

# Android – Dynamic Analysis

**REVERSE ENGINEERING**

**João Paulo Barraca**

**deti** universidade de aveiro  
departamento de eletrónica,  
telecomunicações e informática

# Dynamic Analysis



**Static Analysis:** Open the application and deduct how it works

Researcher must deduct the Data Flow

External Data or Actions may change the application behavior

- Change the code path
- Inject instructions

Issues may be found on the sequence of events, or on the state machine



**Dynamic Analysis:** Observe the application while it is running, allowing to obtain information about the dynamic characteristics.

# Dynamic Analysis

- **Look into specific aspects of an application, while it is executing**
- **Objective:** Observe dynamic behavior of the application and determine the role of each code
- **What can be analyzed**
  - Messages exchanged with external servers (REST APIs, Web Sockets)
  - Intents sent or received
  - Logs printed (errors, debug messages)
  - Files accessed/created
  - Memory Content
  - With code instrumentation: calls to methods, especially Android API methods

# Logs

- Android log can be used to dynamically analyze relevant aspects of application execution
  - Explicit log entry produced by the application or by system components
  - Implicit logs produced with errors
    - Exceptions produce stack traces which expose call flow
  - Some system events
  - May be used to detect leaks

```
$ adb logcat
```

# Network MiTM

- **Interactions with external APIs can be intercepted and analyzed**
  - Useful to identify communication with domains with low reputation
  - Useful to identify unprotected communications
    - Especially dangerous if dealing with authentication, private data or download of dynamic components
  
- **Black box approach: observe how the app behaves**
  - We may simply observe
  - ... or we manipulate/filter traffic

# Network MiTM

- **Packet Dumps**

- run applications and capture traffic with a packet sniffer
- Non encrypted APIs can be analyzed with ease
  - The endpoint IP address may constitute an indicator by itself
    - Communication with flagged domains, validation that a service is invoked
- Using `wireshark` (`androiddump`)

