax, 0x2000005
            0x7fffa5e2fa39      4989ca          mov r10, rcx
            0x7fffa5e2fa3c      0f05            syscall
```

isa[*j] (<lib>) <sym>

- show the address of an exported symbol
- if exported multiple times with different addresses, all of them are shown

```
[0x7fffa5e2fa34]> \isa open
0x7fffa5e2fa34
0x110120c20

[0x7fffa5e2fa34]> \isaj open~{}
[
  {
    "library": "Twitter",
    "name": "open",
    "address": "0x7fffa5e2fa34"
  },
  {
    "library": "dyld",
    "name": "open",
    "address": "0x110120c20"
  }
]

[0x7fffa5e2fa34]> \isaj libsystem_kernel.dylib open~{}
[
  {
    "library": "libsystem_kernel.dylib",
    "name": "open",
    "address": "0x7fffa5e2fa34"
  }
]
```

ic <class>

- inspect ObjC / Java classes
- if <class> is specified, it'll show all methods of that class
- without <class> it'll show all loaded class names

```
[0x7fffa5e2fa34]> \ic~AppDelegate
Tweetie2AppDelegate

[0x7fffa5e2fa34]> \ic Tweetie2AppDelegate~Did
0x0000000107f7572a - settingsScribeTimerDidFire:
0x0000000107f76300 - screenDidChange:
0x0000000107f77546 - colorPreferencesDidChange:
0x0000000107f75612 - userStreamDidClose:
0x0000000107f77540 - appearancePreferencesDidChange:
0x0000000107f775f2 - fontPreferencesDidChange:
0x0000000107f77680 - mentionsPreferencesDidChange:
0x0000000107f75f72 - applicationDidFinishLaunching:

[0x7fffa5e2fa34]> \ic* Tweetie2AppDelegate~DidFire
f sym.objc.Tweetie2AppDelegate.settingsScribeTimerDidFire = 0x0000000107f7572a
```

ip <protocol>

- list ObjC protocols, or methods of a protocol

```
[0x00000000]> \ip~Twit
TwitterAuthenticated
TwitterStreamErrorObject
TwitterScribableItem
TwitterAccountDelegate
TwitterFoundMediaProtocol
[0x00000000]> \ip TwitterAuthenticated
- account
```

fd[*j] <address>

- resolve a symbol given its address
- may be slow

```
[0x00000000]> \fd 0x7fffa5e2fa34
function __open 0x7fffa5e2fa34
function open 0x7fffa5e2fa34
function __open 0x7fffa5e2fa34
function open 0x7fffa5e2fa34
```

search

```
/[x][j] <string|hexpairs>  Search hex/string pattern
/w[j] string               Search wide string
/v[1248][j] value          Search for a value
```

- results are displayed in a familiar way
- obey r2's e~search configuration (flags, limits, ...)
- can be controlled by r2frida-specific eval options

```
[0x00000000]> \e~search
e search.in=perm:r--
e search.quiet=false
[0x00000000]> \e search.in=?
Specify which memory ranges to search in, possible values:

    perm:---        filter by permissions (default: 'perm:r--')
    current         search the range containing current offset
    path:pattern    search ranges mapping paths containing 'pattern'
```

debugging commands

dm[.|j]

- **dm** show memory maps
- **dm.** show the map containing the current offset

```
[0x00000000]> \dm~Twitter~rw
0x0000000108381000 - 0x0000000108481000 rw- /Applications/Twitter.app/Contents
/MacOS/Twitter
0x0000000108511000 - 0x0000000108523000 rw- /Applications/Twitter.app/Contents
/Frameworks/Growl.framework/Versions/A/Growl
```

dmp <addr> <size> <perms>

- change page permissions

```
[0x00000000]> \dm~Twitter~rw
0x000000010838f000 - 0x0000000108481000 rw- /Applications/Twitter.app/Contents
/MacOS/Twitter
0x0000000108511000 - 0x0000000108523000 rw- /Applications/Twitter.app/Contents
/Frameworks/Growl.framework/Versions/A/Growl

[0x00000000]> ?v 0x000000010d027000 - 0x000000010cf35000
0xf2000

[0x00000000]> \dmp 0x000000010cf35000 0xf2000 r-x
true

[0x00000000]> \dm~Twitter~0x000000010cf35000 -
0x000000010cf35000 - 0x000000010d027000 r-x /Applications/Twitter.app/Contents
/MacOS/Twitter

