

Cross-platform reversing with Frida

Ole André Vadla Ravnås

Motivation

- Existing tools often not a good fit for the task at hand
- Creating a new tool usually takes too much effort
- Short feedback loop: reversing is an iterative process
- Use one toolkit for multi-platform instrumentation
- Future remake of oSpy

Cross-platform reversing with Frida

oSpy

The screenshot shows the oSpy application interface. At the top, there is a menu bar with 'File', 'Edit', 'Go', 'Capture', 'Options', 'View', 'Tools', and 'Help'. Below the menu bar is a filter field and a search field containing 'Find: ASCII string'. The main area is a table of network events.

Index	Type	Timestamp	FunctionName	ReturnAddress	Sender	Description	Comment
232	📍	20:50:16	getaddrinfo	0x771c6575 [WININET.dll]	msnmsgr.exe [pid=3468, tid=2180]	nodename=login.live.com, servname=NULL	
235	📍	20:50:16	getaddrinfo	0x771c6575 [WININET.dll]	msnmsgr.exe [pid=3468, tid=2180]	nodename=login.live.com, servname=NULL	
237	★	20:50:16	connect	CTCPNetworkLayer::ConnectToIP	msnmsgr.exe [pid=3468, tid=2372]	204.204.204.204:52428: connecting to 65.54.239.140:1863	
238	★	20:50:16	connect	0x771c818c [WININET.dll]	msnmsgr.exe [pid=3468, tid=2180]	0.0.0.0:3900: connecting to 65.54.183.202:443	
▶ 307	➡	20:50:19	send	CTCPNetworkLayer::Send	msnmsgr.exe [pid=3468, tid=2372]	10.0.0.11:3901: Sent 33 bytes to 65.54.239.140:1863	VER
310	➡	20:50:19	send	CTCPNetworkLayer::Send	msnmsgr.exe [pid=3468, tid=2372]	10.0.0.11:3901: Sent 72 bytes to 65.54.239.140:1863	CVR
313	➡	20:50:19	send	CTCPNetworkLayer::Send	msnmsgr.exe [pid=3468, tid=2372]	10.0.0.11:3901: Sent 32 bytes to 65.54.239.140:1863	USR
321	⬅	20:50:19	recv	CTCPNetworkLayer::OnSocketRead	msnmsgr.exe [pid=3468, tid=2372]	10.0.0.11:3901: Received 33 bytes from 65.54.239.140:1863	VER
325	⬅	20:50:19	recv	CTCPNetworkLayer::OnSocketRead	msnmsgr.exe [pid=3468, tid=2372]	10.0.0.11:3901: Received 197 bytes from 65.54.239.140:1863	XFR
327	➡	20:50:19	SecureSend	0x7721d77d [WININET.dll]	msnmsgr.exe [pid=3468, tid=2180]	10.0.0.11:3900: Sent 546 bytes to 65.54.183.202:443	<POST /RST.srf => 200 OK
329	➡	20:50:19	SecureSend	0x7721d77d [WININET.dll]	msnmsgr.exe [pid=3468, tid=2180]	10.0.0.11:3900: Sent 3525 bytes to 65.54.183.202:443	...POST /RST.srf => 200 OK...
335	🚫	20:50:19	closesocket	CTCPNetworkLayer::OnSocketClose	msnmsgr.exe [pid=3468, tid=2372]	10.0.0.11:3901: connection to 65.54.239.140:1863 closed	
366	★	20:50:19	connect	CTCPNetworkLayer::ConnectToIP	msnmsgr.exe [pid=3468, tid=2372]	204.204.204.204:52428: connecting to 207.46.108.49:1863	
374	⬅	20:50:19	SecureRecei...	0x7721dce9 [WININET.dll]	msnmsgr.exe [pid=3468, tid=2180]	10.0.0.11:3900: Received 25 bytes from 65.54.183.202:443	...POST /RST.srf => 200 OK...
396	➡	20:50:20	send	CTCPNetworkLayer::Send	msnmsgr.exe [pid=3468, tid=2372]	10.0.0.11:3902: Sent 33 bytes to 207.46.108.49:1863	VER

Below the table is a hex dump for event #307:

```

0000: 56 45 52 20 31 20 4d 53 4e 50 31 35 20 4d 53 4e VER.1.MSNP15.MSN
0010: 50 31 34 20 4d 53 4e 50 31 33 20 43 56 52 30 0d P14.MSNP13.CVR0.
0020: 0a
    
```

On the right side, there is a 'Node' pane showing a tree view with '0' selected under the 'VER' node.

A 'Backtrace for #307 - send' window is open, showing a list of function addresses:

```

msnmsgr.exe:0x45d4ef (CTCPNetworkLayer::Send)
msnmsgr.exe:0x45d5b8
msnmsgr.exe:0x45d6b1 (CMNSConnection::SendNetMessage)
msnmsgr.exe:0x86cf1c
msnmsgr.exe:0x48d0d4
msnmsgr.exe:0x879460
msnmsgr.exe:0x4a9596
msnmsgr.exe:0x530070
    
```

The 'Backtrace' window also has a 'Go to address in IDA' button and a 'Close' button.

Cross-platform reversing with Frida

oSpy

The screenshot shows the oSpy application window with a menu bar (File, Edit, Capture, View, Help) and a search bar containing 'ASCII string'. Below the menu is a table of network events. The third event (index 3) is selected, showing a 'SecureSend' operation. Below the table is a hex dump of the captured data, with the ASCII representation of the data highlighted in yellow.

