Digital Interruption

Prototyping and reverse engineering with frida

+44 (0)161-820-3056

www.digitalinterruption.com

jahmel@digitalinterruption.com

DIGITAL INTERRUPTION

THIS WORKSHOP

- Introduction to rapid reverse engineering with Frida
- Practical exercises (limit the theory)
- View the code! No need to read assembly
- Linux/android
- !exploitation
- 2 hours (realistically less)

PREREQUISITES

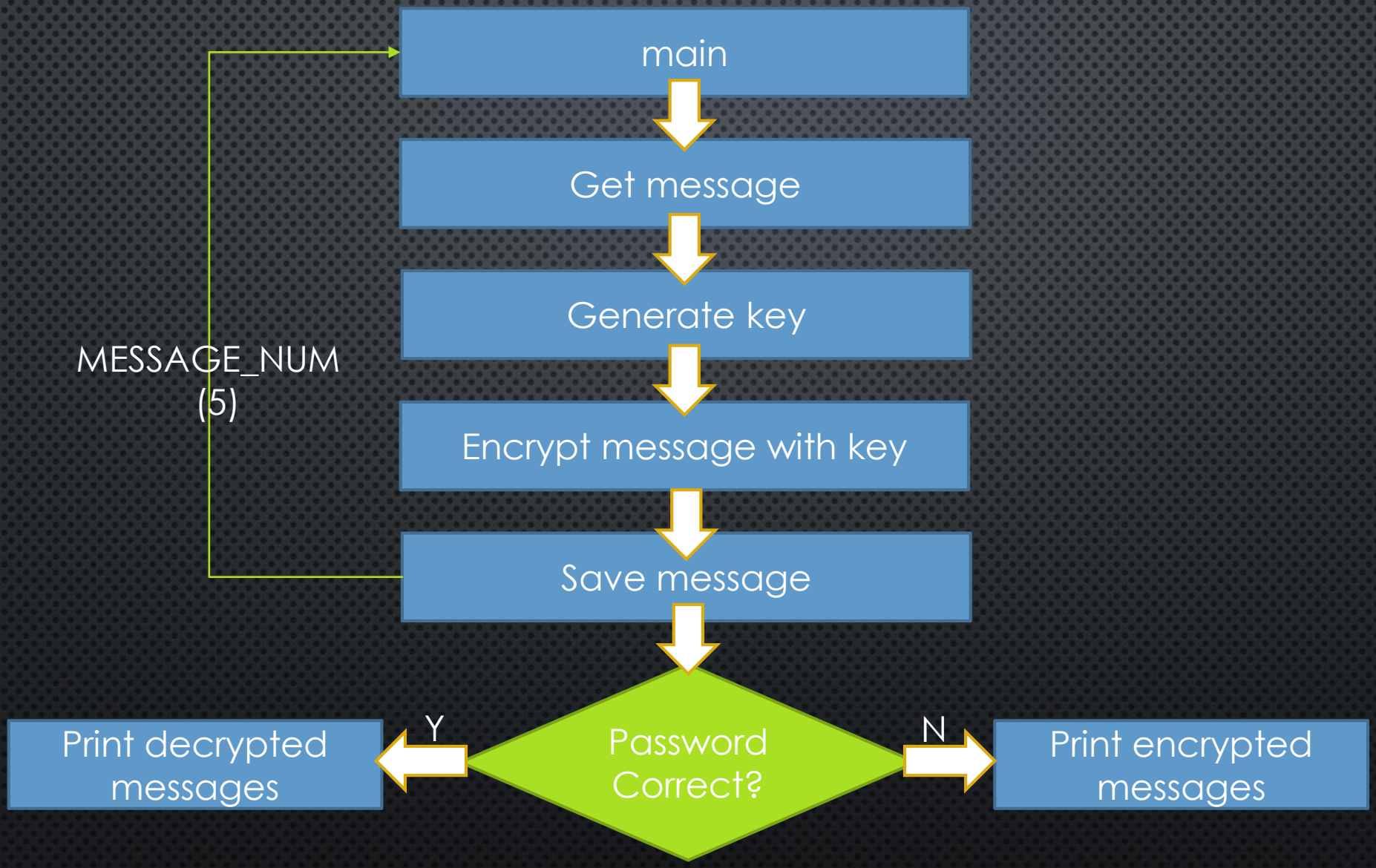
- Laptop running Linux
- C Compiler (or trust my binary)
- Frida
- Read simple C code
- Code simple JavaScript

