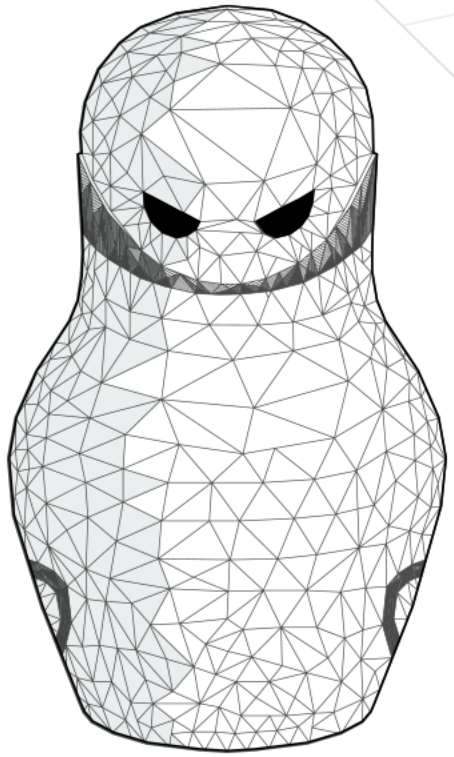


2015 ZERO NIGHTS

Cross-platform reversing with Frida

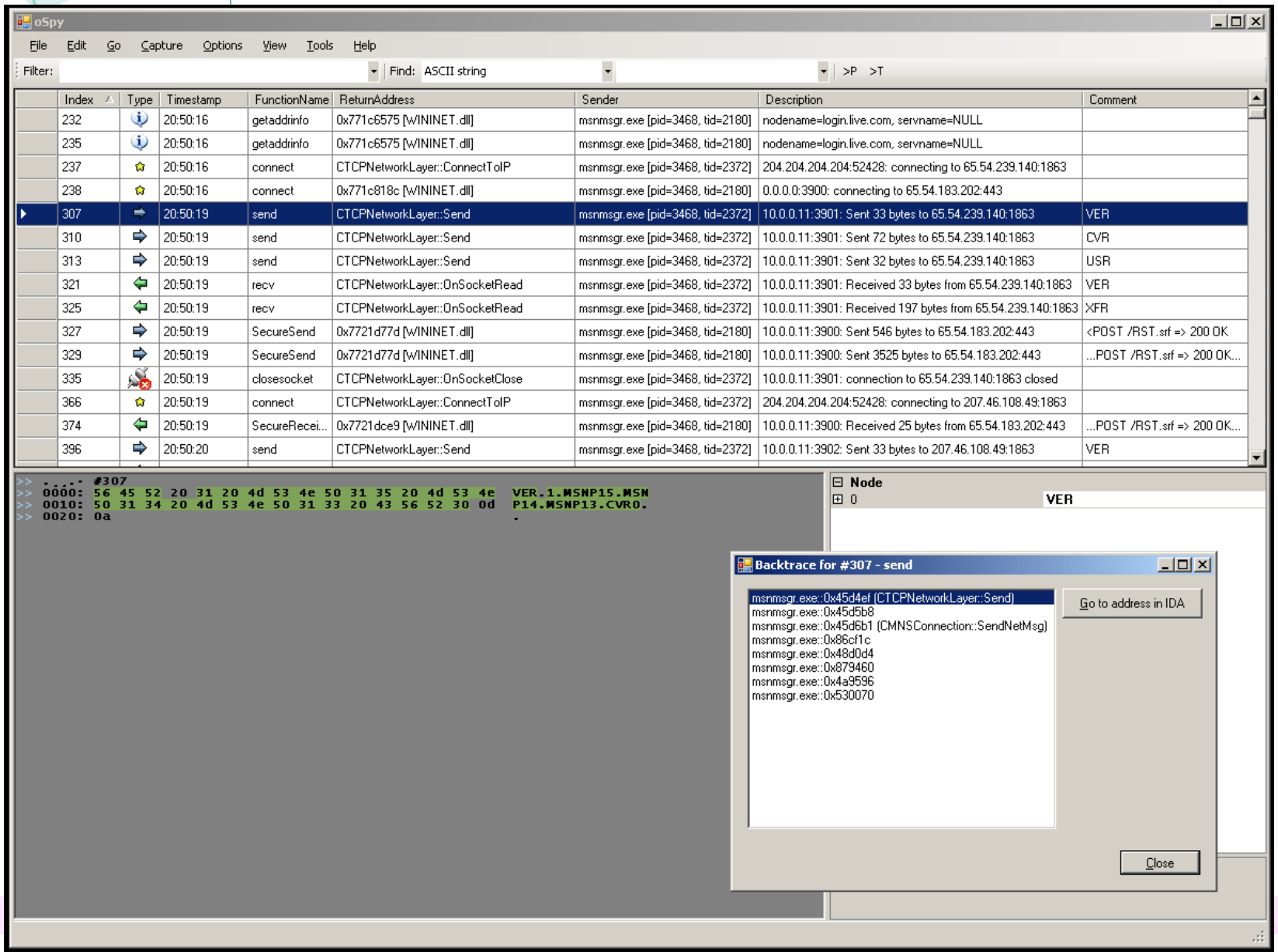
Ole André Vadla Ravnås

Motivation

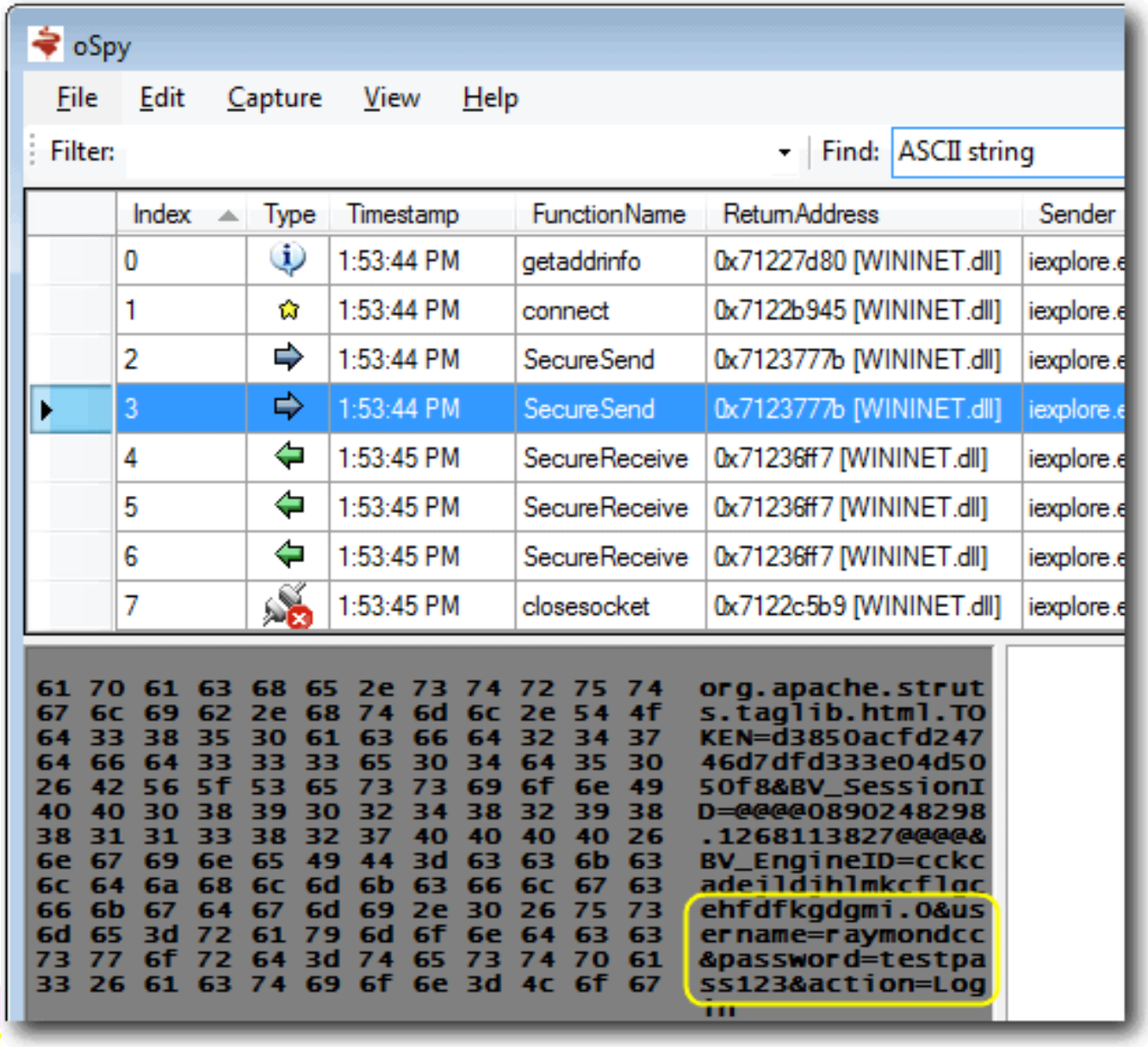
- Existing tools often not a good fit for the task at hand
- Creating a new tool usually takes too much effort
- Short feedback loop: reversing is an iterative process
- Use one toolkit for multi-platform instrumentation
- Future remake of oSpy