Index	Type	Timestamp	FunctionName	ReturnAddress	Sender
0		1:53:44 PM	getaddrinfo	0x71227d80 [WININET.dll]	iexplore.e
1		1:53:44 PM	connect	0x7122b945 [WININET.dll]	iexplore.e
2		1:53:44 PM	SecureSend	0x7123777b [WININET.dll]	iexplore.e
3		1:53:44 PM	SecureSend	0x7123777b [WININET.dll]	iexplore.e
4		1:53:45 PM	SecureReceive	0x71236ff7 [WININET.dll]	iexplore.e
5		1:53:45 PM	SecureReceive	0x71236ff7 [WININET.dll]	iexplore.e
6		1:53:45 PM	SecureReceive	0x71236ff7 [WININET.dll]	iexplore.e
7		1:53:45 PM	closesocket	0x7122c5b9 [WININET.dll]	iexplore.e

```

61 70 61 63 68 65 2e 73 74 72 75 74  org.apache.struts
67 6c 69 62 2e 68 74 6d 6c 2e 54 4f  s.taglib.html.TO
64 33 38 35 30 61 63 66 64 32 34 37  KEN=d3850acfd247
64 66 64 33 33 33 65 30 34 64 35 30  46d7dfd333e04d50
26 42 56 5f 53 65 73 73 69 6f 6e 49  50f8&BV_SessionI
40 40 30 38 39 30 32 34 38 32 39 38  D=@@@0890248298
38 31 31 33 38 32 37 40 40 40 40 26  .1268113827@@@&
6e 67 69 6e 65 49 44 3d 63 63 6b 63  BV_EngineID=cckc
6c 64 6a 68 6c 6d 6b 63 66 6c 67 63  adeildihlmkcflac
66 6b 67 64 67 6d 69 2e 30 26 75 73  ehfdkkgdgmj.0&us
6d 65 3d 72 61 79 6d 6f 6e 64 63 63  ername=raymondcc
73 77 6f 72 64 3d 74 65 73 74 70 61  &password=testpa
33 26 61 63 74 69 6f 6e 3d 4c 6f 67  ss123&action=Log
    
```

Cross-platform reversing with Frida

oSpy

The screenshot displays the oSpy application interface, which is used for analyzing network traffic. The main window is titled "Conversations" and shows a list of network conversations. The selected conversation is from 10.0.0.11:1145 to 207.46.104.25:1863. The traffic is categorized into VER (Verification), CVR (Content Verification), USR (User), and GCF (Get Content Filter) messages. The GCF message contains a list of policies, including a "SHIELDS" policy with various rules and regex patterns. The interface also shows the raw HTTP request and response for the selected conversation, including headers like "Accept", "User-Agent", "Host", "Content-Length", "Connection", and "Cache-Control". The response status is "HTTP/1.1 200 OK".

```
10.0.0.11:1145 <-> 207.46.104.25:1863 <UNKNOWN ENDPOINTS>
```

[14:20:40] VER 1 MSNP15 MSNP14 MSNP13 CVR0

CVR 2 0x0409 winnt 5.1 1386 MSNMSG 8.1.0178 msmsgs tryggve1@gmail.com

USR 3 SSO I tryggve1@gmail.com

[14:20:41] VER 1 MSNP15 MSNP14 MSNP13 CVR0

CVR 2 8.1.0178 8.1.0178 8.0.0787 http://msgr.d1service.microsoft.com/down

GCF 0 3866

```
<Policies>
  <Policy type="SHIELDS">
    <config>
      <shield>
        <cl1 maj="7" min="0" minbid="0" maxbid="9999" deny="" />
      </shield>
      <block>
        <hashes>
          </hashes>
        </block>
        <regex>
          <intext value="L1pCLN8pZ14q"/>
          <intext value="L1pCLNNJ14q"/>
          <intext value="L1pncn91CHBpyR1cmwCLN8ocC4q"/>
          <intext value="L1pncn91cc13dnyZwucGhwL1o"/>
          <intext value="L1pnyXsZCJ5XCSWAHAUKg="/>
          <intext value="L1pzdGFmZ1wucGhwL1o"/>
          <intext value="L1pwawNzXC5WAHAUKg="/>
          <intext value="L1pyB3R0Zw50b21hd9J1c1wud:MuKq="/>
        </regex>
      </config>
    </Policy>
  </Policies>
```

USR 3 SSO S MBI_KEY_OLD B13aM1UpBoxHOP230WHLGOQpd/G/dy1et80wadPjK8BuovJR

POST /RST.srf HTTP/1.1

Accept: text/*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; login.live.com
Host: login.live.com
Content-Length: 3525
Connection: Keep-Alive
Cache-Control: no-cache

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/04/soapsec/wsse/">
  <Header>
    <ps:AuthInfo xmlns:ps="http://schemas.microsoft.com/Passport/SoapEngine/2008/04/PassportAuth" />
    <ps:HostInApp>
      {7108E71A-9926-4FCB-BCC9-9A903F32E423}
    </ps:HostInApp>
    <ps:BinaryVersion>
      4
    </ps:BinaryVersion>
    <ps:UIVersion>
      4
    </ps:UIVersion>
    <ps:UITVersion>
      1
    </ps:UITVersion>
    <ps:Cookies>
    </ps:Cookies>
    <ps:RequestParams>
      AQAAAAIAAAB5YwQAAAAxMDQ0
    </ps:RequestParams>
    <ps:AuthInfo>
  </Header>
  <Body>
  </Body>
</Envelope>
```

[14:20:42] HTTP/1.1 100 Continue

HTTP/1.1 200 OK

Connection: close
Date: Sun, 04 Feb 2007 13:20:37 GMT
Server: Microsoft-IIS/6.0
PPServer: PPV: 30 H: BAYPPLOGN2B29 V: 0
X-Powered-By: ASP.NET
Content-Type: text/html; charset=iso-8859-1
Expires: Sun, 04 Feb 2007 13:19:37 GMT
Cache-Control: no-cache

What is Frida?

- Dynamic instrumentation toolkit
- Debug live processes
- Scriptable
 - **Execute your own debug scripts inside another process**
- Multi-platform
 - Windows, Mac, Linux, iOS, Android, QNX
- Open Source

Let's explore the basics

1) Build and run the test app that we will instrument:

```
#include <stdio.h>
#include <unistd.h>

Void
f (int n)
{
    printf ("Number: %d\n", n);
}

Int
main ()
{
    int i = 0;

    printf ("f() is at %p\n", f);

    while (1)
    {
        f (i++);
        sleep (1);
    }
}
```

```
$ clang hello.c -o hello
$ ./hello
f() is at 0x106a81ec0
Number: 0
Number: 1
Number: 2
...
```



2) Make note of the address of f(), which is 0x106a81ec0 here.

Basics 1/7: Hooking f() from Node.js

```
'use strict';

const co = require('co');
const frida = require('frida');
const load = require('frida-load');

let session, script;
co(function *() {
  session = yield frida.attach('hello');
  const source = yield load(require.resolve('./agent.js'));
  script = yield session.createScript(source);
  script.events.listen('message', message => {
    console.log(message);
  });
  yield script.load();
});
```

```
'use strict';

Interceptor.attach(ptr('0x106a81ec0'), {
  onEnter(args) {
    send(args[0].toInt32());
  }
});
```

```
$ # install Node.js 5.1
$ npm install co frida frida-load
$ node app.js
{ type: 'send', payload: 531 }
{ type: 'send', payload: 532 }
...
```



Address of f() goes here

Basics 1/7: Hooking f() from Python

```
import frida
import sys

session = frida.attach("hello")
script = session.create_script("""
Interceptor.attach(ptr("0x106a81ec0"), {
    onEnter(Args) {
        send(Args[0].toInt32());
    }
});
""")
def on_message(message, data):
    print(message)
script.on('message', on_message)
script.load()
sys.stdin.read()
```



Address of f() goes here

```
$ pip install frida
$ python app.py
{'type': 'send', 'payload': 531}
{'type': 'send', 'payload': 532}
...
```

Basics 2/7: Modifying function arguments

```
'use strict';  
  
const co = require('co');  
const frida = require('frida');  
const load = require('frida-load');  
  
let session, script;  
co(function *() {  
  session = yield frida.attach('hello');  
  const source = yield load(require.resolve('./agent.js'));  
  script = yield session.createScript(source);  
  yield script.load();  
});
```

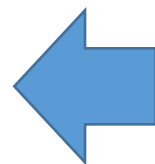
```
'use strict';  
  
Interceptor.attach(ptr('0x106a81ec0'), {  
  onEnter(args) {  
    args[0] = ptr("1337");  
  }  
});
```

```
$ node app.js
```

```
Number: 1281  
Number: 1282  
Number: 1337  
Number: 1337  
Number: 1337  
Number: 1337  
Number: 1296  
Number: 1297  
Number: 1298  
...
```



Once we stop it
the target is back to
normal



Address of f() goes here

Basics 3/7: Calling functions

```
'use strict';  
  
const co = require('co');  
const frida = require('frida');  
const load = require('frida-load');  
  
let session, script;  
co(function *() {  
  session = yield frida.attach('hello');  
  const source = yield load(require.resolve('./agent.js'));  
  script = yield session.createScript(source);  
  yield script.load();  
  yield session.detach();  
});
```

```
'use strict';  
  
const f = new NativeFunction(  
  ptr('0x10131fec0'), 'void', ['int']);  
f(1911);  
f(1911);  
f(1911);
```

```
$ node app.js
```

```
Number: 1281  
Number: 1282  
Number: 1911  
Number: 1911  
Number: 1911  
Number: 1283  
Number: 1284  
Number: 1285  
...
```



Address of f() goes here

Basics 4/7: Sending messages

```
'use strict';

const co = require('co');
const frida = require('frida');
const load = require('frida-load');

let session, script;
co(function *() {
  session = yield frida.attach('hello');
  const source = yield load(require.resolve('./agent.js'));
  script = yield session.createScript(source);
  script.events.listen('message', message => {
    console.log(message);
  });
  yield script.load();
});
```

```
'use strict';

send({
  user: {
    name: 'john.doe'
  },
  key: '1234'
});

oops;
```

```
$ node app.js
```

```
{ type: 'send',
  payload: { user: { name: 'john.doe' }, key: '1234' } }
{ type: 'error',
  description: 'ReferenceError: oops is not defined',
  stack: 'ReferenceError: oops is not defined\n at Object.1
(agent.js:10:1)\n at s (../../node_modules/browser-
pack/_prelude.js:1:1)\n at e (../../node_modules/browser-
pack/_prelude.js:1:1)\n at ../../node_modules/browser-
pack/_prelude.js:1:1',
  fileName: 'agent.js',
  lineNumber: 10,
  columnNumber: 1
}
```

Basics 5/7: Receiving messages

```
'use strict';

const co = require('co');
const frida = require('frida');
const load = require('frida-load');

let session, script;
co(function *() {
  session = yield frida.attach('hello');
  const source = yield load(require.resolve('./agent.js'));
  script = yield session.createScript(source);
  script.events.listen('message', message => {
    console.log(message);
  });
  yield script.load();
  yield script.postMessage({ magic: 21 });
  yield script.postMessage({ magic: 12 });
});
```

```
$ node app.js
```

```
{ type: 'send', payload: 42 }
{ type: 'send', payload: 36 }
```

```
'use strict';

let i = 2;
function handleMessage(message) {
  send(message.magic * i);
  i++;
  recv(handleMessage);
}
recv(handleMessage);
```

Basics 6/7: Blocking receives