WHOAMI

- Jahmel Harris
- Pen Tester/Security Researcher at Digital Interruption Mobile | Radio | Reverse Engineering

- @JayHarris_Sec
- @mcrgreyhats
- @DI_Security

Target Application


main
Get message
Generate key
Encrypt message with key
Save message
MESSAGE_NUM
(5)
Password
Correct?
Y
N
Print decrypted
messages
Print encrypted
messages

@JayHarris_Sec

Terminal

File Edit View Search Terminal Help

```
~/./exercise
Enter Message:
Message 1
Enter Message:
Message 2
Enter Message:
Message 3
Enter Message:
Message 4
Enter Message:
Message 5
Enter Password
password
message 0:Message 1
message 1:Message 2
message 2:Message 3
message 3:Message 4
message 4:Message 5
~/
```

@JayHarris_Sec

Terminal

File Edit View Search Terminal Help

```
~/./exercise
Enter Message:
message 1
Enter Message:
message 2
Enter Message:
message 3
Enter Message:
message 4
Enter Message:
message 5
Enter Password
IDontKnowThePassword
message 0:"n◆_qe◆g
message 1:6◆◆g>◆x◆
message 2:H◆v◆M◆
message 3:◆◆◆◆◆◆◆◆◆◆`
message 4:Hx◆P◆◆o◆◆
~/
```

Can we bypass the need for a password?

EXERCISE 0

- CONFIGURE ENVIRONMENT FOR WORKSHOP
  - pip install frida
  - DOWNLOAD EXERCISE.C (http://bit.ly/2rlZeuQ)
  - $ make exercise
  - sudo sysctl kernel.yama.ptrace.scope=0

WHAT IS REVERSE ENGINEERING?

- Reproducing something based on extracted knowledge
- Understanding the behaviour of a binary
- Lengthy process that requires skill

If it's so hard, why do it?

- Source code recovery
- Interoperability
- Fun!
- Vulnerability research

APPLICATION HOOKING

- Invaluable tool in dynamic analysis
- View internal state
- Add logging
- Change application logic

APPLICATION HOOKING

```
read(0,buffer,255);
```

```
read(int,char*,int){
    //code
}
```

```
message = encrypt(buffer,key,255);
```

```
encrypt(char*,char*,int){
    //code
}
```

```
sendMessage(message,255);
```

APPLICATION HOOKING

```
read(0,buffer,255);
```

```
read(int,char*,int){
    buffer="our string";
}
```

```
message = encrypt(buffer,key,255);
```