Cross-platform reversing with Frida

oSpy

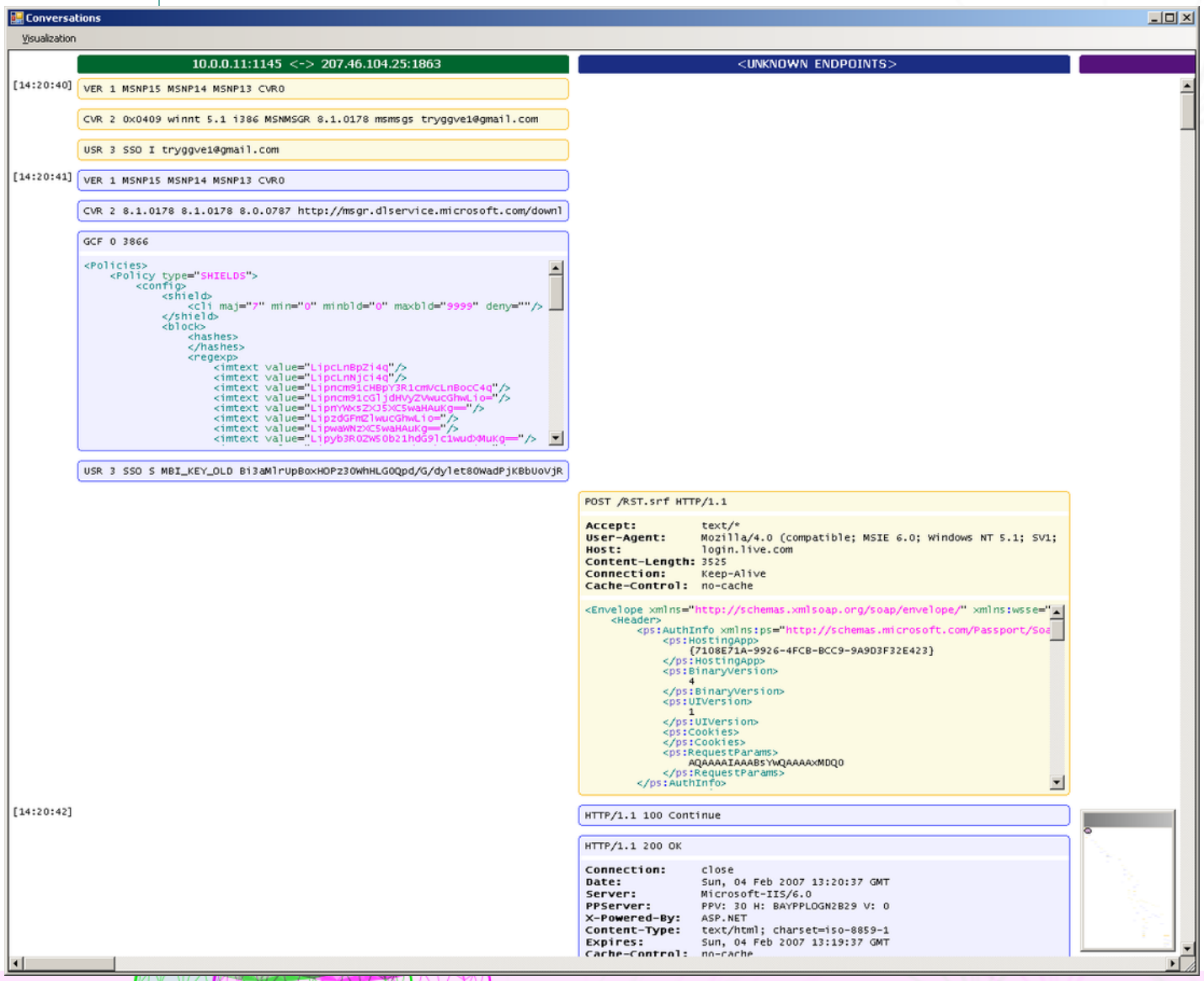


oSpy


oSpy
File Edit Capture View Help
Filter:
Find: ASCII string
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.strut
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 adeildihlmkcflgc
66 6b 67 64 67 6d 69 2e 30 26 75 73 ehfdfkgdgmi.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



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:

```c
#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();
});
```

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

```
'use strict';

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

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();
});
```

```
$ 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

```
'use strict';

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

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();
});
```

```
$ node app.js
```

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

```
'use strict';