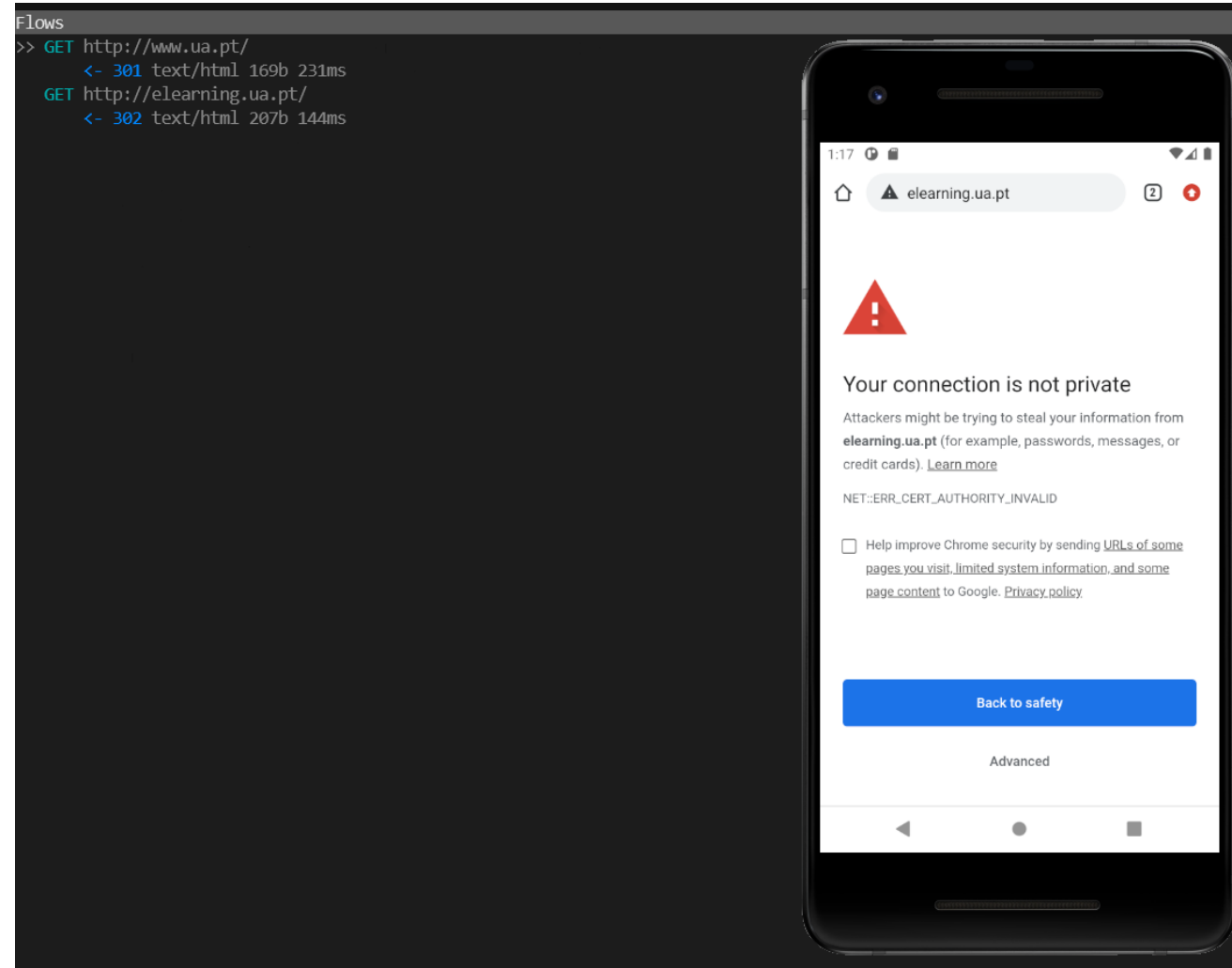
- **Traffic flows**

- run applications with a HTTP/HTTPS proxy configured to intercept all traffic
- injecting a CA Certificate in the device allows generating custom certificates for secure endpoints

# Traffic Flows

- Using an HTTP proxy with Active TLS interception capability
  - Proxy will generate certificates for all hosts accessed
  - Certificates are signed by a single CA
  - CA must be installed to the device

Using mitmproxy, without CA installed  
Alternatives: Charles, ZAP, Burp



# Trusted Certificates

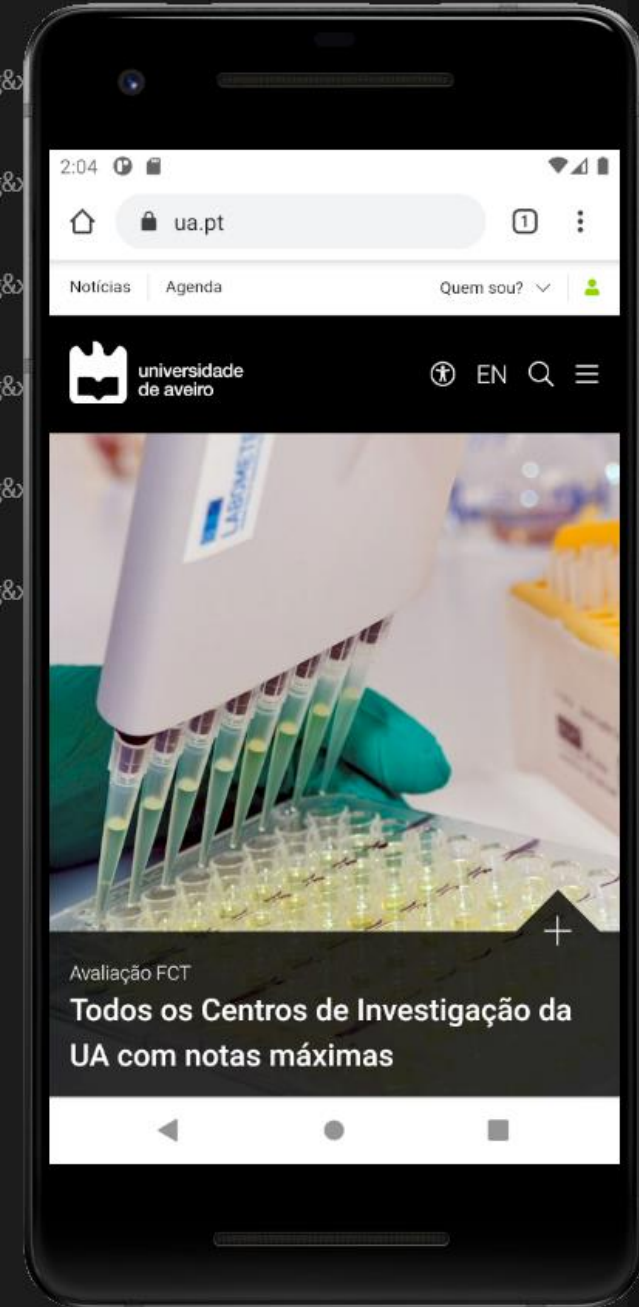
- Standard X509 certificates in PEM format
  - Preinstalled by the manufacturer
  - Cannot be changed by users
  - Users can add custom certificates, but they are frequently ignored by the applications
- On an Android system, trusted roots are at `/system/etc/security/cacerts`
  - Folder with PEM certificates
- `/system` partition is read only on release devices
  - In recent versions of Android the same is also true for the emulator
  - Alternative: mount a tmpfs at the certificate location
    - But changes are lost on reboot