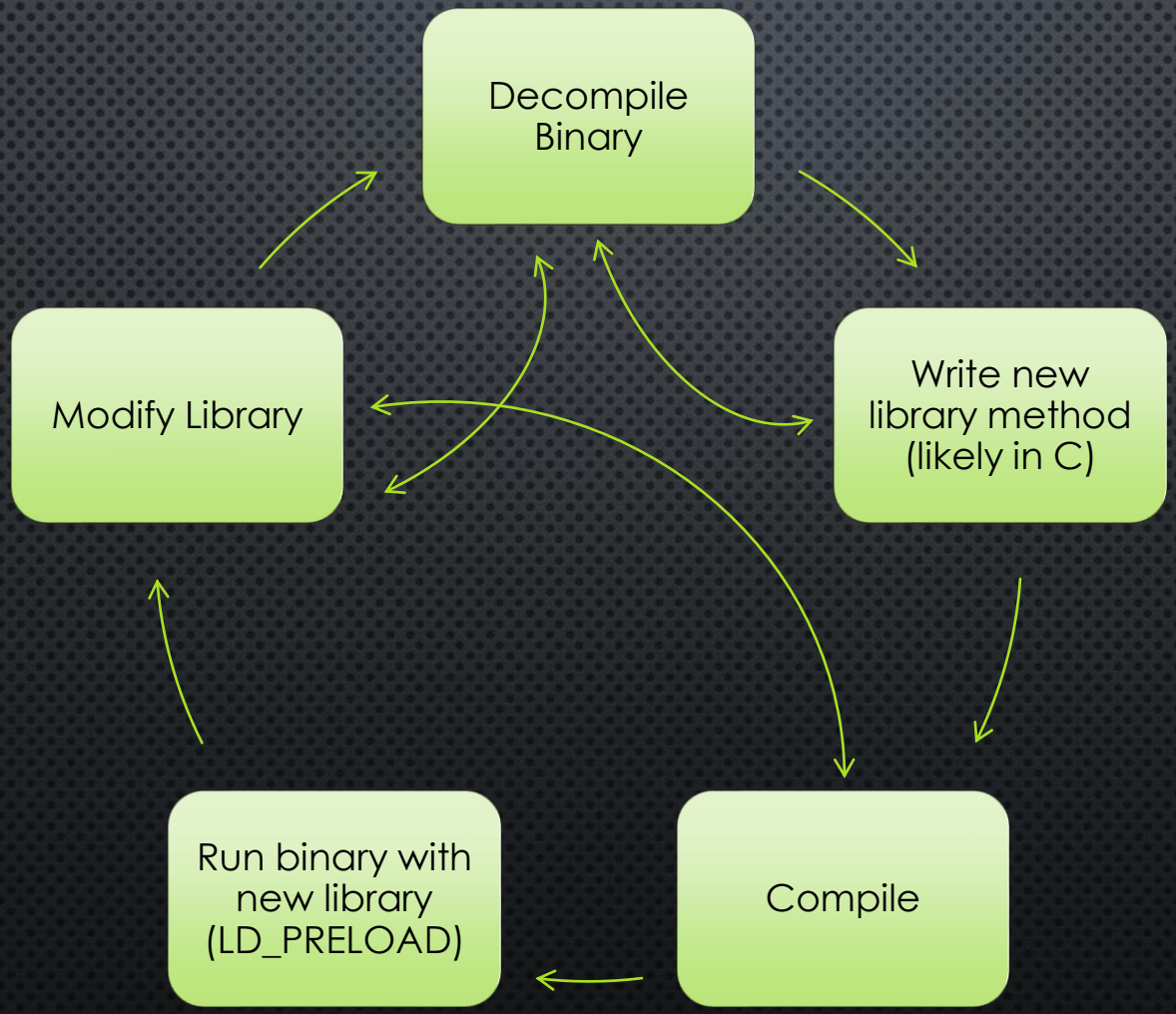
```
encrypt(char*,char*,int){
    log(args);
    encrypt(args[0],"000000",args[2]);
}
```

```
sendMessage(message,255);
```

```
encrypt(char*,char*,int){
    //code
}
```

