@jayHarris_Sec

Digital
Interruption

Prototyping and reverse engineering with frida

+44 (0)161-820-3056

www.digitalinterruption.com

jahmel@digitalinterruption.com

DIGITAL INTERRUPTION

This Talk

- Introduction to Frida
- Using Frida for reverse engineering
- Practical demonstrations
- Linux/android
- !exploitation

whoami

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

- @JayHarris_Sec
- @mcrgreyhats
- @DI_Security

Intro to frida

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?

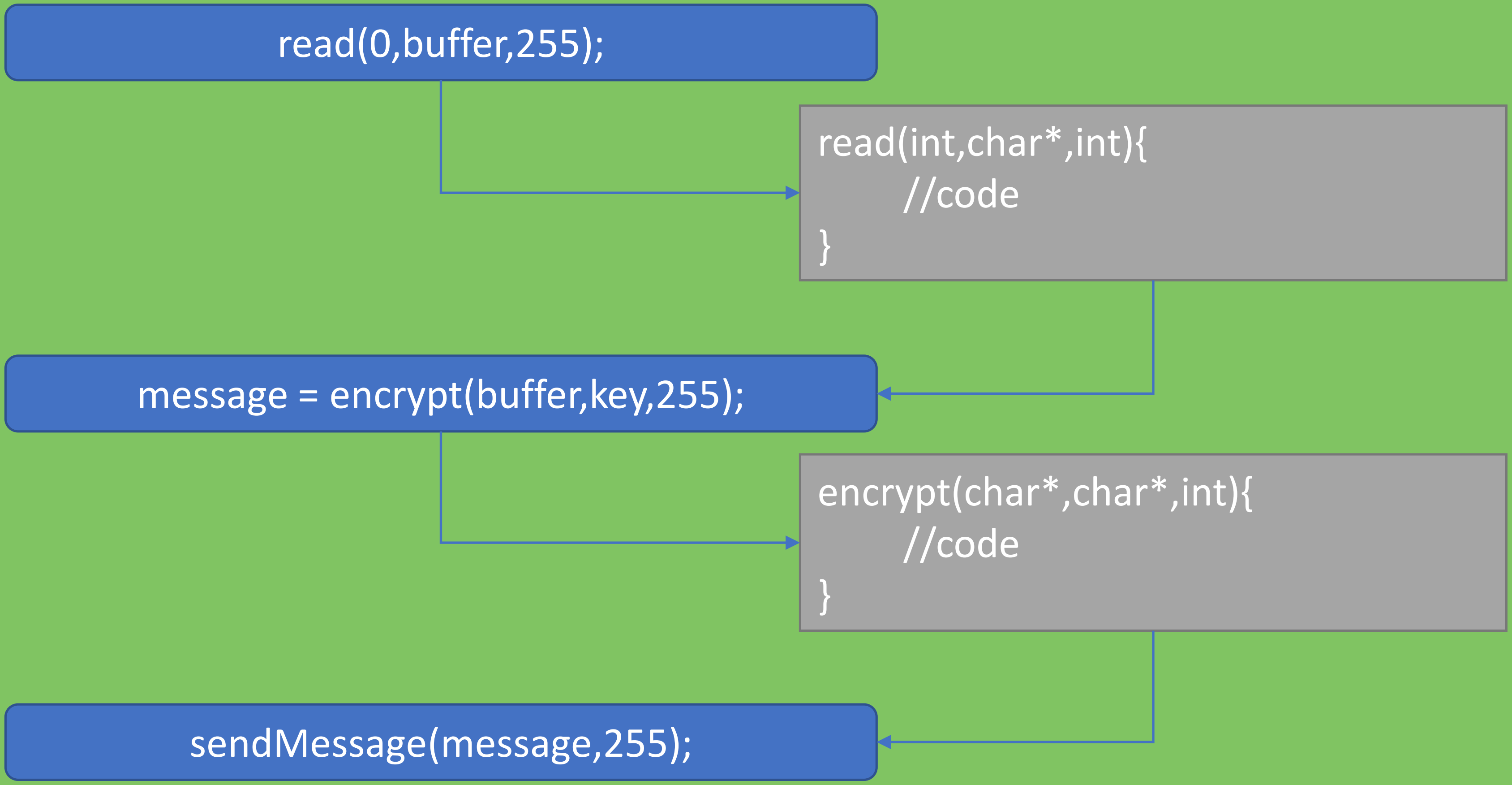
- Source code recovery
- Interoperability
- Fun!
- Vulnerability research