## Using mitmproxy, with CA installed

```
GET https://www.google.com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&w5A8wcQ... HTTP/2.0
<- 200 text/javascript [content missing] 1ms
GET https://www.google.com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&_w5A8wc... HTTP/2.0
GET https://www.google.com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&=_w5A8w... HTTP/2.0
<- 200 text/javascript [content missing] 1ms
GET https://www.google.com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&psi=_w5... HTTP/2.0
GET https://www.google.com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&2&psi=_... HTTP/2.0
<- 200 text/javascript 104b 190ms
GET https://www.ua.pt/ HTTP/2.0
<- 304 [no content] 263ms
GET https://www.ua.pt/static/css/bundle.24c9cca8.css HTTP/2.0
<- 304 [no content] 277ms
GET https://www.ua.pt/static/js/bundle.24c9cca8.js HTTP/2.0
<- 304 [no content] 350ms
GET https://www.ua.pt/styles/bootstrap-grid.min.css HTTP/2.0
<- 304 [no content] 270ms
GET https://www.ua.pt/fontawesome/css/all.css HTTP/2.0
<- 304 [no content] 303ms
GET https://www.ua.pt/styles/entypo.css HTTP/2.0
<- 304 [no content] 364ms
GET https://www.ua.pt/styles/slick.min.css HTTP/2.0
<- 304 [no content] 403ms
GET https://www.ua.pt/styles/slick-theme.min.css HTTP/2.0
<- 304 [no content] 392ms
GET https://www.ua.pt/styles/system-bar.css HTTP/2.0
<- 304 [no content] 381ms
GET https://www.ua.pt/fontawesome/webfonts/fa-light-300.woff2 HTTP/2.0
```

[1/33]



[\*:9090]

