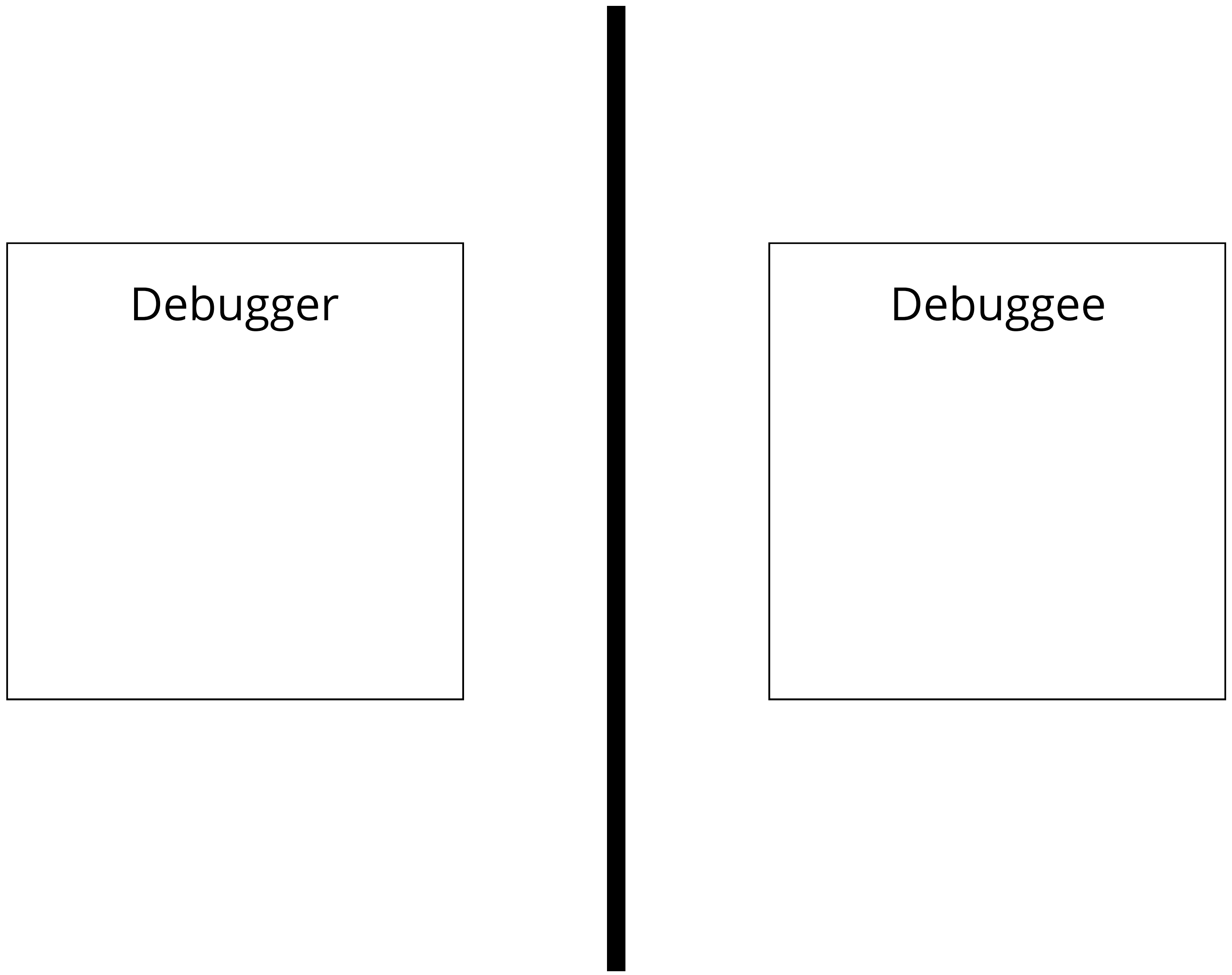
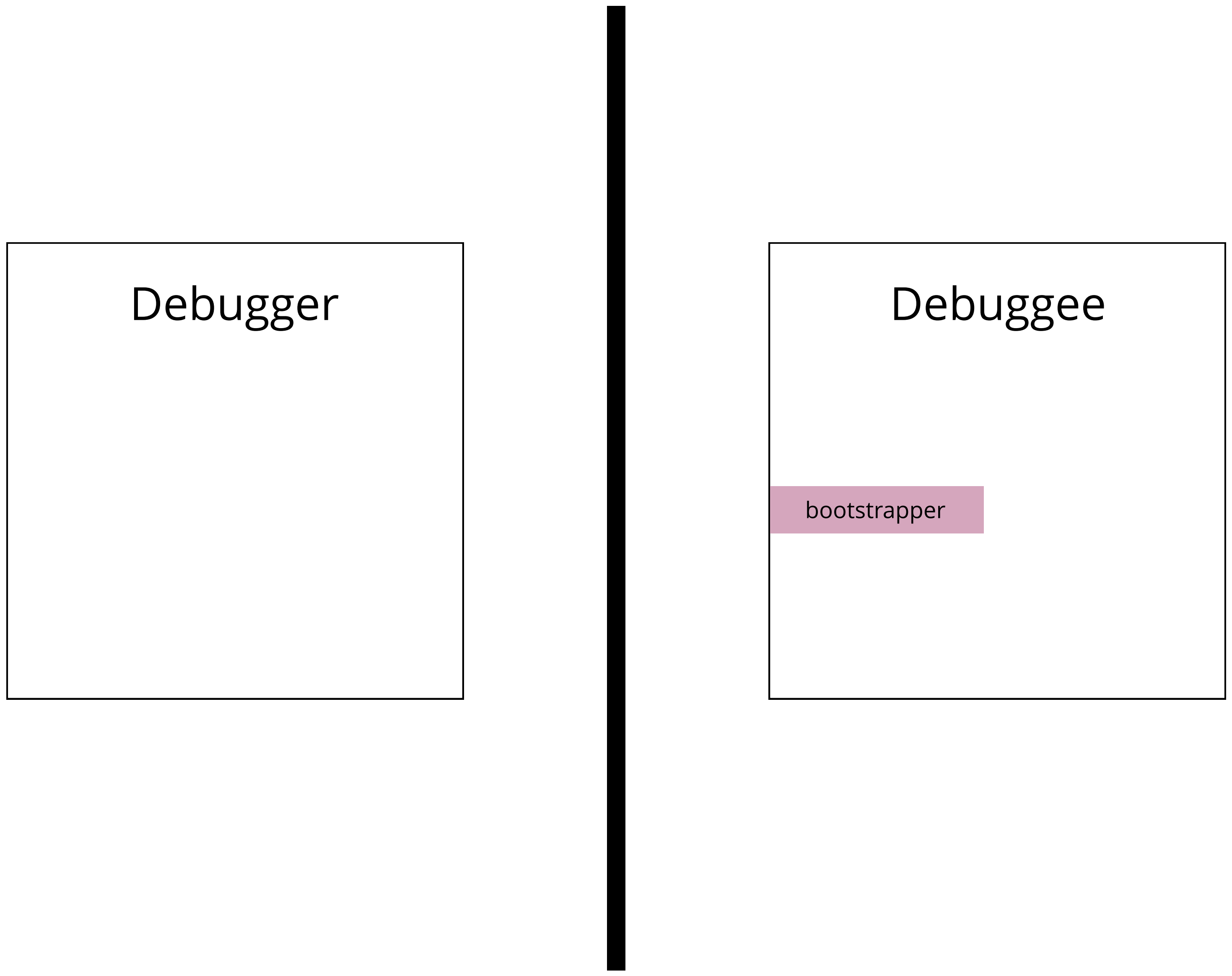