```
'use strict';  
  
const co = require('co');  
const frida = require('frida');  
const load = require('frida-load');  
  
let session, script;  
co(function *() {  
  session = yield frida.attach('hello');  
  const source = yield load(require.resolve('./agent.js'));  
  script = yield session.createScript(source);  
  script.events.listen('message', message => {  
    const number = message.payload.number;  
    script.postMessage({ number: number * 2 });  
  });  
  yield script.load();  
});
```

```
'use strict';  
  
Interceptor.attach(ptr('0x106a81ec0'), {  
  onEnter(args) {  
    send({ number: args[0].toInt32() });  
    const op = recv(reply => {  
      args[0] = ptr(reply.number);  
    });  
    op.wait();  
  }  
});
```

```
$ node app.js
```

```
Number: 2183  
Number: 2184  
Number: 4370  
Number: 4372  
Number: 4374  
Number: 4376  
Number: 4378  
Number: 2190  
Number: 2191  
Number: 2192  
...
```



Once we stop it
the target is back to
normal



Address of f() goes here

Basics 7/7: RPC

```
'use strict';

const co = require('co');
const frida = require('frida');
const load = require('frida-load');

let session, script;
co(function *() {
  session = yield frida.attach('hello');
  const source = yield load(require.resolve('./agent.js'));
  script = yield session.createScript(source);
  yield script.load();
  const api = yield script.getExports();
  const result = yield api.disassemble('0x106a81ec0');
  console.log(result);
  yield session.detach();
});
```

```
'use strict';

rpc.exports = {
  disassemble(address) {
    return Instruction.parse(ptr(address)).toString();
  }
};
```

```
$ node app.js
push rbp
$
```

 Address of f() goes here

Launch and spy on iOS app

```
'use strict';

const co = require('co');
const frida = require('frida');
const load = require('frida-load');

let session, script;
co(function *() {
  const device = yield frida.getUsbDevice();
  const pid = yield device.spawn(['com.apple.AppStore']);
  session = yield device.attach(pid);
  const source = yield load(require.resolve('./agent.js'));
  script = yield session.createScript(source);
  script.events.listen('message', message => {
    if (message.type === 'send' && message.payload.event === 'ready')
      device.resume(pid);
    else
      console.log(message);
  });
  yield script.load();
})
.catch(console.error);
```

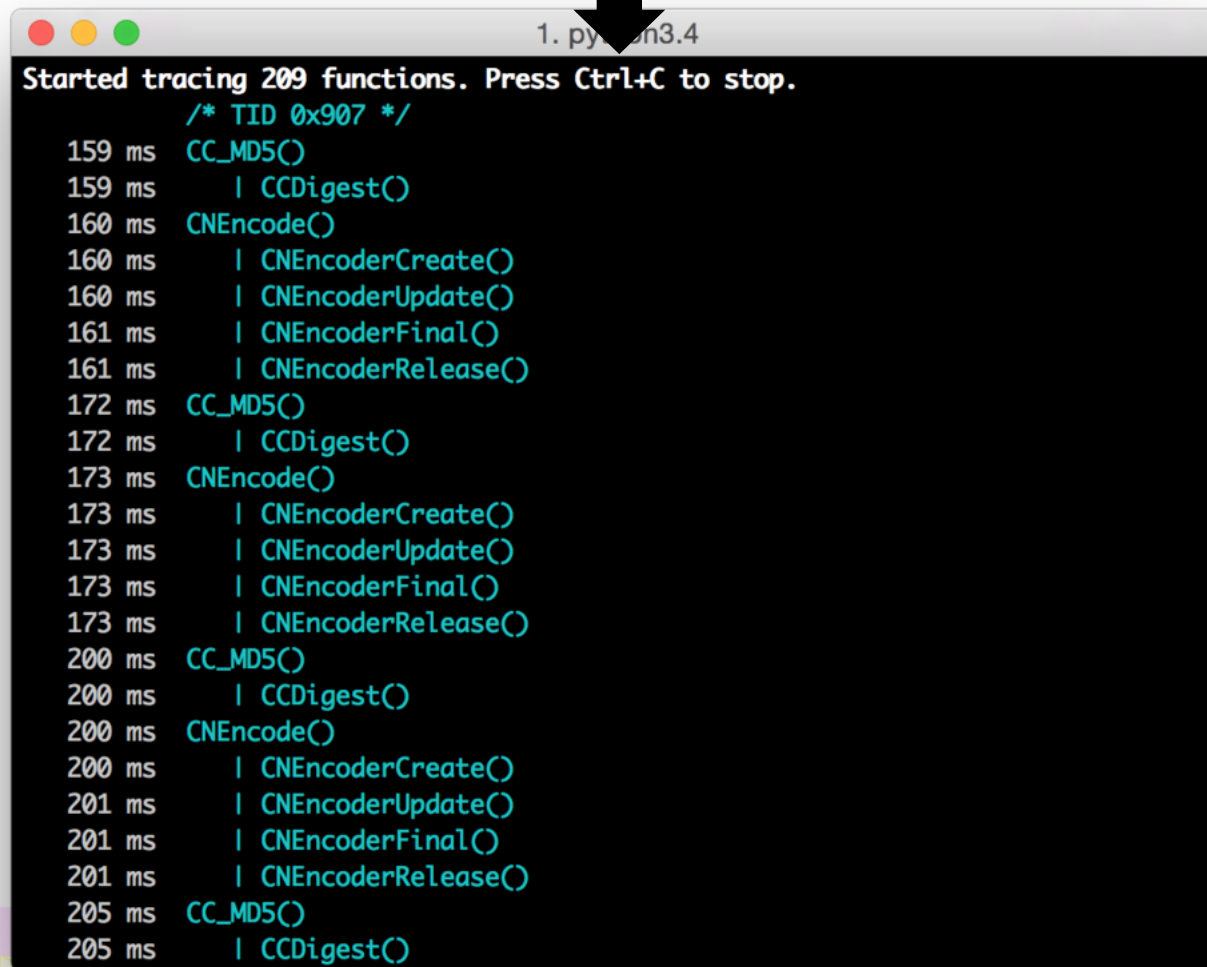
```
'use strict';