# Network MITM - Limitations

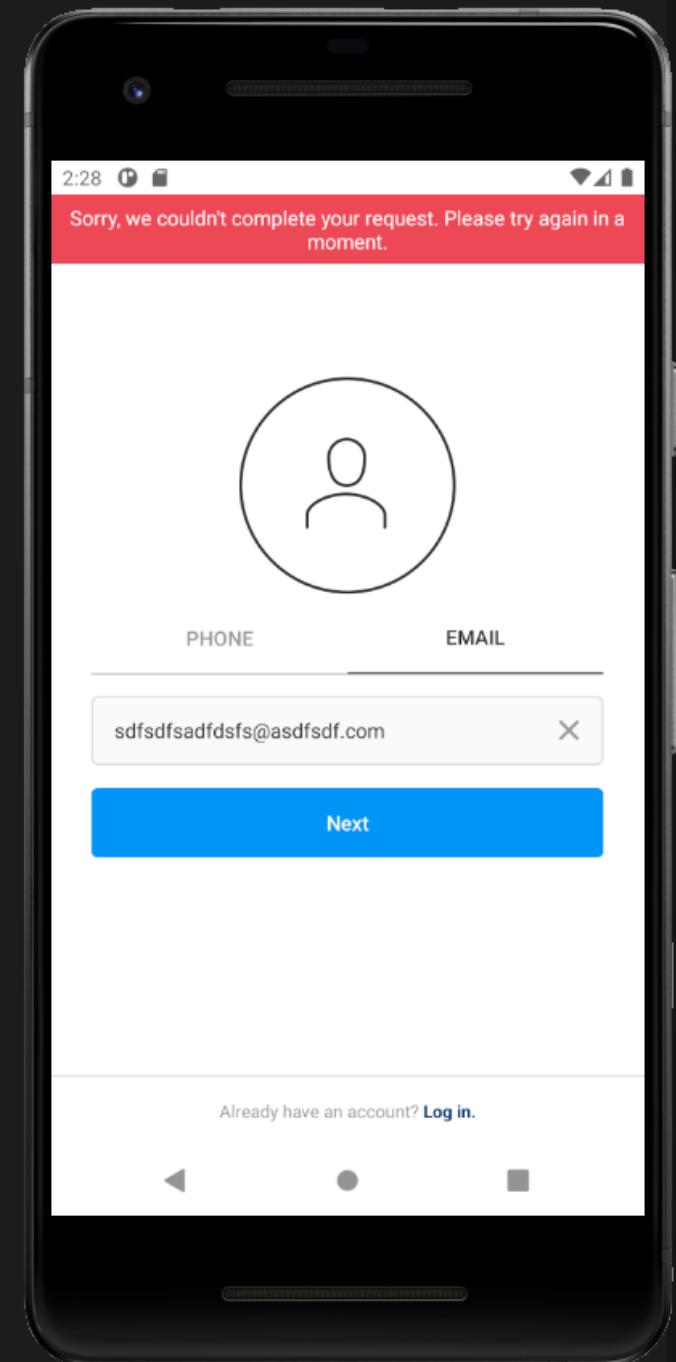
- Packet dumps are limited to unprotected text and metadata
  - Again... it is still relevant as it may produce a valid indicator
  
- Traffic flow analysis is limited to devices where a CA can be injected
  - And where the APP will not use custom CA Certificates
  - And where the APP will not use Certificate Pinning

# Certificate Pinning

- **Applications put constraints on the certificates used for verification (Trusted Roots)**
  - They fix (Pin) a certificate/pub key/hash to a hostname
  - Validation of the host authenticity (in TLS) will also include this additional constraints
- **Impact**
  - A Trusted Root can be injected but it will be ignored
    - Application will simply not use it
    - Or the application will have additional checks with detect the injection

# Certificate Pinning - Approaches

- Applications extend the X509TrustManager, overriding the checkServerTrusted method, with custom checks
  - E.g. the Certificate/Public Key/hash is hard coded, and this value is used to validate the certificate
- Using a KeyStore with a predefined list of certificates, ignoring other sources
  - Pins the host certificate
  - Pins an intermediate Certification Authority
  - Pins a Root Certification Authority
- Pinning may create issues for developers as changes to certificates or PKI must be reflected to the applications
  - Soft Fail: just let the application work, even if with limited functionality
  - Hard Fail: an update is forced for the application to work



Applications using Pinning will not communicate through a proxy

# Certificate Pinning - Circumvention

- **If restricted KeyStores are used:** use an emulator or rooted device
  - Enables free manipulation of the keystores, injecting custom certificates
  - Inject certificates to the system keystores
  
- **If Pinned with hard coded information:** modify the application
  - Unpack the application
  - Edit the code, changing the Pin or removing it
    - `smali` may be enough and full decompilation to java is not required
  - Repack and install the application