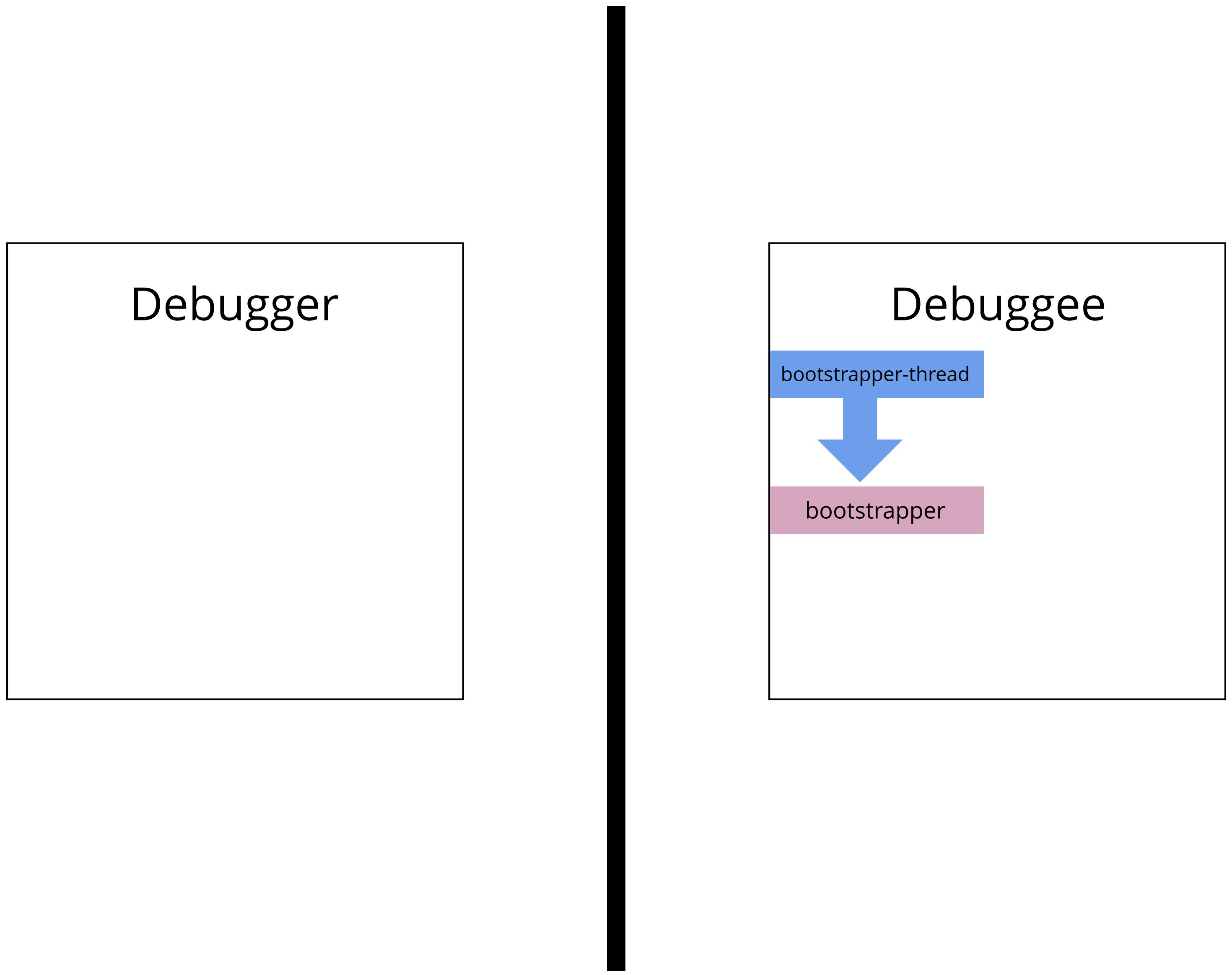
FRIDA

