FRAPL
Next Generation Reverse Engineering Framework

Alex Hude

Max Bazaliy

October 22-23, 2016
Who we are

Alex Hude
- Melbourne, Australia
- Blackmagic Design
- Hardware, XNU
- Fried Apple team

Max Bazaliy
- Kyiv, Ukraine
- Lookout
- XNU, Linux, LLVM
- Fried Apple team
Modern Reverse Engineering

Static approach
- Disassemblers
- Code analyzers
- Decompilers
- IDA as a choice

Dynamic approach
- Code instrumentation
- Dynamic analyzers
- Debuggers
- Frida as a choice
Static analysis challenges

- Missed context (CPU registers, stack, memory)
- Hard to follow code execution flow (obfuscation)
- Hard to follow data flow (encryption)
- Hard to follow indirect function calls
Debugging challenges

- Anti debugging tricks
- Data loss during restarts
- Execution flow may be changed under debugging
- No way to hook/replace existing code easily
Dynamic instrumentation challenges

- Code disassembly still missed
- High learning curve
- Usually requires to write a lot of code
- Hard to maintain multiple things at a time
I NEED TO WRITE CODE

FOR MY FRIDA HOOKS
What is FRAPL?

FRAPL

= 

Frida scripts + FridaLink
Frida Scripts

- Node.js client (attach, spawn, RPC, script loading)
- Node.js server script (RPC, GCD, iOS/macOS bindings)
- Common operations wrappers (objc hooks etc)
- Utility functions (memory dumps, logging)
FridaLink

- IDA plugin that implements UI controls to Frida
- Socket protocol between IDA & Frida Client (JSON)
- RPC protocol for between Frida Client Server (JSON)
- FridaLink.js (Frida script)
FridaLink architecture

HOST

IDA
FridaLink plugin
python

Frida Client
Node.js

[ JSON ]
socket (localhost)

[ JSON ]
Frida RPC

(REMOTE)

Target
Frida Server
Java Script

October 22-23, 2016
FridaLink goals

- Bring static analysis info from IDA to Frida
- Use dynamic info from Frida for IDA analysis
- Monitor runtime state directly from IDA
- Control Frida agent directly from IDA
FridaLink features

- Direct function hooks made easy
- Function replacement made easy
- Module loading made easy
- Custom scripts support
FridaLink features

- CPU context monitoring
- Memory monitoring
- SQLite database support
- Helpers and project save/restore
SO YOU TELLING ME

NO NEED TO WRITE SCRIPTS
FridaLink - Overall View
FridaLink – Hooks

- Instruction hooks
- Instruction breakpoints (hook with wait)
- IDB (local) function hooks
- Import function hooks
FridaLink – Function Replacement

Replace local function

Replace Import function
FridaLink – Module Loading

- Automatic (on backtrace)
- Manual
FridaLink – Custom Scripts

Execute custom script dialog
FridaLink – CPU Context Monitoring

Stack

CPU context

Backtrace

October 22-23, 2016
FridaLink – Memory Monitoring

Memory manager

Add new memory watchpoint

Memory content
FridaLink – SQLite Support

Set up DB

Query execution

Load script
FridaLink – Helpers and more

Address converter

FRAPL logs
macOS Application Demo

ZSH Profile (iMac) (node)

~ Projects > FRAPL ./create_project.sh -f ~/Projects/iTunes ; cd ~/Projects/iTunes
~ Projects > node ./Client.js -c theme_example.json -n iTunes ./server.js

FRAPL: establish FridaLink automatically
FRAPL: starting mode set to attach by name
FRAPL: target location set to local
FRAPL: bind export from FrAFridaLink.js
FRAPL: script source is loaded
FRAPL: process 'include' directives
FRAPL: include('FRAPL/FrACommon.js')
FRAPL: include('FRAPL/FrAServerCore.js')
FRAPL: include('FRAPL/FrAGCD.js')
FRAPL: include('FRAPL/FrADfch.js')
FRAPL: include('FRAPL/FrAUtil.js')
FRAPL: include('FRAPL/FrAFridaLink.js')
FRAPL: attaching to target by name...
FRAPL: server script created
FRAPL: message listener set
FRAPL: server script loaded
FRAPL: FridaLink established
FRAPL: Module list request complete
FRAPL: Delete all memory ranges from monitor
FRAPL: Remove all FridaLink instruction hooks
FRAPL: Remove all FridaLink function hooks

October 22-23, 2016
iOS Application Demo

```
~ Projects  FRAPL
~ Projects  iTunes

FRAPL: establish FridaLink automatically
FRAPL: starting mode set to attach by PID
FRAPL: target location set to remote
FRAPL: bind export from FrAFridaLink.js
FRAPL: script source is loaded
FRAPL: process 'include' directives
FRAPL: include('FRAPL/FrACommon.js')
FRAPL: include('FRAPL/FrAServerCore.js')
FRAPL: include('FRAPL/FrAGCD.js')
FRAPL: include('FRAPL/FrADlfcn.js')
FRAPL: include('FRAPL/FrAUtilities.js')
FRAPL: include('FRAPL/FrAFridaLink.js')
FRAPL: attaching to target by PID...
FRAPL: server script created
FRAPL: message listener set
FRAPL: server script loaded
FRAPL: FridaLink established
FRAPL: Module list request complete
FRAPL: Delete all memory ranges from monitor
FRAPL: Remove all FridaLink instruction hooks
FRAPL: Remove all FridaLink function hooks
```
eta son
Future plans

- Kernel support
- Windows support?
- Android support?
- Hack the planet!
@getorix
@mbazaliy