# Dynamic Code Instrumentation

- Applications are implemented with functions
  - Functions have addresses that may be determined
    - Or subverted
  - Java uses further abstractions when using native code
    - Creating strict interaction points towards which is possible to access the external world
- This enables the possibility of manipulating symbols/addresses to instrumentalize code
  - Observe internal structure of the program flow
  - Inject new code by replacing the implementation of a function represented by a symbol

# Dynamic Binary Instrumentation - Why

- Requests to APIs are further encrypted, signed or MITM is not available
  - MITM and packet sniffers are useless
- Application has obfuscated values in RAM, created dynamically, received from the network
  - Static analysis and Decompilation is useless
- Code is loaded dynamically with objects received
  - Static analysis and Decompilation will have no code to analyze
- Many values are hard coded (keys, urls...)
  - Patching takes too long and becomes expensive



# Custom Signatures are used

POST /login HTTP/1.1

Host: social.io

Proxy-Connection: keep-alive

Content-Length: X

Accept: text/html, application/xhtml-xml,application/xml;q=0.9,image/webp,\*/\*,q=0.8

Origin: <http://social.io>

Content-Type: application/x-www-form-urlencoded

Cookie: SessionId=0+qxnaYZLjpnLwHBcKmRcTexTWk=

username=john&password=xpto&signature=2rf+roJPEdCOSL0XXusHBcA0BGk=

# Data is encrypted

```
POST /login HTTP/1.1
Host: social.io
Proxy-Connection: keep-alive
Content-Length: X
Accept: text/html, application/xhtml-
xml,application/xml;q=0.9,image/webp,*/*,q=0.8
Origin: http://social.io
Content-Type: application/x-www-form-urlencoded
Cookie: SessionId=0+qxnaYZLjpnLwHBcKmRcTexTvk=
```

```
authData=3NH71S+7P8YeafgnBvXzJ1RzJdXm51VNPQYMWF iIMl8Znr7+vGDNTcms8LHDUaC/lK2xRF/L
bPMwQ0pB+ZyB6PfYNaf5fIh/IGdlQZJrgXXgDDT7Mn2d259vzcdmBA3pJ04cLxGNnLSvdorYF+mLN7yik
zEagUWGfQe1nYzu30T3947kqSORQuc4PTzuFKUXlo1CcuVYvr5gt6ykfk9ACGVwyywGBG30eFxNKi0kme
iBYxB8EJlmCF/xojM59gcGDv61ytidhVs=
```

# Many others

- Retrieving a call flow
  - Map which methods are used, and what is the actual code execution flow
- Identify arguments of Android API methods
  - Log traffic and calls
  - Allows intercepting data even with encrypted connections
    - Interception happens before data is encrypted
- Modify arguments of Android API methods
  - Fuzzing
  - Filter/modify data to trigger additional behavior
  - Trigger custom events
- Circumvent protections to enable further analysis
- Application is obfuscated and it is difficult to obtain the actual algorithm

# FRIDA – How?

- Set of tools (framework) for Dynamic Code Instrumentation
  - Instruments the Application Code with hooks
  - Not specific for Android, and may be used on other applications and Operating Systems
  
- Allows:
  - Tracing network communications at the method level
  - Understand how the application behaves
  - Manipulate the methods called, arguments and return codes
  - ...