Application Hooking

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

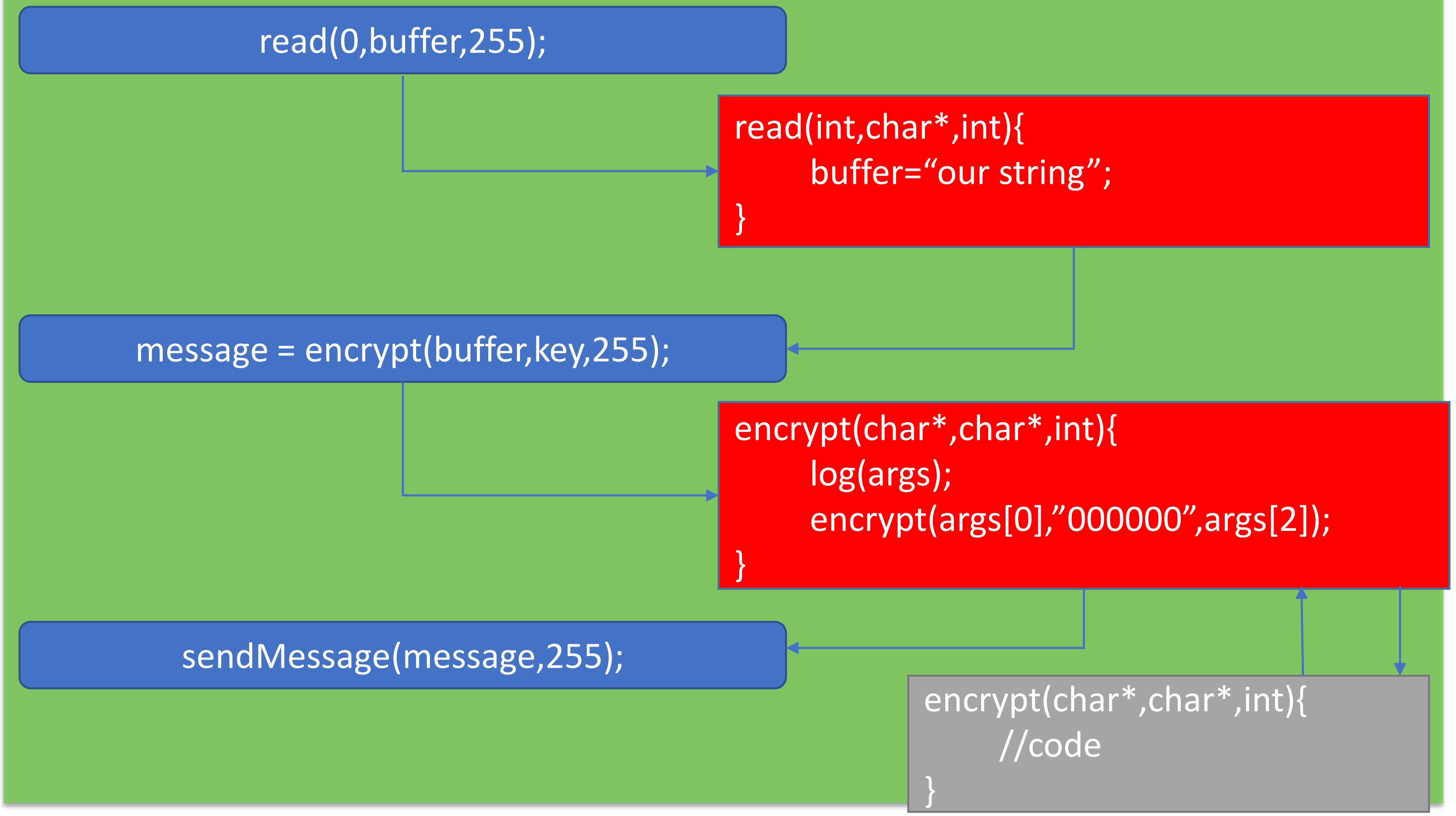
@jayHarris_Sec

Application Hooking


read(0,buffer,255);
read(int,char*,int){
//code
}
message = encrypt(buffer,key,255);
encrypt(char*,char*,int){
//code
}
sendMessage(message,255);