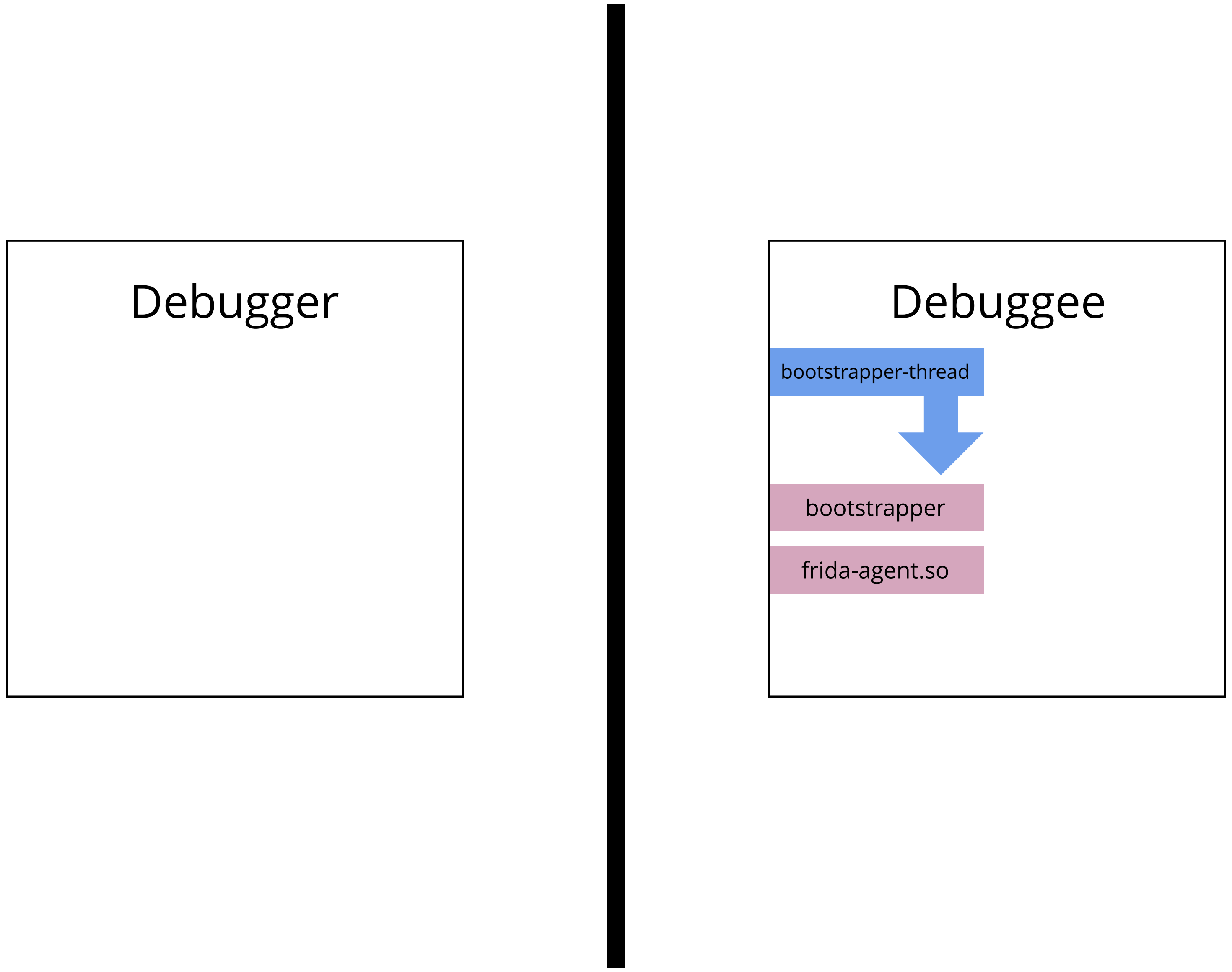
the ultimate static analysis on dynamic steroids