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

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();
});
```

```
$ 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

```
'use strict';

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

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

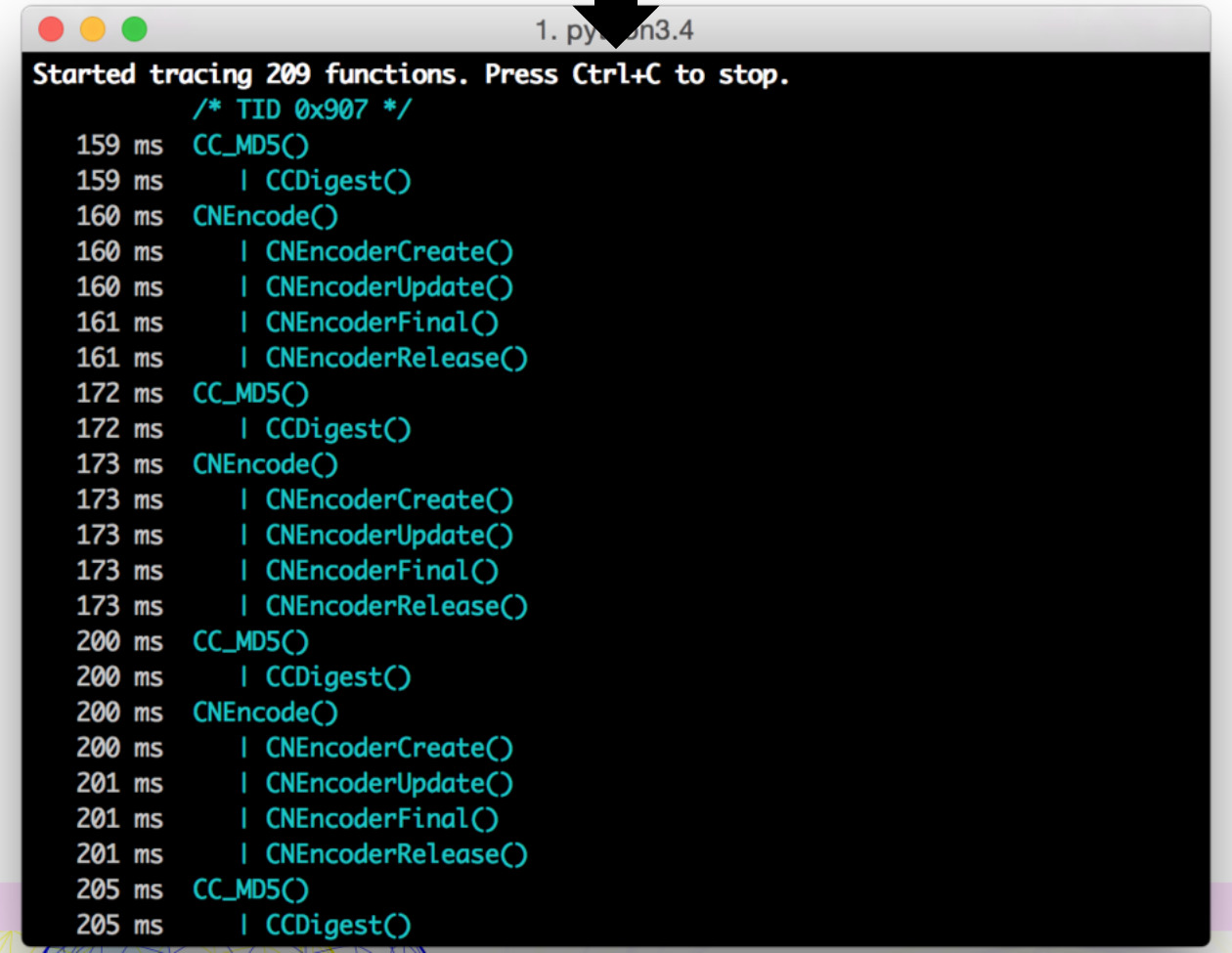
```
'use strict';