@jayHarris_Sec

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]);
}
```

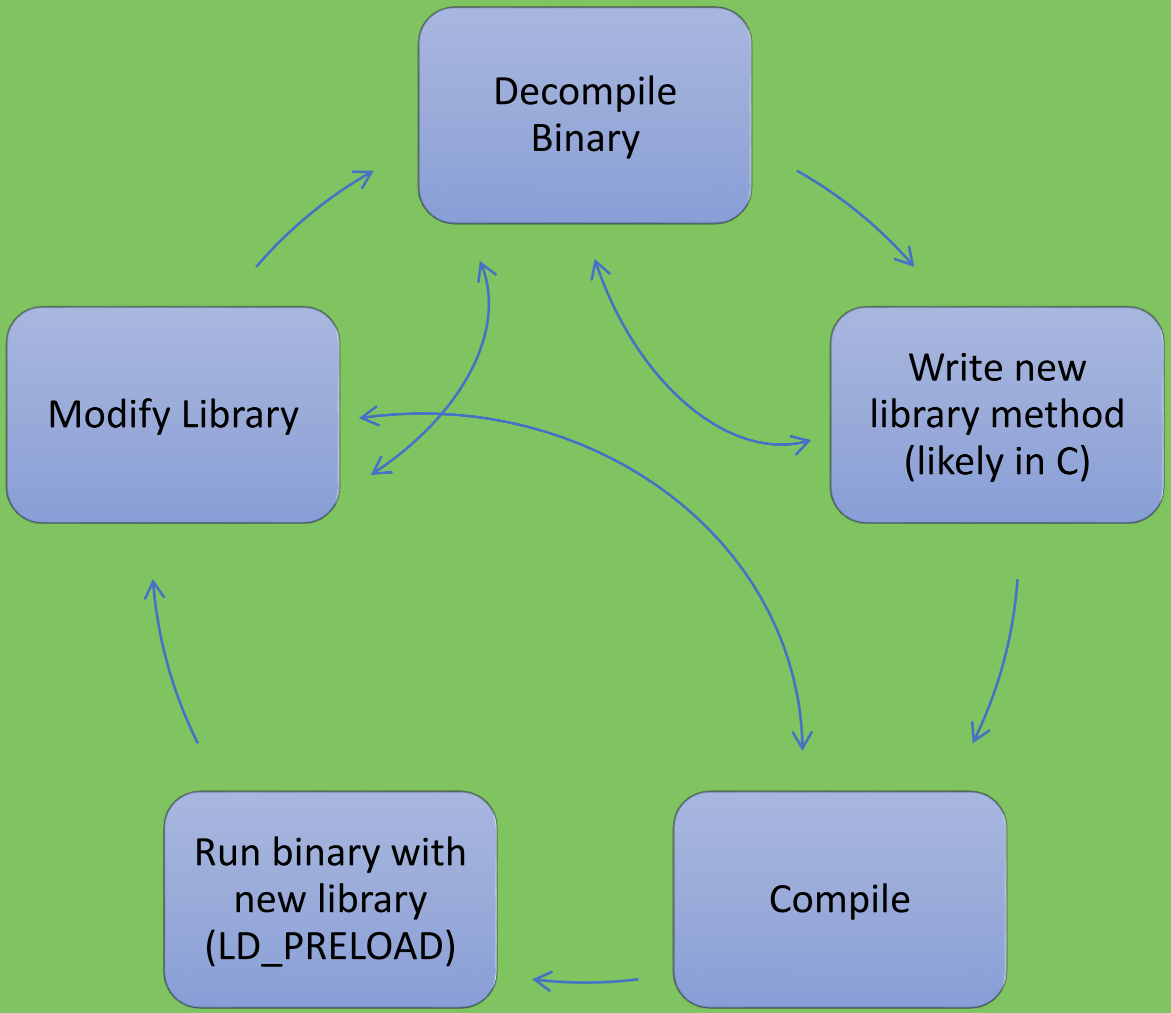
```
sendMessage(message,255);
```

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

Before Frida

@jayHarris_Sec