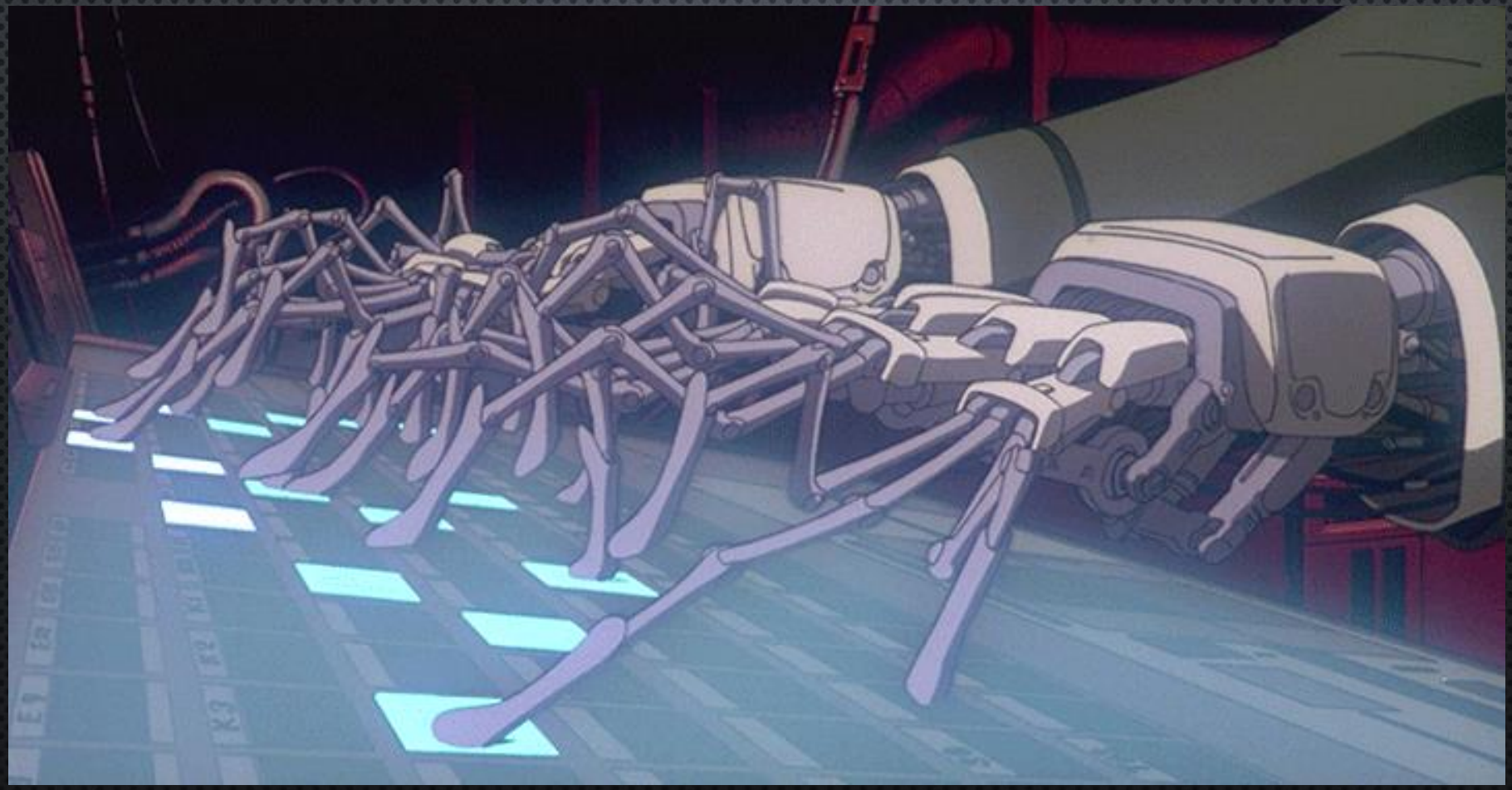
BEFORE FRIDA

BEFORE FRIDA



DEMO

AFTER FRIDA

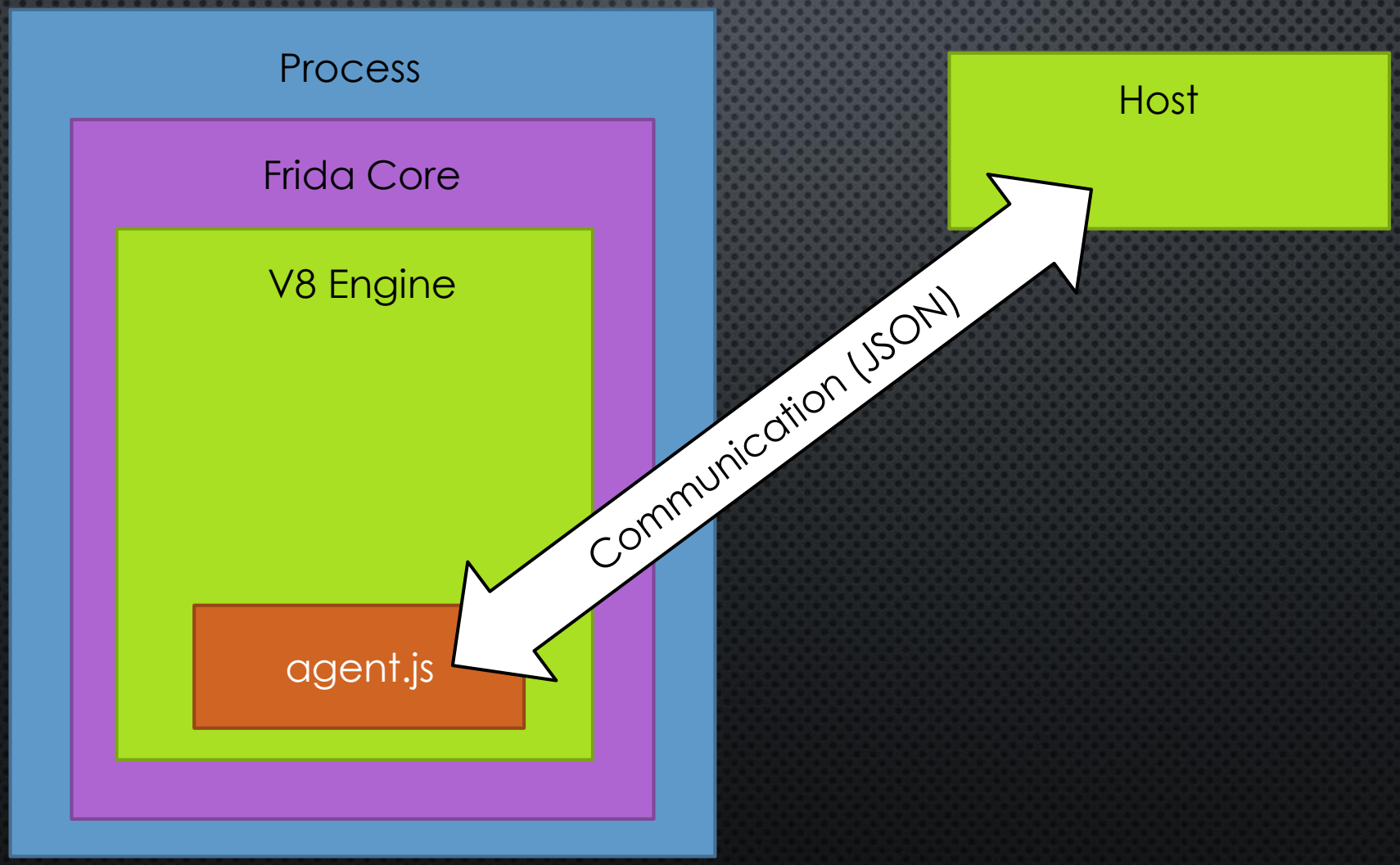


DEMO

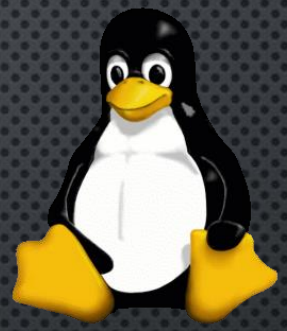
FRIDA

- Framework for instrumentation
- Injecting JavaScript into application (whaaaa!)
- Most importantly – a framework for building tools

FRIDA


Process
Frida Core
V8 Engine
agent.js
Host
Communication (JSON)

FRIDA



EXERCISE 1

- Inject JavaScript into process
- Explore Frida API
  - What is the current threadID?
  - What Modules are loaded into the process?
  - What are the addresses of the libc functions?