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
```



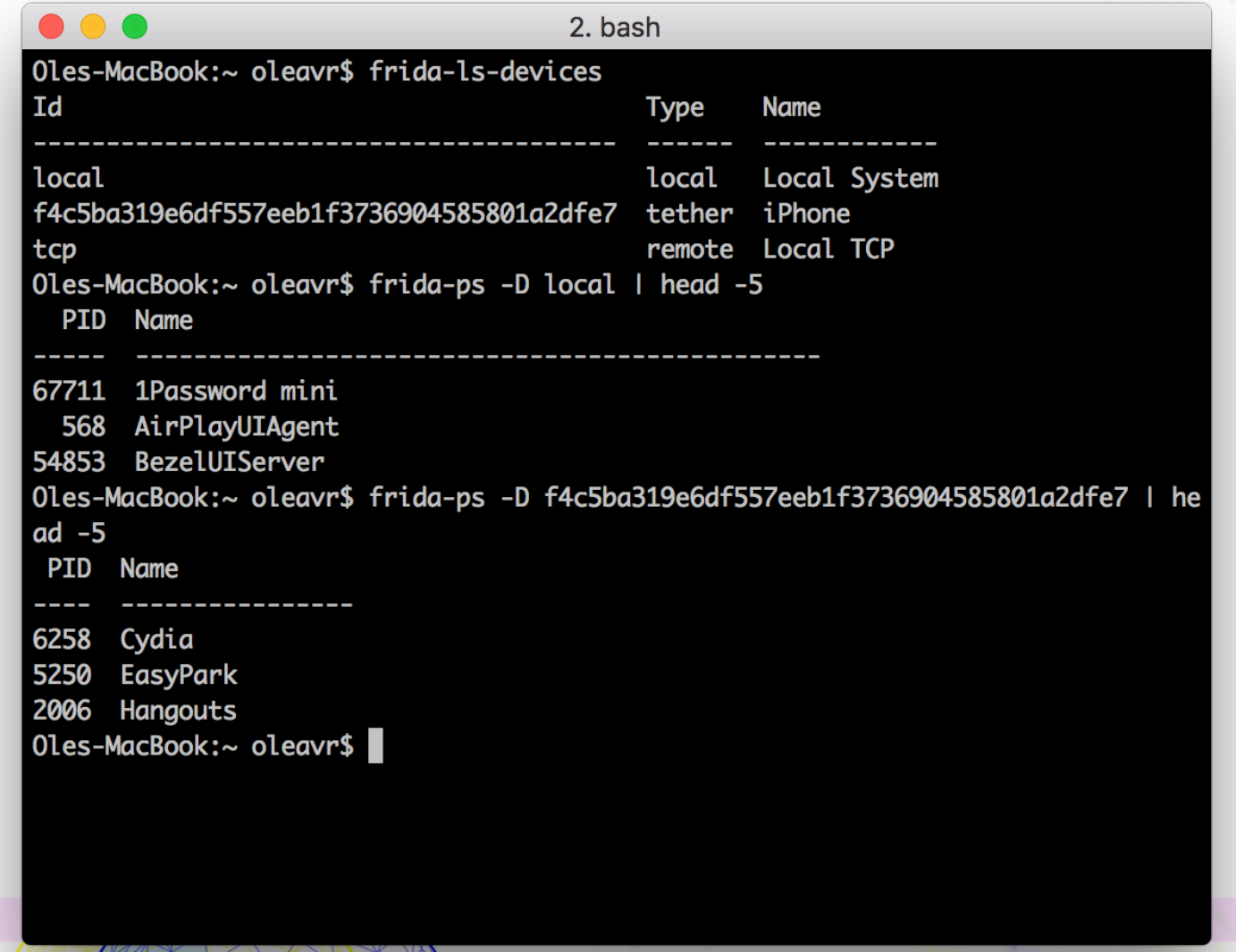
Dump iOS UI

```
'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();
});
```

```
$ 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);
…
```

```
'use strict';