Before Frida


Decompile Binary
Write new library method (likely in C)
Compile
Run binary with new library (LD_PRELOAD)
Modify Library

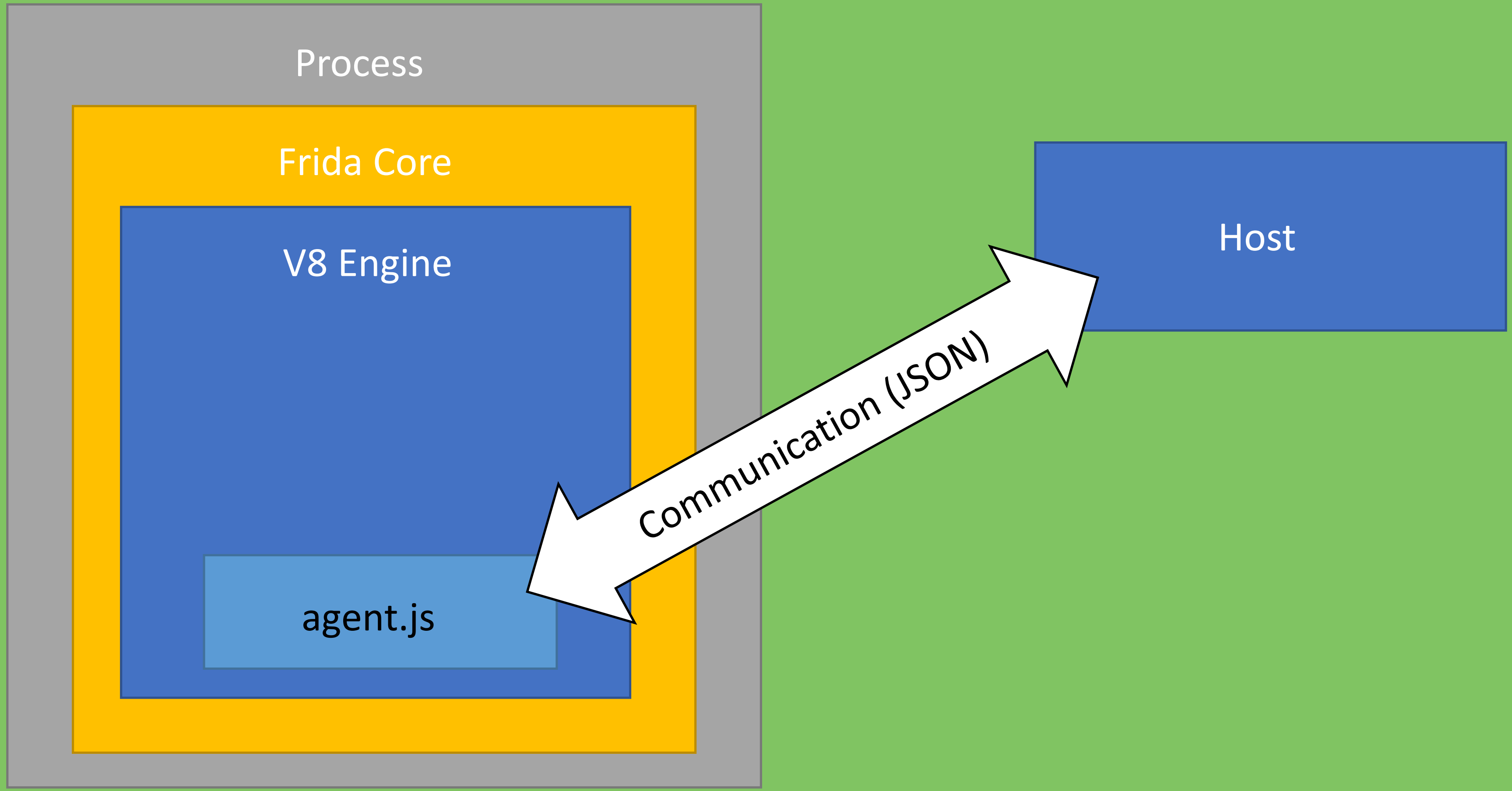
After Frida

Frida

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