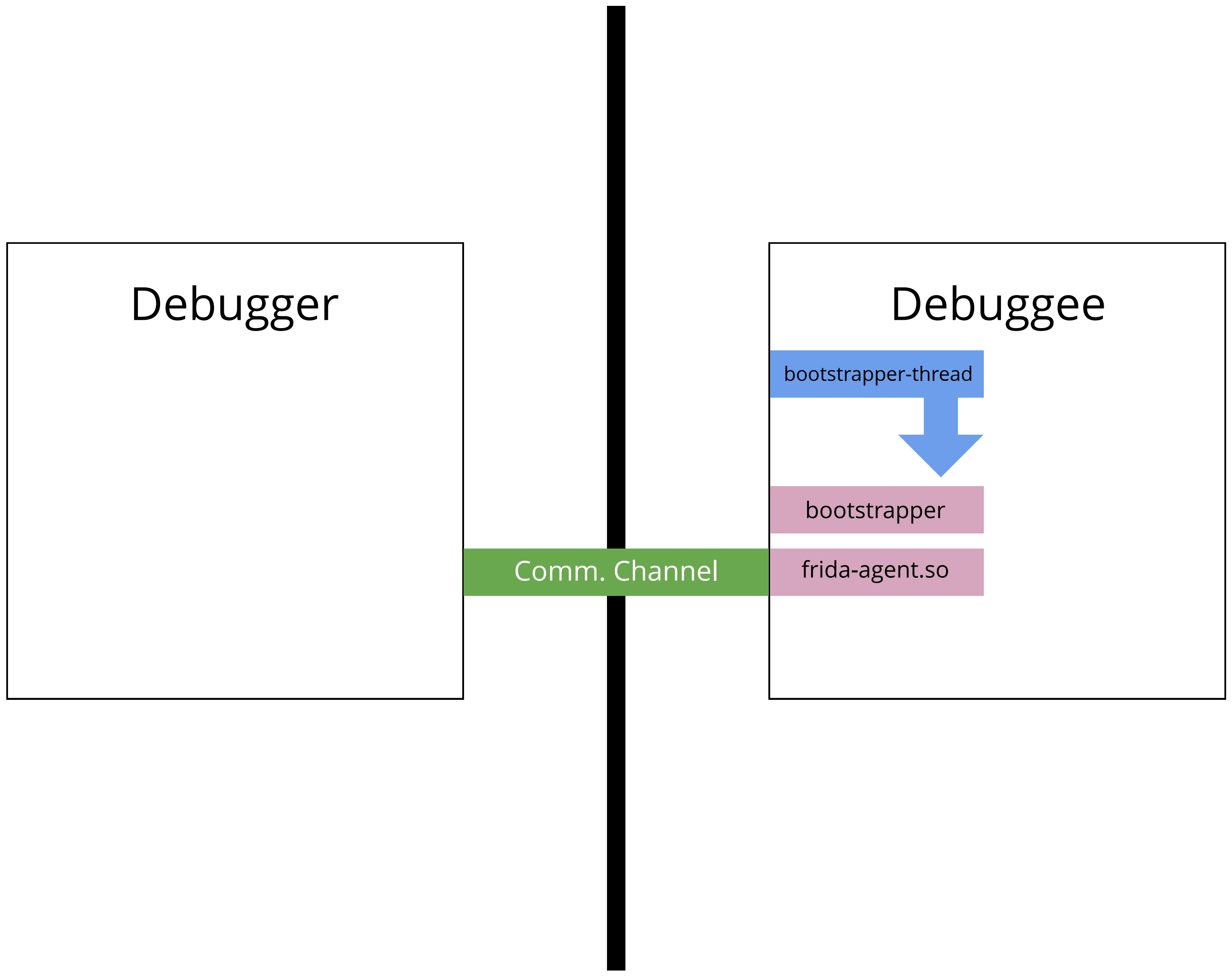
CON
2016

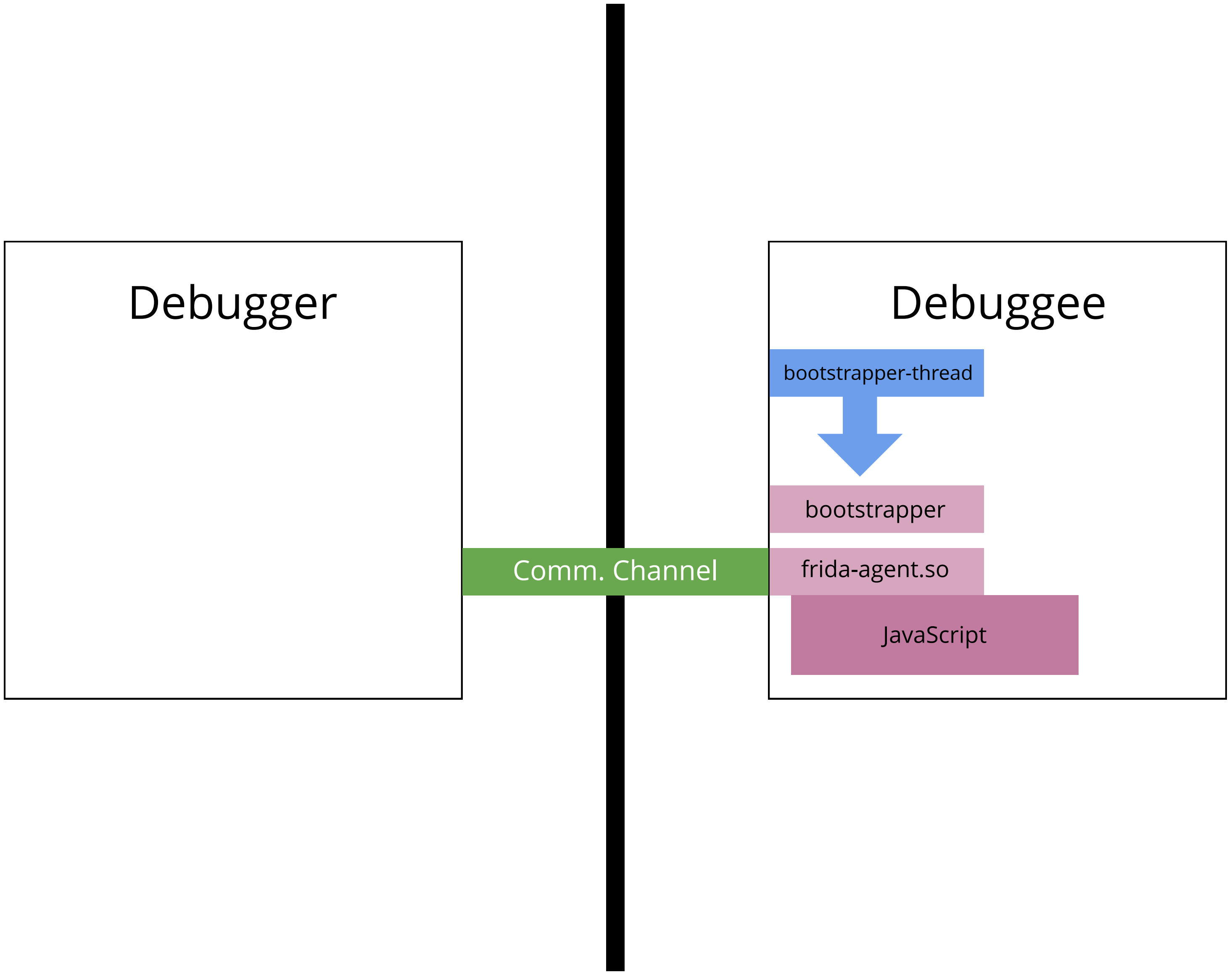

Debugger
Debuggee


Debugger
Debuggee
bootstrapper


Debugger
Debuggee
bootstrapper-thread
bootstrapper


Debugger
Debuggee
bootstrapper-thread
bootstrapper
frida-agent.so