[0x00000000]>
```

hijack

- **di[0,1,-1] [addr]** replace return value of function
- **e patch.code** set to true to pseudosign writes

```
[0x00000000]> \ic LicenseManager~isRegistered
0x0000000107f7572a - isRegistered

[0x00000000]> \di1 0x0000000107f7572a
```

```
[0x00000000]> \e patch.code=true

[0x00000000]> wx 90909090 @ 0x1076ae6ce
```

use heap

- **dma <size>** allocate <size> bytes
- **dmas <string>** allocate a string
- **dmad <addr> <size>** duplicate <size> bytes from <addr>
- **dmal** list allocations created with dma[s]
- **dma- (<addr>...)** deallocate at <addr> (or all)

```
[0x00000000]> \dmas filename.txt
0x128ec2a20
[0x00000000]> \dma 8
0x128ebde40
[0x00000000]> \dmal
0x128ebde40     " C( ("
0x128ec2a20     "filename.txt S@ (@,("
[0x00000000]> \e patch.code=false
[0x00000000]> wv8 0x15eedeadbabebeef @ 0x128ebde40
[0x00000000]> pxq 8 @ 0x128ebde40
0x128ebde40  0x15eedeadbabebeef                         ........
[0x00000000]> \dma- 0x128ebde40 0x128ec2a20
```

threads & process

- **dp** show PID
- **dpt** show thread ids
- **dr** show thread registers

```
[0x00000000]> \dp
27116
[0x00000000]> \dptj
[775,41219,44551,55555,48667,48963,29495,
72755,7991]
[0x00000000]> \drj~{[1].id}
41219
[0x00000000]> \drj~{[1].context[rsp]}
0x114677dc8
[0x00000000]> pxq 64 @ `\drj~{[1].context[rsp]}`
0x114677dc8  0x00007fffa5e27797  0x0000000000000000   .w..............
0x114677dd8  0x0000000600000006  0x000000000000005c   ........\.......
0x114677de8  0x000000010fb16000  0x000000010d5bd99f   .`........[.....
0x114677df8  0x00000001146bf140  0x000000001d0008ff   @.k.............
```

env

- manipulate environment

```
[0x00000000]> \env
__CF_USER_TEXT_ENCODING=0x1F5:0x0:0x0
SHELL=/bin/bash
PATH=/usr/bin:/bin:/usr/sbin:/sbin
LOGNAME=ftamagni
XPC_SERVICE_NAME=com.twitter.twitter-mac.44188
USER=ftamagni
XPC_FLAGS=0x0
APP_SANDBOX_CONTAINER_ID=com.twitter.twitter-mac
[...]

[0x00000000]> \env APP_SANDBOX_CONTAINER_ID
APP_SANDBOX_CONTAINER_ID=com.twitter.twitter-mac

[0x00000000]> \env APP_SANDBOX_CONTAINER_ID=nope
APP_SANDBOX_CONTAINER_ID=nope

[0x00000000]> \env NEWVAR=true
NEWVAR=true
```

library injection

- **dl <libpath>** open a library using dlopen
- **dl2 <libpath> [<main>]** open a library using Frida

```
[0x00000000]> \dl /usr/lib/libr_core.dylib
0x105679b00

[0x00000000]> \dxc r_core_new
"0x10c160000"

[0x00000000]> \dmas ?E i am inside
0x10ac90010

[0x00000000]> \dxc r_core_cmd_str 0x10c160000 0x10ac90010
"0x170134dc0"

[0x00000000]> ps @ 0x170134dc0
 .--.     .-------------.
 | _|     |             |
 | O O   <  i am inside |
 |  |  |  |             |
 || | /   `-------------'
 |`-'|
 `---'
```

dxc [sym|addr] [args..]

- **call a function** by symbol name or address, passing the given arguments

```
[0x00000000]> \dmas /bin/ls
0x12b837110

[0x00000000]> \dma 4096
0x107c44000

[0x00000000]> \dxc stat 0x12b837110 0x107c44000
"0x0"

[0x00000000]> pf.stat iN4wN4N4 st_dev st_ino st_mode st_uid st_gid

[0x00000000]> pf.stat @ 0x107c44000
  st_dev : 0x107c44000 = 16777220
  st_ino : 0x107c44004 = 23969031
 st_mode : 0x107c44008 = 0x81ed
  st_uid : 0x107c4400a = 1
  st_gid : 0x107c4400e = 0
```

dd[-] [fd] ([newfd])

- list, dup2 or close filedescriptors
- useful to get STDIN from a file or write STDOUT to a file