@jayHarris_Sec

Frida



Frida

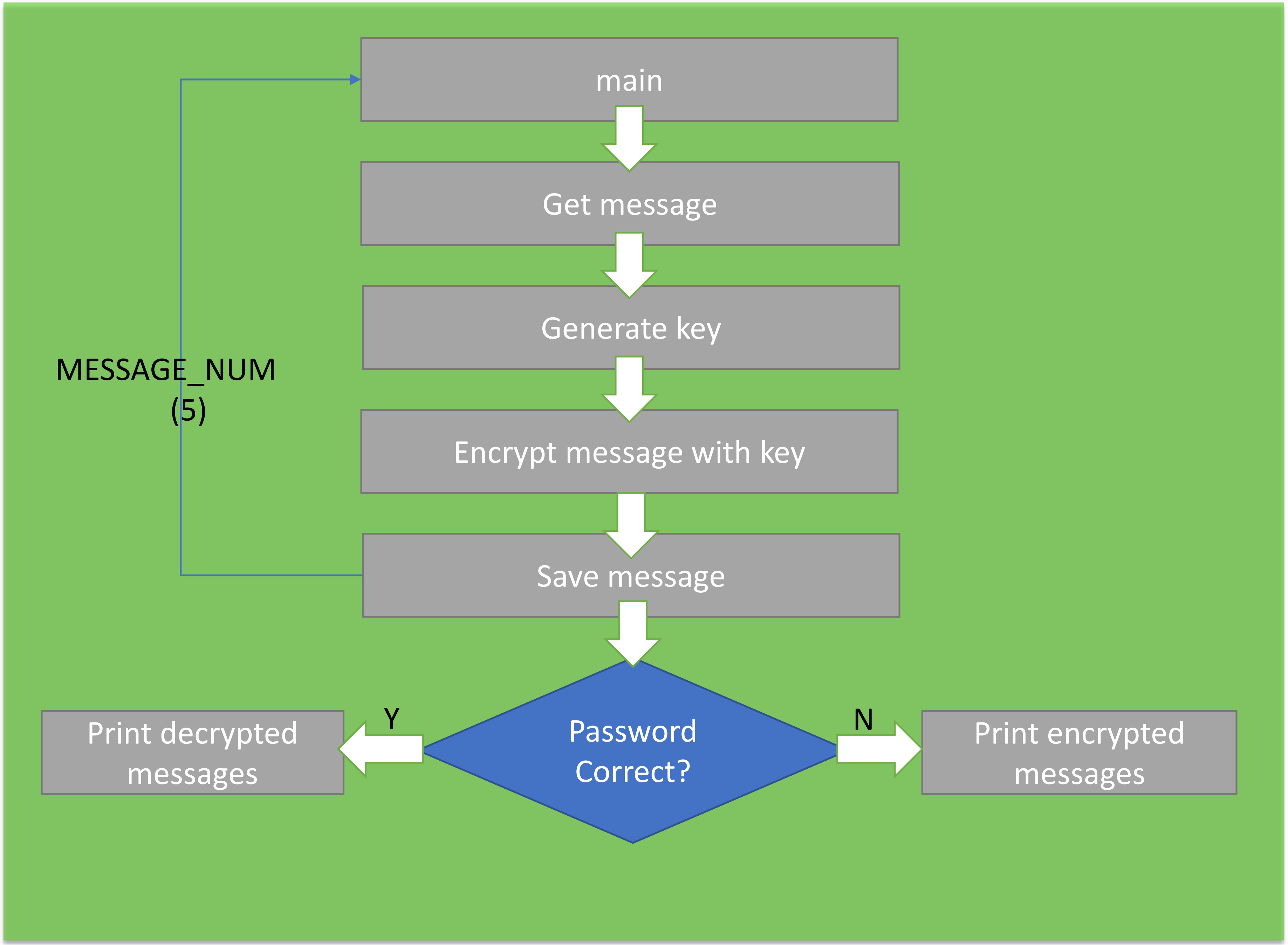
Mac OS

iOS

QNX™

Target Application

@jayHarris_Sec



Demo

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");
}
```

```
onLeave: function(log,retval,state)
{
}
```