Module.enumerateExports('libcommonCrypto.dylib', {
  onMatch(e) {
    if (e.type === 'function') {
      try {
        Interceptor.attach(e.address, {
          onEnter(args) {
            send({ event: 'call', name: e.name });
          }
        });
      } catch (error) {
        console.log('Ignoring ' + e.name + ': ' + error.message);
      }
    }
  },
  onComplete() {
    send({ event: 'ready' });
  }
});
```

```
$ node app.js
{ type: 'send', payload: { event: 'call', name: 'CC_MD5' } }
{ type: 'send', payload: { event: 'call', name: 'CCDigest' } }
{ type: 'send', payload: { event: 'call', name: 'CNEncode' } }
...
```

But there's an app for that

```
$ sudo easy_install frida  
$ frida-trace -U -f com.apple.AppStore -I libcommonCrypto.dylib
```



```
Started tracing 209 functions. Press Ctrl+C to stop.  
/* TID 0x907 */  
159 ms CC_MD5()  
159 ms | CCDigest()  
160 ms CNEncode()  
160 ms | CNEncoderCreate()  
160 ms | CNEncoderUpdate()  
161 ms | CNEncoderFinal()  
161 ms | CNEncoderRelease()  
172 ms CC_MD5()  
172 ms | CCDigest()  
173 ms CNEncode()  
173 ms | CNEncoderCreate()  
173 ms | CNEncoderUpdate()  
173 ms | CNEncoderFinal()  
173 ms | CNEncoderRelease()  
200 ms CC_MD5()  
200 ms | CCDigest()  
200 ms CNEncode()  
200 ms | CNEncoderCreate()  
201 ms | CNEncoderUpdate()  
201 ms | CNEncoderFinal()  
201 ms | CNEncoderRelease()  
205 ms CC_MD5()  
205 ms | CCDigest()
```

Dump iOS UI

```
'use strict';

const co = require('co');
const frida = require('frida');
const load = require('frida-load');

let session, script;
co(function *() {
  const device = yield frida.getUsbDevice();
  const app = yield device.getFrontmostApplication();
  if (app === null)
    throw new Error("No app in foreground");
  session = yield device.attach(app.pid);
  const source = yield load(require.resolve('./agent.js'));
  script = yield session.createScript(source);
  script.events.listen('message', message => {
    console.log(message.payload.ui);
    session.detach();
  });
  yield script.load();
});
```

```
'use strict';

ObjC.schedule(ObjC.mainQueue, () => {
  const window = ObjC.classes.UIWindow.keyWindow();
  const ui = window.recursiveDescription().toString();
  send({ ui: ui });
});
```

```
$ node --harmony dump-ui.js
<UIWindow: 0x15fe3ca40; frame = (0 0; 375 667);
autoresize = W+H; gestureRecognizers = <NSArray:
0x17424c1e0>; layer = <UIWindowLayer: 0x17023dcc0>>
  | <UIView: 0x15fd2dbd0; frame = (0 0; 375 667);
autoresize = W+H; layer = <CALayer: 0x174432320>>
  | | <UIView: 0x15fe64250; frame = (0 0; 375 667);
autoresize = W+H; layer = <CALayer: 0x170235340>>
  | | | <UIView: 0x15fd506e0; frame = (0 0; 375 667);
...

```

Hold on a sec, what if I have many phones connected?

```
2. bash
Oles-MacBook:~ oleavr$ frida-ls-devices
Id                               Type      Name
-----
local                             local     Local System
f4c5ba319e6df557eeb1f3736904585801a2dfe7  tether    iPhone
tcp                                 remote    Local TCP
Oles-MacBook:~ oleavr$ frida-ps -D local | head -5
  PID  Name
-----
67711  1Password mini
  568  AirPlayUIAgent
54853  BezelUIServer
Oles-MacBook:~ oleavr$ frida-ps -D f4c5ba319e6df557eeb1f3736904585801a2dfe7 | head -5
  PID  Name
-----
6258  Cydia
5250  EasyPark
2006  Hangouts
Oles-MacBook:~ oleavr$
```

Which apps are installed?

```
2. bash
Oles-MacBook:~ oleavr$ frida-ps -Uai
PID  Name                Identifier
----  -
6258  Cydia                com.saurik.Cydia
5250  EasyPark             net.easypark.app
2006  Hangouts             com.google.hangouts
6172  I R C Cloud          com.irccloud.IRCcloud
6111  LinkedIn             com.linkedin.Linkedin
559   Mail                 com.apple.mobilemail
6418  Messenger            com.facebook.Messenger
1666  Safari               com.apple.mobilesafari
5962  Settings             com.apple.Preferences
2313  Slack                com.tinyspeck.chatlyio
6012  Snapchat             com.toyopagroup.picaboo
6053  WhatsApp             net.whatsapp.WhatsApp
-    1Password            com.agilebits.onepassword-ios
-    Activity             com.apple.Fitness
-    Afterlight          com.simonfilip.AfterGlow
-    Airbnb              com.airbnb.app
-    App Store           com.apple.AppStore
-    Authenticator       com.google.Authenticator
-    BMSSM               com.cactusapp.aai
-    BankID              no.bankid.client
-    BensinPris          no.bitfactory.BensinPris.Release
-    Big Day             com.whatisid.bigday
```


Cross-platform reversing with Frida

Speaking of apps, we also have a REPL:

```

1. Python
Oles-MacBook:~ oleavr$ frida Twitter

( )
| |
| |   Frida 6.0.1 - A world-class dynamic instrumentation framework
| |
| `-' |   Commands:
| |   help      -> Displays the help system
| |   object?   -> Display information about 'object'
| |   exit/quit -> Exit
| |
| |   More info at http://www.frida.re/docs/home/
| |
| |
| |

[Local::ProcName::Twitter]-> ObjC.classes.
ABActiveTextRanges
ABBackgroundProxy
ABCAAnimationCallbackDelegate
ABFileManager
ABFlavoredRange
ABGPS
ABGPSRequestInfo
ABGroupedRowView
ABHTTPMultipartFormData

```

Cross-platform reversing with Frida

The REPL is your best friend for prototyping scripts

```
1. Python
Oles-MacBook:7-repl oleavr$ frida -n Twitter -l agent.js

( )
| | Frida 6.0.1 - A world-class dynamic instrumentation framework
| |
|`-'| Commands:
| | help -> Displays the help system
| | object? -> Display information about 'object'
| | exit/quit -> Exit
| |
| | More info at http://www.frida.re/docs/home/
| |
|`-.'

Attaching...
Agent running on darwin/x64
[Local::ProcName::Twitter]-> stats()
App has 6330 classes loaded!
undefined
[Local::ProcName::Twitter]-> %reload
Agent running on darwin/x64
[Local::ProcName::Twitter]-> stats()
Twitter has 6330 classes loaded!
undefined
[Local::ProcName::Twitter]->
```

Uninstall iOS app

```
'use strict';  
  
const LSAApplicationWorkspace = ObjC.classes.LSAApplicationWorkspace;  
const onProgress = new ObjC.Block({  
  retType: 'void',  
  argTypes: ['object'],  
  implementation: (progress) => {  
    console.log('onProgress: ' + progress);  
  }  
});  
  
function uninstall(appId) {  
  const workspace = LSAApplicationWorkspace.defaultWorkspace();  
  return workspace.uninstallApplication_withOptions_usingBlock_(appId, null, onProgress);  
}
```

```
$ frida -U SpringBoard -l agent.js
```

Interacting with Objective-C

- *ObjC.available* – is the runtime present?
- *new ObjC.Object(ptr('0x1234'))* – interact with object at 0x1234
- *ObjC.classes* – all loaded classes
 - *Object.keys(ObjC.classes)* to list all names
 - *if ('UIView' in ObjC.classes)* to check for class presence
- *ObjC.protocols* – all loaded protocols
- *[NSURL URLWithString:foo relativeToURL:bar]* translates to *ObjC.classes.NSURL.URLWithString_relativeToURL_(foo, bar)*
- *NSURL['- setResourceValues:error:']* to access instance methods from its class
- Assign to *.implementation* to replace a method
- *ObjC.choose()* – scan heap looking for Objective-C instances

Hooking Objective-C methods

The swizzling way:

```
const method = ObjC.classes.AVAudioSession['- setCategory:error:'];
const originalImpl = method.implementation;
method.implementation = ObjC.implement(method, function (self, sel, category, error) {
    return originalImpl(self, self, category, error);
});
```

The low-level way:

```
const method = ObjC.classes.AVAudioSession['- setCategory:error:'];
Interceptor.attach(method.implementation, {
    onEnter(args) {
    },
    onLeave(retval) {
    }
});
```

Android instrumentation

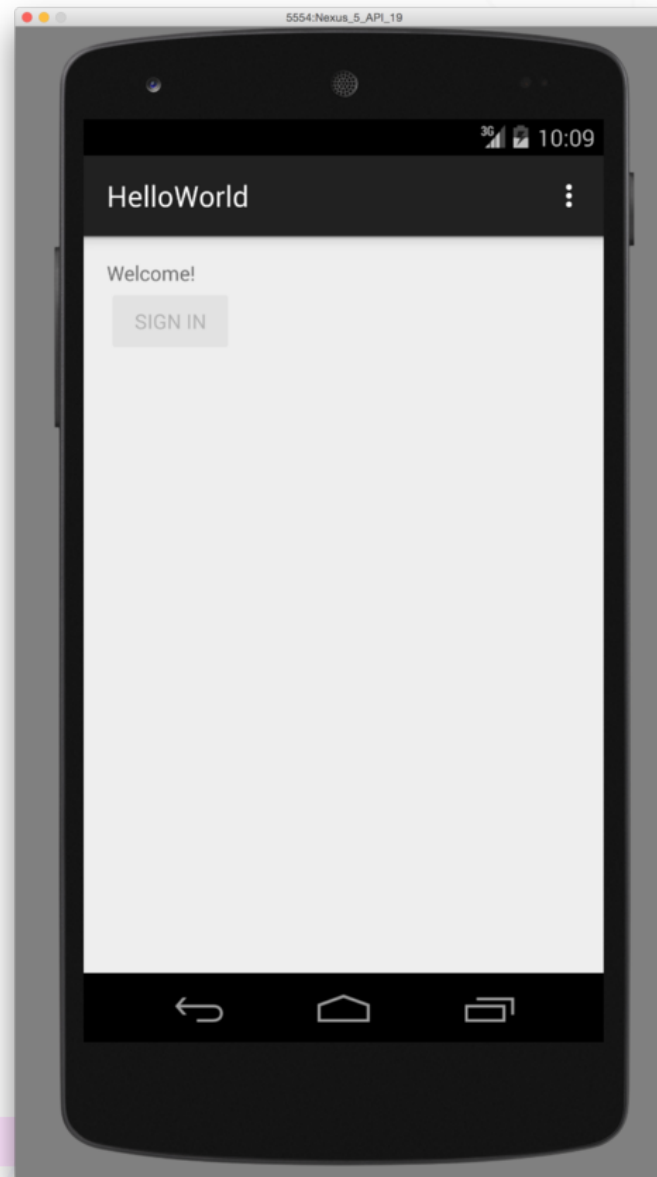
```
'use strict';

const co = require('co');
const frida = require('frida');
const load = require('frida-load');

let session, script;
co(function *() {
  const device = yield frida.getUsbDevice();
  session = yield device.attach('re.frida.helloworld');
  const source = yield load(require.resolve('./agent.js'));
  script = yield session.createScript(source);
  script.events.listen('message', message => {
    console.log(message);
  });
  yield script.load();
});
```