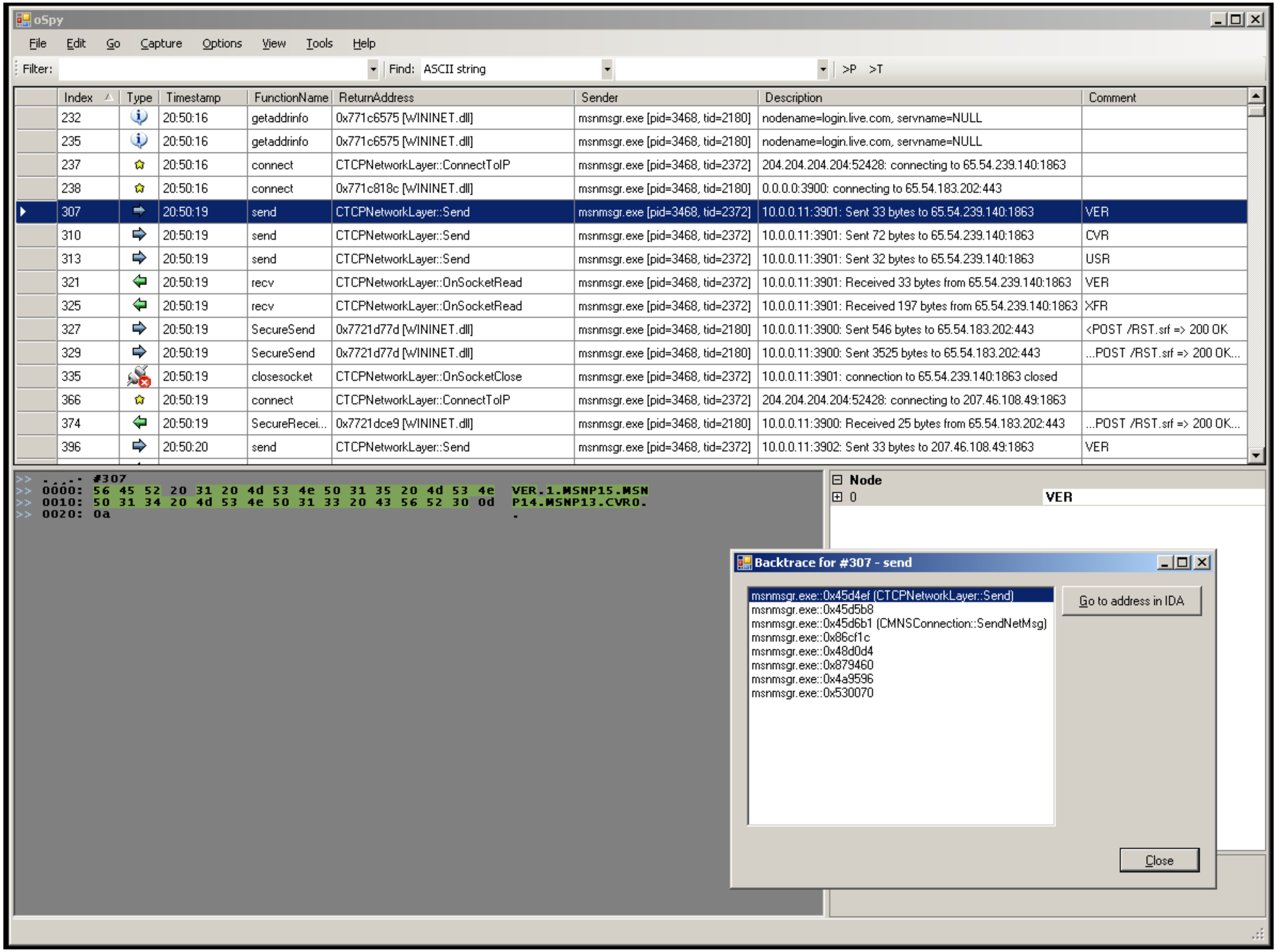

Debugger
Debuggee
bootstrapper-thread
bootstrapper
Comm. Channel
frida-agent.so


Debugger
Debuggee
bootstrapper-thread
bootstrapper
frida-agent.so
JavaScript
Comm. Channel

Demos

Motivation behind Frida

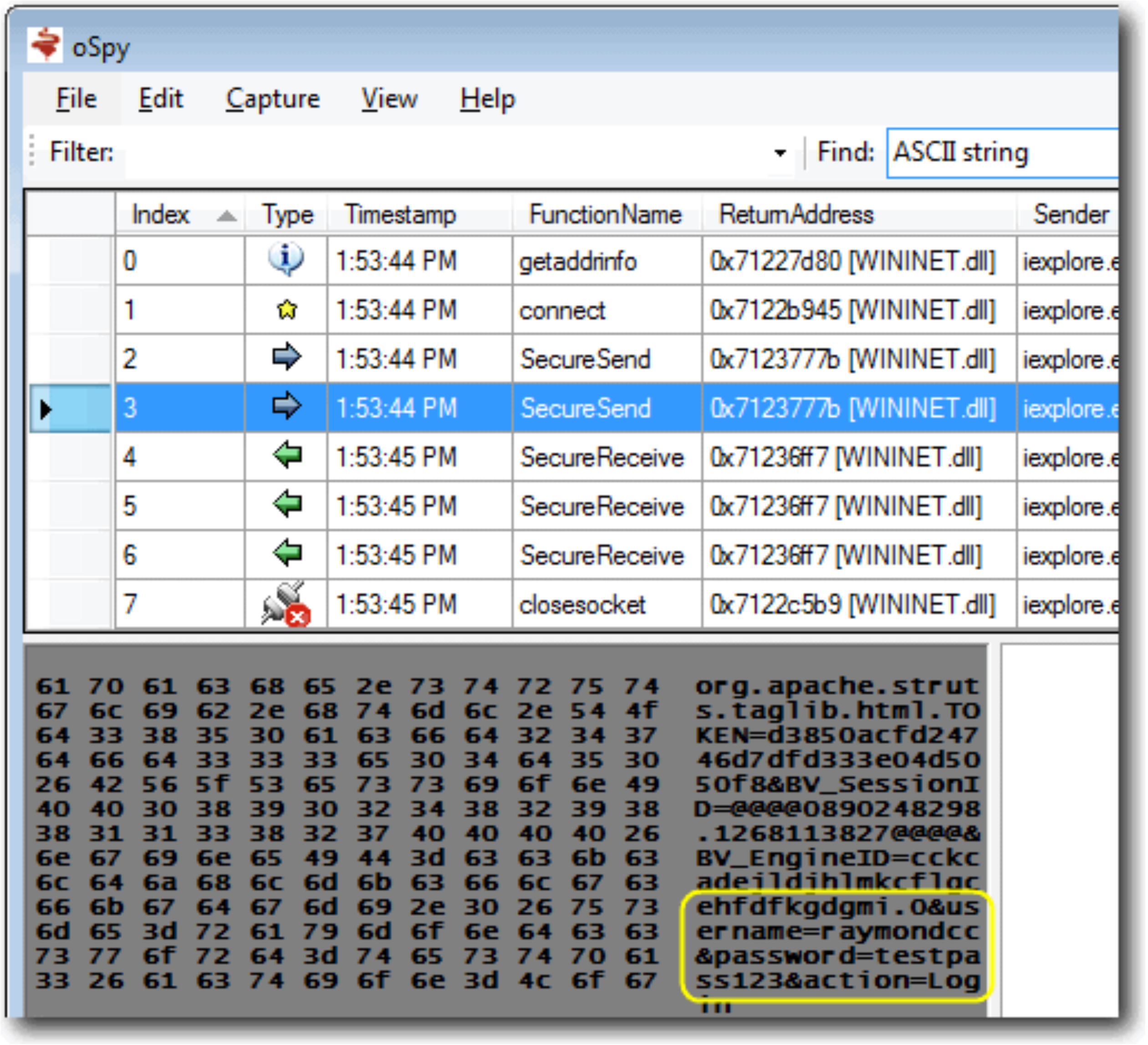
- Existing tools often not a good fit for the task at hand
- Creating a new tool usually takes too much effort
- Short feedback loop: reversing is an iterative process
- Use one toolkit for multi-platform instrumentation
- Future remake of oSpy (see below)