@jayHarris_Sec

Frida-trace

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

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

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

Demo

Frida-trace

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

Demo

Scripting Frida

- bindings make Frida scriptable!
- Bindings for Node.js, python, .NET, QML etc

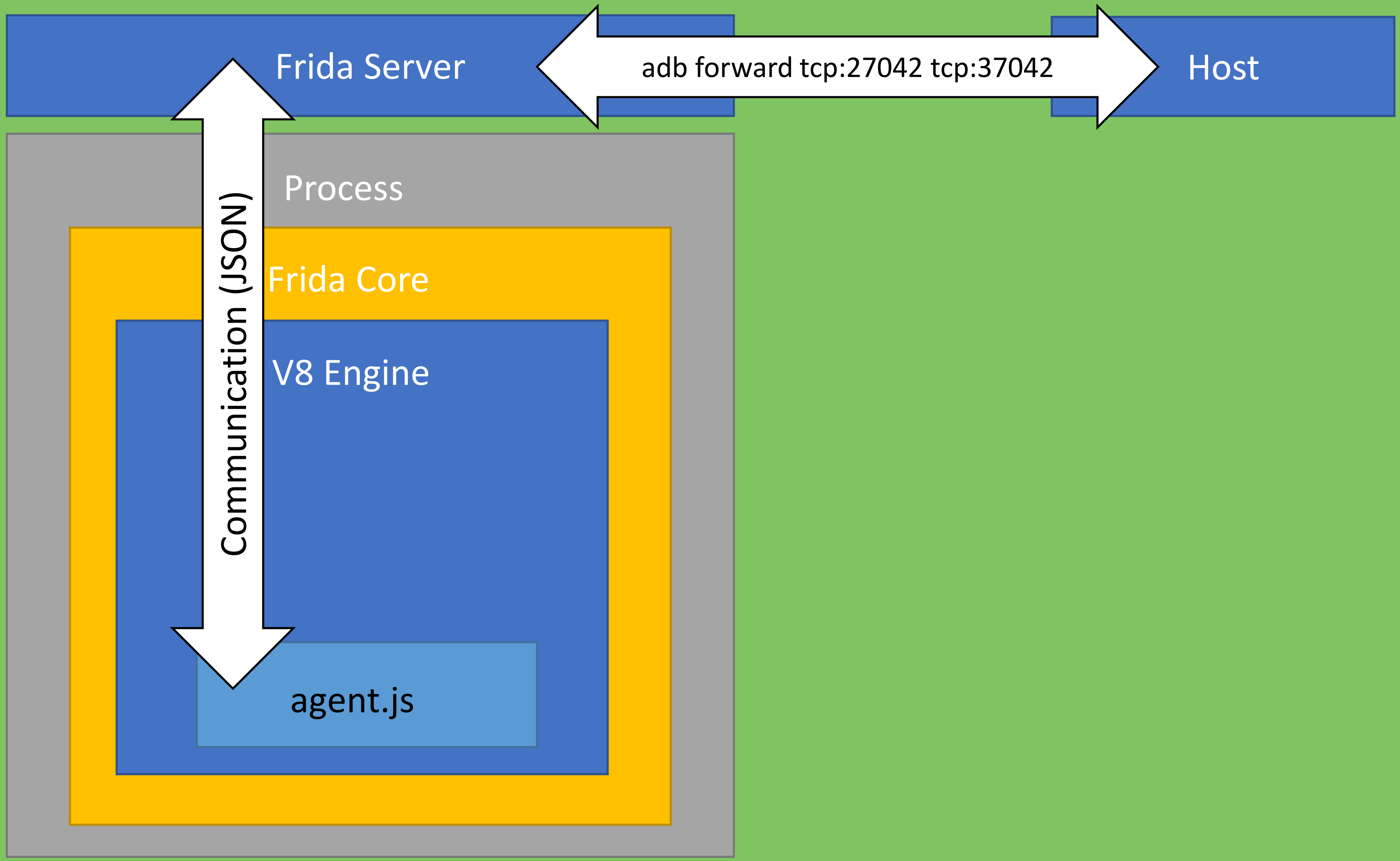
Python template

```
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()
```

Frida and android



Frida and android

- Rooted and non rooted
- Bypass security controls such as SSL Pinning/Root Detection etc

Demo – Bypassing app security

How do we protect against this?

Should we protect against this?