```
'use strict';

Dalvik.perform(() => {
  const MainActivity = Dalvik.use(
    're.frida.helloworld.MainActivity');
  MainActivity.isRegistered.implementation = () => {
    console.log('isRegistered() w00t');
    return true;
  };
});
```



Interacting with Java

- *Java.available* – is the runtime present?
- *Java.perform(fn)* – interact with the Java VM from the given callback
- *Java.cast(ptr('0x1234'), Java.use("android.os.Handler"))* – interact with object at 0x1234
- Constructors are exposed as *\$new()*, and overloads can be selected as with any methods:
Handler.\$new.overload("android.os.Looper").call(Handler, looper)
- *Java.enumerateLoadedClasses()* – all loaded classes
- Assign to *.implementation* to replace a method
- *Java.choose()* – scan heap looking for Java instances

Hooking Java methods

```
const Handler = classFactory.use("android.os.Handler");  
  
Handler.dispatchMessage.implementation = function (msg) {  
  // Chain up to the original implementation  
  this.dispatchMessage(msg);  
};
```

Early instrumentation

1. `pid = frida.spawn(["/bin/lis"])`
2. `session = frida.attach(pid)`
3. `script = session.create_script("your script")`
4. <apply instrumentation> – recommend RPC for this: `script.exports.init()`
5. `frida.resume(pid)` – application's main thread will enter `main()`

For mobile apps specify its identifier: `spawn(["com.apple.AppStore"])`

Forgot what it was? Use `frida-ps -ai`

How about implicitly spawned processes? Enter spawn gating!

1. *device.on('spawned', on_spawned)*
2. *device.enable_spawn_gating()*
3. *device.enumerate_pending_spawns()*

Examples:

<https://gist.github.com/oleavr/ae7bcbbb9179852a4731>

Only implemented for iOS and Android.

Backtraces

```
'use strict';

Interceptor.attach(Module.findExportByName('libSystem.B.dylib', 'connect'), {
  onEnter() {
    console.log('connect called from:\n\t' +
      Thread.backtrace(this.context, Backtracer.ACCURATE).join('\n\t'));
  }
});
```

```
$ frida -n Spotify -l agent.js
[Local::PID::66872]-> connect called
from:      0x106de3a36
           0x106de6851
           0x10753d092
           0x10753ecd1
```

Backtraces with debug symbols

```
'use strict';

Interceptor.attach(Module.findExportByName('libSystem.B.dylib', 'connect'), {
  onEnter() {
    console.log('connect called from:\n\t' +
      Thread.backtrace(this.context, Backtracer.ACCURATE)
        .map(DebugSymbol.fromAddress)
        .join('\n\t'));
  }
});
```

```
$ frida -n Spotify -l agent.js
[Local::ProcName::Twitter]-> connect called from:
0x7fff9b5dd6b1 libsystem_network.dylib!get_host_counts
0x7fff9b60ee4f libsystem_network.dylib!tcp_connection_destination_create
0x7fff9b5e2eb7 libsystem_network.dylib!tcp_connection_destination_add
0x7fff9b5e2e5a libsystem_network.dylib!__tcp_connection_start_host_block_invoke
0x7fff9b6079a5 libsystem_network.dylib!tcp_connection_host_resolve_result
0x7fff9ece7fe0 libsystem_dnssd.dylib!handle_addrinfo_response
```


Best practices

- Use Node.js bindings to *frida-load* your agent.js so you can:
 - Split your script into multiple files
 - Use Frida modules from the community
 - Reuse thousands of modules from npm
- Use ES6 features to write modern JavaScript – Frida support it
- REPL is great for prototyping with `-l` and `%reload`

Injecting errors

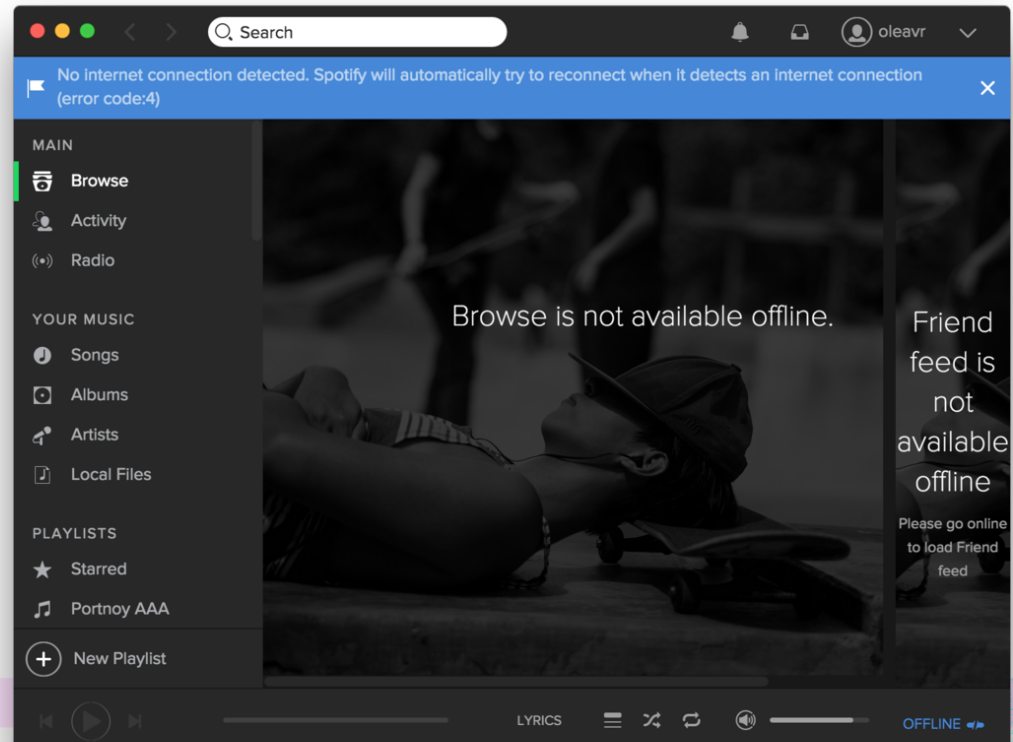
```
'use strict';