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

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

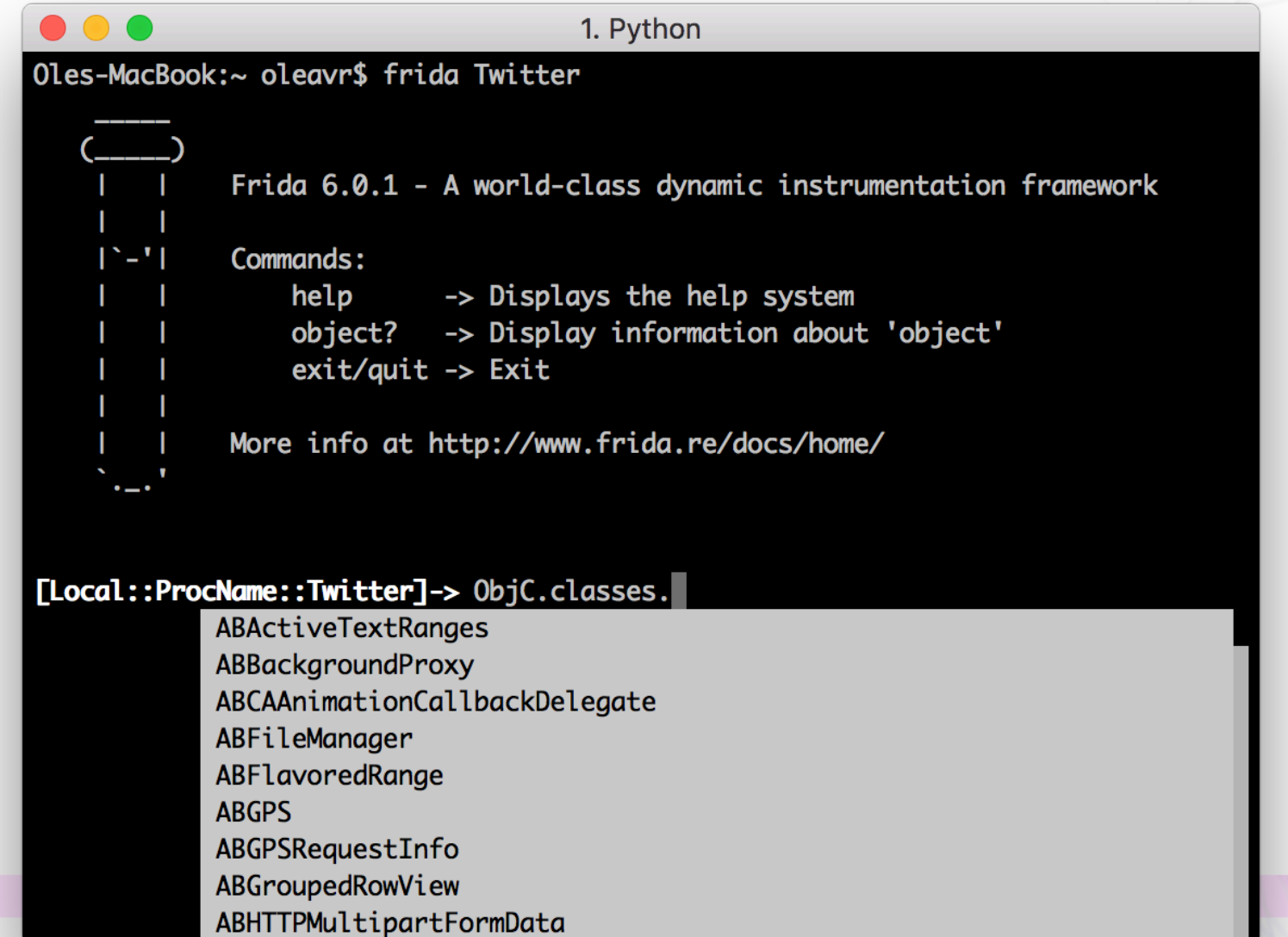


Which apps are installed?

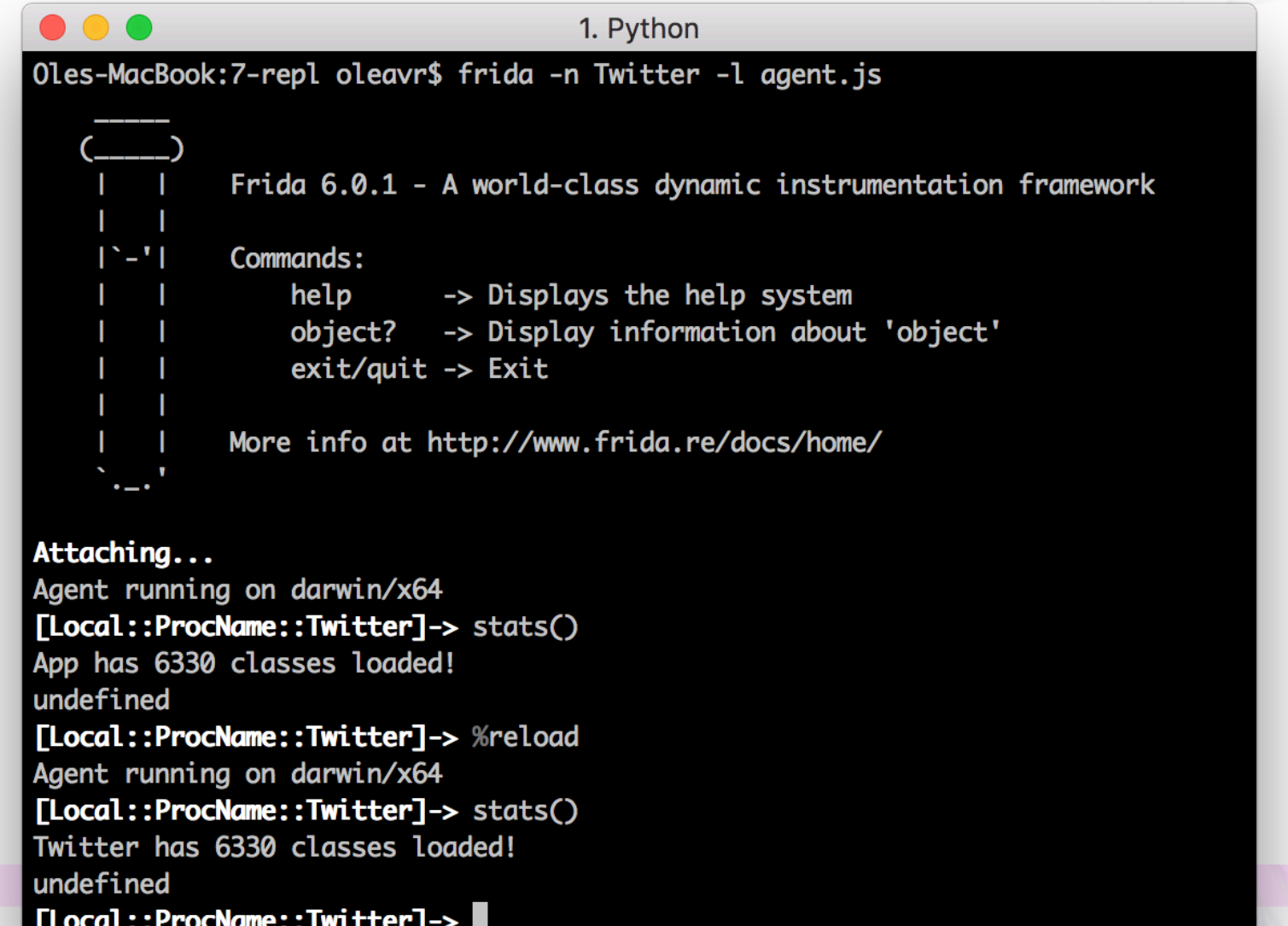
2. bash

```
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.cactosapp.aai
   -  BankID                no.bankid.client
   -  BensinPris            no.bitfactory.BensinPris.Release
   -  Big Day               com.whatisid.bigday
```