FRIDA

- FRIDA
- FRIDA-PS
- FRIDA-TRACE

FRIDA-TRACE

- Written using Frida (and installed with frida)
- Creates JavaScript file for hooked functions (by name)
- Can use wildcards (Frida-trace –I "*" process)

FRIDA-TRACE

```
char* encryptedMessage = encryptMessage(message,key,255);
```

```
onEnter: function(log,args,state){
    log("encryptMessage");
    log(Memory.readUtf8String(args[1]));
}
```

```
onLeave: function(log,retval,state)
{
     retval.replace(0x00);
}
```

FRIDA-TRACE

```
char* encryptedMessage = encryptMessage(message,key,255);
```

```
onEnter: function(log,args,state){
    log("encryptMessage");
    log(Memory.readUtf8String(args[1]));
}
```

```
onLeave: function(log,retval,state)
{
    retval.replace(0x00);
}
```

FRIDA-TRACE

```
char* encryptedMessage = encryptMessage(message, key, 255);
```

```
onEnter: function(log,args,state){
    log("encryptMessage");
    log(Memory.readUtf8String(args[1]));
}
```

```
onLeave: function(log,retval,state)
{
    retval.replace(0x00);
}
```

FRIDA-TRACE

```
char* encryptedMessage = encryptMessage(message,key,255);
```

```
onEnter: function(log,args,state){
    log("encryptMessage");
    log(Memory.readUtf8String(args[1]);
}
```

```
onLeave: function(log,retval,state)
{
    retval.replace(0x00);
}
```

FRIDA-TRACE

```
char buffer[255];
encryptMessage(message,key,buffer,255);
```

```
onEnter: function(log,args,state){
  log(Memory.readUtf8String(args[2]); //garbage
  this.buf = args[2];
}
```

```
onLeave: function(log,retval,state)
{
  log(Memory.readUtf8String(this.buf));
}
```

DEMO

EXERCISE 2

- Hook the process to log "read"
  - What about the arguments?
  - Pointers?
- Modify "rand()" to affect the encrypted data

FRIDA-TRACE

- Require memory address
  - Easy with imported functions + Frida-trace
- What about internal functions?
  - Hint: objdump

FRIDA-TRACE

```
$ objdump -d exercise | grep -i "functionName"
 af3:   e8 fd 01 00 00                callq  cf5 <functionName>
0000000000000cf5 <functionName>:
```

FRIDA-TRACE

```
$ objdump -d exercise | grep -i "functionName"
 af3:   e8 fd 01 00 00                callq  cf5 <functionName>
0000000000000cf5 <functionName>:
```

WE NOW KNOW THE OFFSET

FRIDA-TRACE

```
$ objdump -d exercise | grep -i "functionName"
 af3:   e8 fd 01 00 00                callq  cf5 <functionName>
0000000000000cf5 <functionName>:
```

```
$ frida-trace –a exercise!0x0f5
```

```
[Local::ProcName::printRandNumber]-> Process.enumerateModulesSync()
```

EXERCISE 3

- Change "encryptString()" to print the key
- How can the "checkPassword()" function be bypassed?

SCRIPTING FRIDA

- BINDINGS MAKE FRIDA SCRIPTABLE!
- BINDINGS FOR NODE.JS, PYTHON, .NET, QML ETC

PYTHON TEMPLATE

```
import frida
import sys
def on_message(message, data):
    print message[‘payload’]

jscode = """
send(“hello world”);
"""

session = frida.attach(“process")
script = session.create_script(jscode)
script.on('message', on_message)
script.load()
sys.stdin.read()
```