const co = require('co');
const frida = require('frida');
const load = require('frida-load');

let session, script;
co(function *() {
  session = yield frida.attach(process.argv[2]);
  const source = yield load(require.resolve('./agent.js'));
  script = yield session.createScript(source);
  script.events.listen('message', message => {
    console.log(message);
  });
  yield script.load();
})
.catch(console.error);
```

```
Interceptor.replace(connect,
  new NativeCallback((socket, address, addressLen) => {
    ...
    if (port === 80 || port === 443 || port === 4070) {
      this.errno = ECONNREFUSED;
      return -1;
    } else {
      return connect(socket, address, addressLen);
    }
  }));
```

```
$ node app.js Spotify
connect() family=2 ip=78.31.9.101 port=80 blocking!
connect() family=2 ip=193.182.7.242 port=80 blocking!
connect() family=2 ip=194.132.162.4 port=443 blocking!
connect() family=2 ip=194.132.162.4 port=80 blocking!
connect() family=2 ip=194.132.162.212 port=80 blocking!
connect() family=2 ip=194.132.162.196 port=4070 blocking!
connect() family=2 ip=193.182.7.226 port=443 blocking!
```



All calls between two recv() calls

```
'use strict';

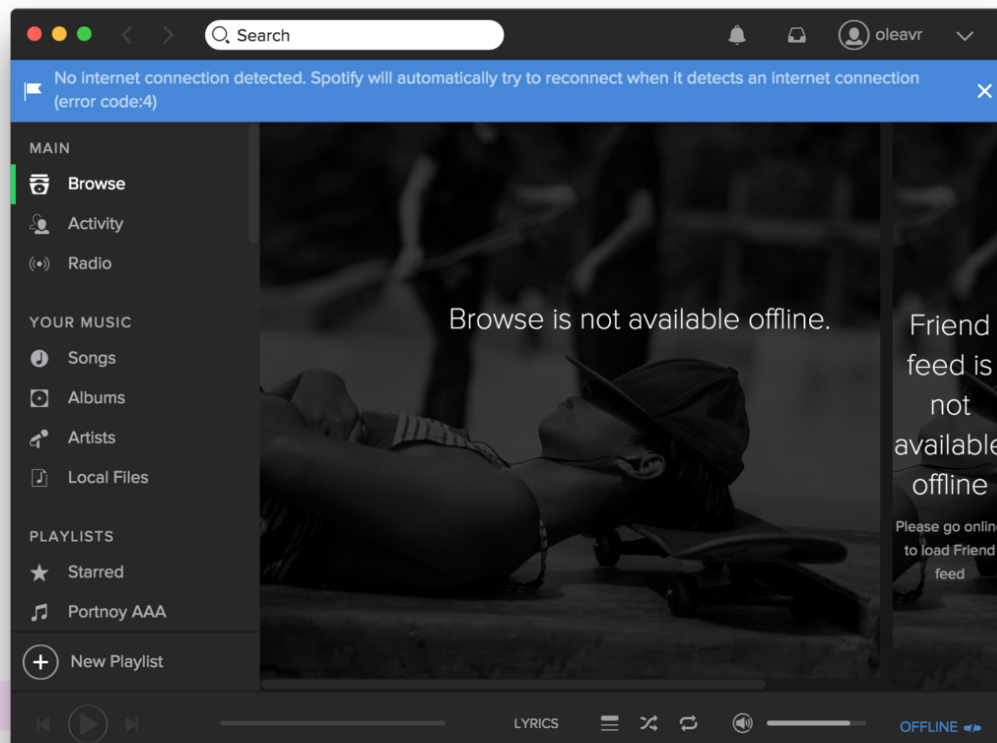
const co = require('co');
const frida = require('frida');
const load = require('frida-load');

co(function *() {
  ...
  yield script.load();
});
```

```
'use strict';

...
Stalker.follow({
  events: {
    call: true
  },
  onReceive(events) {
    blobs.push(events);
    if (state === COLLECTING) {
      sendResult();
      state = DONE;
    }
  }
});
```

```
$ node app.js Spotify
connect() family=2 ip=78.31.9.101 port=80 blocking!
connect() family=2 ip=193.182.7.242 port=80 blocking!
connect() family=2 ip=194.132.162.4 port=443 blocking!
connect() family=2 ip=194.132.162.4 port=80 blocking!
connect() family=2 ip=194.132.162.212 port=80 blocking!
connect() family=2 ip=194.132.162.196 port=4070 blocking!
connect() family=2 ip=193.182.7.226 port=443 blocking!
```



Cross-platform reversing with Frida

Questions?

Twitter: @oleavr

Thanks!

Please drop by **#frida** on FreeNode, and don't forget to join our mailing list:
<https://groups.google.com/d/forum/frida-dev>