Speaking of apps, we also have a REPL:



The REPL is your best friend for prototyping scripts



Uninstall iOS app

```
'use strict';

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

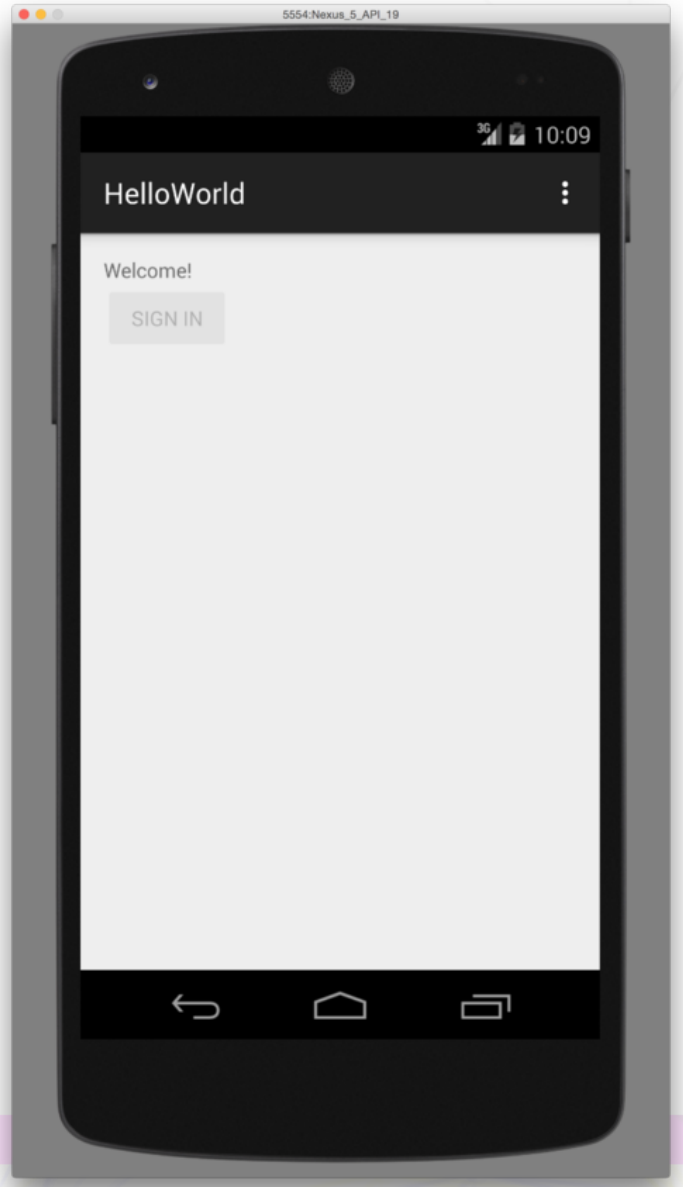
The low-level way:

```
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/ls”])*
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).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