Index	Type	Timestamp	FunctionName	ReturnAddress	Sender	Description	Comment
232	ℹ	20:50:16	getaddrinfo	0x771c6575 [WININET.dll]	msnmsgr.exe [pid=3468, tid=2180]	nodename=login.live.com, servname=NULL	
235	ℹ	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

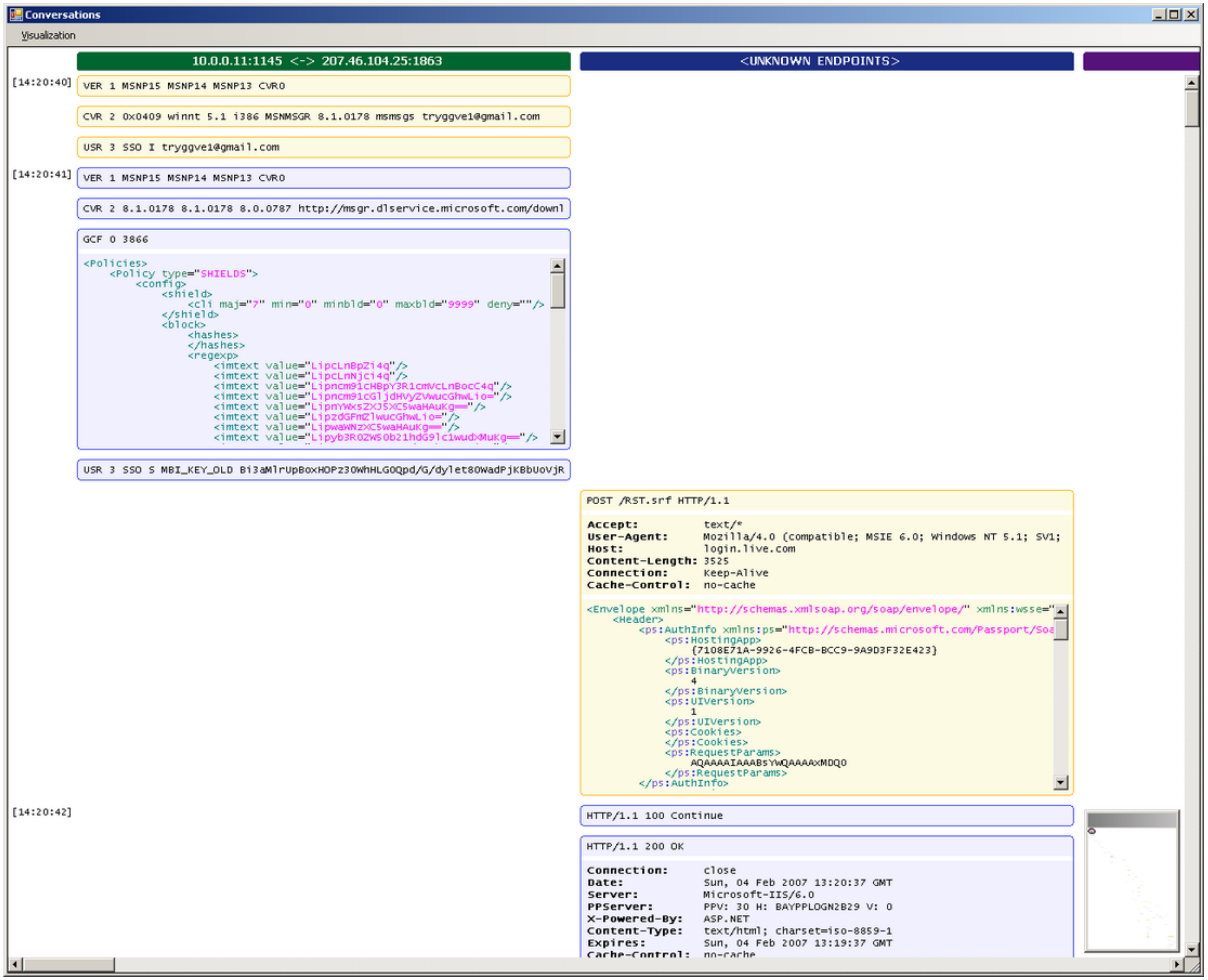
```
>>  .....  #307
>>  0000: 56 45 52 20 31 20 4d 53 4e 50 31 35 20 4d 53 4e   VER.1.MSNP15.MSN
>>  0010: 50 31 34 20 4d 53 4e 50 31 33 20 43 56 52 30 0d   P14.MSNP13.CVR0.
>>  0020: 0a                                                .
```

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



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   adeildjhlmkcflgc
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
                                      in
```



```
[14:20:40] VER 1 MSNP15 MSNP14 MSNP13 CVR0
           CVR 2 0x0409 winnt 5.1 i386 MSNMSGR 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.dlservice.microsoft.com/downl
           GCF 0 3866
```