```
$ r2 "frida:///bin/ls /"
[0x00000000]> \dmas output.txt
0x101db2510
[0x00000000]> \dxc open 0x101db2510 0x202
"0xb"
[0x00000000]> \dd 0xb 1
1
[0x00000000]> \resume
```

non-blocking trace

- **dt <addr> ..** trace list of addresses (backtrace)
- **dtr <addr> (<regs>...)** trace register values
- **dtf <addr> [fmt]** trace function and format params
- **dt-** clear all tracing

```
[0x00000000]> \dtf objc_msgSend xz
true
[0x00000000]> objc_msgSend 0x60800005afd0,"count" = 0x2
objc_msgSend 0x60800005afd0,"objectAtIndex:" = 0x61000013c3e0
objc_msgSend 0x60800005afd0,"objectAtIndex:" = 0x61800013f040
objc_msgSend 0x7fffab5a3c58,"self" = 0x7fffab5a3c58
objc_msgSend 0x7fffab5a3cd0,"self" = 0x7fffab5a3cd0
objc_msgSend 0x7fffab5a3cf8,"mutablePlaceholder" = 0x6000000138a0
objc_msgSend 0x7fffab5a3cd0,"allocWithZone:" = 0x6000000138a0
```

breakpoints

- **db <addr>|<symbol>** place a "breakpoint"
- **db- <addr>|*** remove breakpoints
- **dc** continue breakpoints or resume a spawned process

```
[0x00000000]> \db exit
{
  "0x7fffa5d9647b": {
    "name": "exit",
    "stopped": false,
    "address": "0x7fffa5d9647b",
    "continue": false,
    "handler": {}
  }
}
[0x00000000]> \eval breakpoints~{0x7fffa5d9647b.stopped}
false
[0x00000000]> \dc
resumed spawned process.
[0x00000000]> \eval breakpoints~{0x7fffa5d9647b.stopped}
true
[0x00000000]> \dc
Continue 1 thread(s).
[0x00000000]>
```

stalker trace

- **dtSf[*j] [sym|addr]** trace function using the stalker
- **dtS[*j] seconds** trace all threads for given seconds

- these commands are blocking
- they're guaranteed to finish
- specific r2frida eval configuration variables

stalker config

```
[0x00000000]> \e~stalker
e stalker.timeout=300
e stalker.event=compile
e stalker.in=raw
[0x00000000]> \e stalker.event=?
Specify the event to use when stalking, possible values:

    call            trace calls
    ret             trace returns
    exec            trace every instruction
    block           trace each basic block execution
    compile         trace basic blocks once

[0x00000000]> \e stalker.in=?
Restrict stalker results based on where the event
has originated:

    raw             stalk everywhere (the default)
    app             stalk only in the app module
    modules         stalk in app module and libs
```

stalker demo

extending r2frida

run frida code

- **eval <code>** run inline js in the agent
- **[space] <code>** run inline Cycript-style code (via mjolner)
- run js script **. path/to/script.js**

```
[0x00000000]> \eval new ApiResolver('objc').enumerateMatchesSync('*[* *password*]')~{[0]}
{"name":"+[TFNTwitterAccount passwordResetURLStringForUsername:]","address":"0x101175d80"}

[0x00000000]> \ [[[UIApplication sharedApplication] windows][0] rootViewController]
#"<TFNScreenBoundsLockedContainerViewController: 0x1058426d0>"
```

plugins

- js files (es5 syntax, for now), run in Frida agent
- register themselves as plugins
- access to all Frida features
- each command returns a string

```
'use strict';

r2frida.pluginRegister('resolver', function (commandName) {
  if (commandName === 'res0') {
    return function (args) {
      var query = args.join(' ');
      return new ApiResolver('objc').enumerateMatchesSync(query)
        .map(function (match) {
          return match.address + '\t' + match.name;
        })
        .join('\n');
    }
  }
});
```

plugins

- load them using **\. path/to/plugin.js** just like a generic frida script
- then use the provided commands directly
- unload with **\.-name**, list with **\.** without args

```
$ r2 frida://Twitter
[0x00000000]> \. resolver.js
true
[0x00000000]> \resO *[* sharedTwitter]
0x108018e6b     +[Twitter sharedTwitter]
[0x00000000]> \.
resolver
[0x00000000]> \.-resolver
true
[0x00000000]>
```

questions?