DEMO

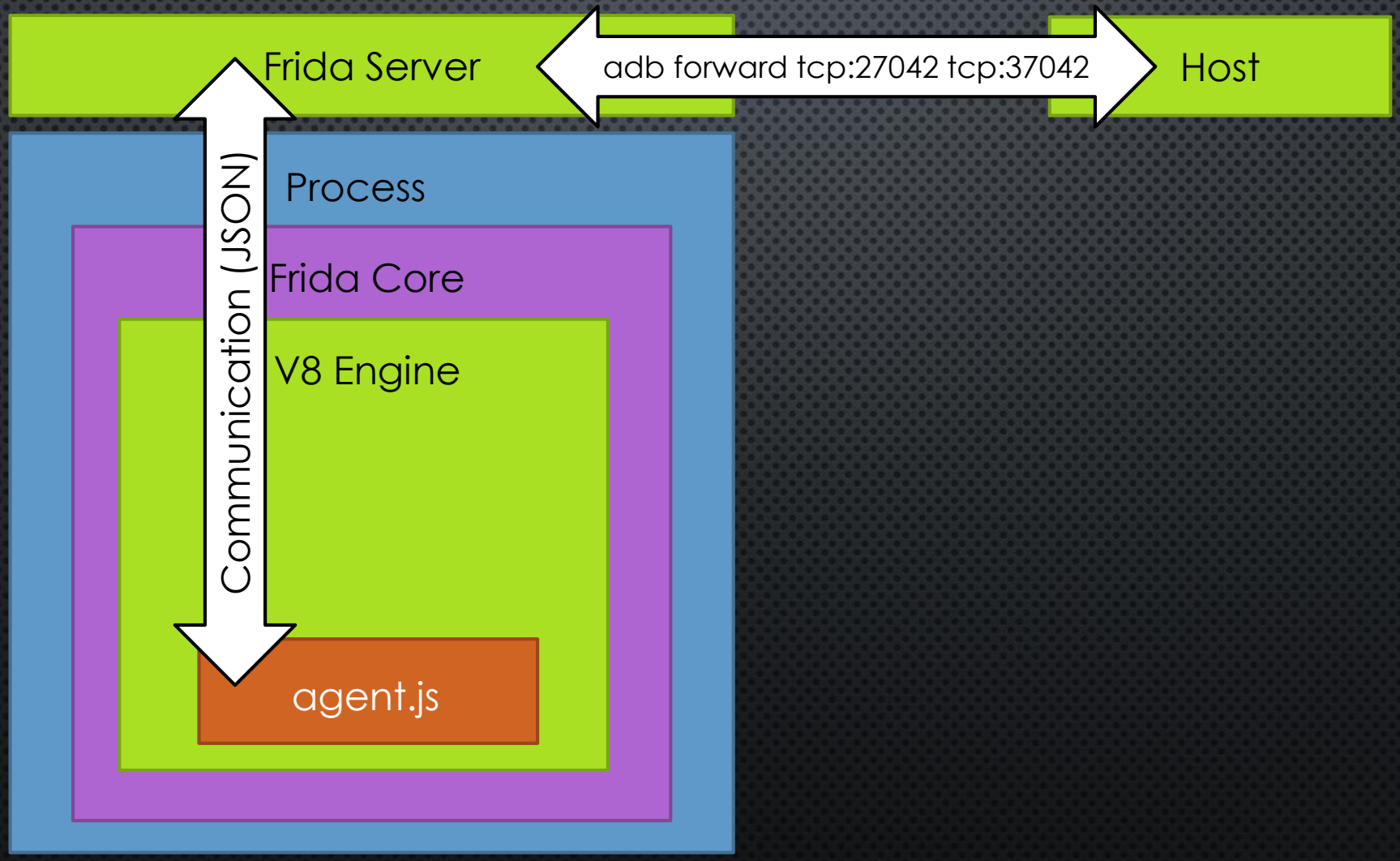
EXERCISE 4

- Create Python script to run decryptMessage()
- Create Python script to replace PrintAllEncryptedMessages with decryptAllMessages()
  - Hint: NativeFunction()

FRIDA AND ANDROID

- ROOTED AND NON ROOTED

FRIDA AND ANDROID



DEMO – BYPASSING APP SECURITY

How do we protect against this?

Should we protect against this?