```
<Policies>
    <Policy type="SHIELDS">
        <config>
            <shield>
                <cli maj="7" min="0" minbld="0" maxbld="9999" deny=""/>
            </shield>
            <block>
                <hashes>
                </hashes>
                <regexp>
                    <imtext value="LipcLnBpZi4q"/>
                    <imtext value="LipcLnNjci4q"/>
                    <imtext value="Lipncm91cHBpY3R1cmVcLnBocC4q"/>
                    <imtext value="Lipncm91cGljdHVyZVwucGhwLio="/>
                    <imtext value="LipnYWxsZXJ5XC5waHAuKg=="/>
                    <imtext value="LipzdGFmZ1wucGhwLio="/>
                    <imtext value="LipwaWNzXC5waHAuKg=="/>
                    <imtext value="Lipyb3R0ZW50b21hdG91c1wudXMuKg=="/>
```

```
           USR 3 SSO S MBI_KEY_OLD Bi3aMlrUpBoxHOPz30WhHLG0Qpd/G/dylet80WadPjKBbUoVjR
```

```
POST /RST.srf HTTP/1.1

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

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsse="
    <Header>
        <ps:AuthInfo xmlns:ps="http://schemas.microsoft.com/Passport/Soa
            <ps:HostingApp>
                {7108E71A-9926-4FCB-BCC9-9A9D3F32E423}
            </ps:HostingApp>
            <ps:BinaryVersion>
                4
            </ps:BinaryVersion>
            <ps:UIVersion>
                1
            </ps:UIVersion>
            <ps:Cookies>
            </ps:Cookies>
            <ps:RequestParams>
                AQAAAAIAAABsYWQAAAAxMDQ0
            </ps:RequestParams>
        </ps:AuthInfo>
```

```
[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

Why would you need Frida?

- For reverse-engineering
- For programmable debugging
- For dynamic instrumentation
- But ultimately: To enable rapid development of new tools for the task at hand

Architecture

- Highly modular and decoupled
- Instrumentation core written in C (frida-gum)
  - C++ and JavaScript language bindings
- Easy to use high-level API: "run JS in that process"
  - Packages instrumentation core in a library
  - Injects that using a per OS injector component
  - Communicates with it over a per OS transport
  - Bindings: C, Node.js, Python, .NET, Swift, Qt
- Philosophy: only bare metal building blocks, community provides use-case-specific modules in npm, e.g. frida-fs, frida-screenshot, frida-uikit, frida-uiwebview, etc.

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.

Hooking f() from Node.js

```
'use strict';

const co = require('co');
const frida = require('frida');
const fs = require('mz/fs');

let session, script;
co(function *() {
  session = yield frida.attach('hello');
  const source = yield fs.readFile(
      require.resolve('./agent.js'), 'utf-8')
  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

Hooking f() from Python

```python
import frida
import sys

session = frida.attach("hello")
script = session.create_script("""
Interceptor.attach(ptr("0x106a81ec0"),
    onEnter: function(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}
…
```

There are also language-bindings for QML, .NET, etc. The API is the same except

for local conventions like create_script() vs createScript().

We will stick to the Node.js bindings for the remainder of this presentation.

Modifying function arguments

```
'use strict';

const co = require('co');
const frida = require('frida');
const fs = require('mz/fs');

let session, script;
co(function *() {
  session = yield frida.attach('hello');
  const source = yield fs.readFile(
      require.resolve('./agent.js'), 'utf-8')
  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

Calling functions

```
'use strict';

const co = require('co');
const frida = require('frida');
const fs = require('mz/fs');

let session, script;
co(function *() {
  session = yield frida.attach('hello');
  const source = yield fs.readFile(
      require.resolve('./agent.js'), 'utf-8')
  script = yield session.createScript(source)
  yield script.load();
});
```

```
$ node app.js
```

```
Number: 1879
Number: 1911
Number: 1911
Number: 1911
Number: 1880
…
```

```
'use strict';

const f = new NativeFunction(
    ptr('0x106a81ec0'), 'void', ['int']);
f(1911);
f(1911);
f(1911);
```

Address of f() goes here

Sending messages

```
'use strict';

const co = require('co');
const frida = require('frida');
const fs = require('mz/fs');

let session, script;
co(function *() {
  session = yield frida.attach('hello');
  const source = yield fs.readFile(
      require.resolve('./agent.js'), 'utf-8')
  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 Ob
  fileName: 'agent.js',
  lineNumber: 10,
  columnNumber: 1 }
```

Receiving messages

```
'use strict';

const co = require('co');
const frida = require('frida');
const fs = require('mz/fs');

let session, script;
co(function *() {
  session = yield frida.attach('hello');
  const source = yield fs.readFile(
      require.resolve('./agent.js'), 'utf-8')
  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);
```

Blocking receives

```
'use strict';

const co = require('co');
const frida = require('frida');
const fs = require('mz/fs');

let session, script;
co(function *() {
  session = yield frida.attach('hello');
  const source = yield fs.readFile(
      require.resolve('./agent.js'), 'utf-8');
  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

Launch and spy on iOS app

```javascript
'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);
```

```javascript
'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.dyli
```

io plugin code walkthrough

Questions?

Twitter: @oleavr

Thanks!

Code is at:

https://github.com/nowsecure/r2frida

Soon also available in r2pm.