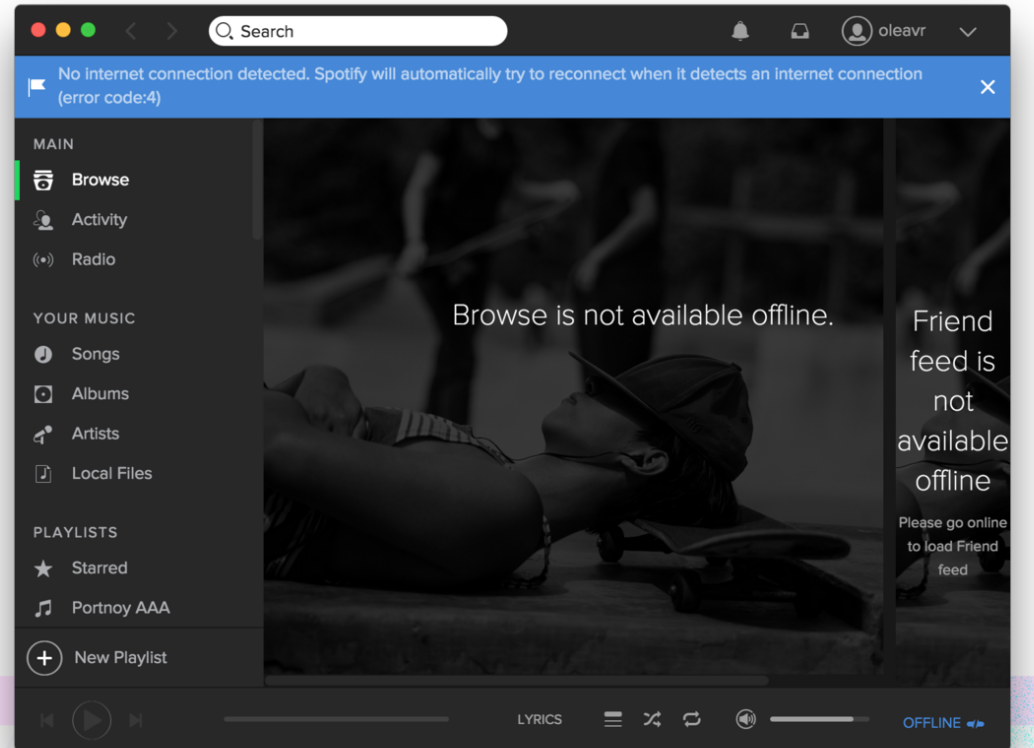
- Use Node.js bindings to *frida-load* your agent.js so you can:
  - Split your script into multiple files
  - Use Frida modules from the community
  - Reuse thousands of modules from npm
- Use ES6 features to write modern JavaScript – Frida support it
- REPL is great for prototyping with -l and %reload

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!
```

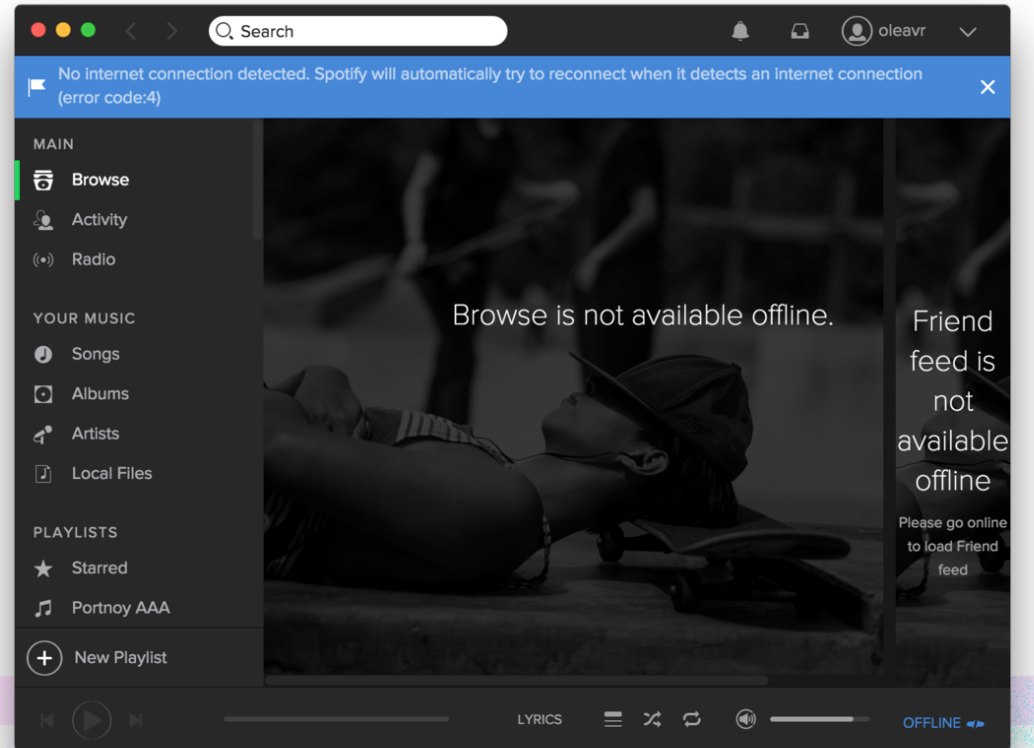


All calls between two recv() calls

```
'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!
```



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