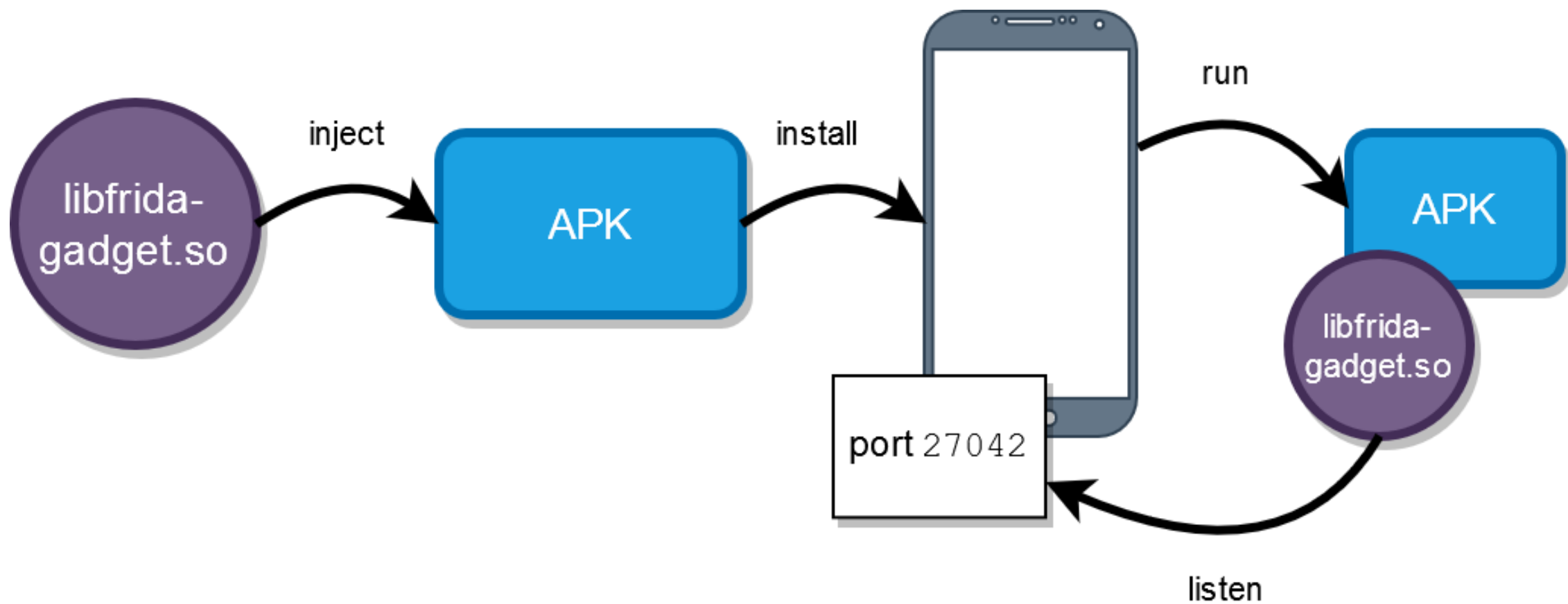
# FRIDA – How?

- Frida-core injecting a Google V8 JS Engine into the App scope
  - Frida-core Written in C
  - GumJS (the JS part) is packed as a shared library and loaded into the app
- GumJS has access to the application memory
  - Can be hooked to methods and intercept calls, even native APIs
- GumJS API allow interaction with GumJS from an external client
  - Because GumJS resides the application address space, it has full access to its structures

# FRIDA – mode Embedded

- Frida agent is embedded as a dependency of an existing application library
  - Requires the application to have an .so
  - `libfrida-gadget.so` is injected into the existing library and loaded at the same time
- Alternative: existing library is modified in order to load the additional library
  - Requires patching the code in a function will be called (e.g. `JNI_OnLoad`)
- Alternative: patching the `smali` code to load the library
  - Obtain APK
  - Extract smali
  - Change smali
  - Pack it and install
- Method implies that the application is repacked/resigned

# FRIDA – mode Embedded



# FRIDA – mode Embedded – How?

```
$ apktool d app.apk
$ cp libfrida-gadget.so target/lib/arm
$ python3
>>> import lief
>>> native = lief.parse("target/lib/arm/libsomething.so")
>>> native.add_library("libfrida-gadget.so")
>>> native.write("target/lib/arm/libsomething.so")
>>> exit

$ apktool b target
... sign ... install
```



# FRIDA – mode Embedded – Smali - How

- Unpack the app using apktool
- Patch the smali with

```
const-string v0, "frida-gadget"  
invoke-static {v0}, Ljava/lang/System; ->loadLibrary(Ljava/lang/String;)V
```

- Where? In the main activity constructor
  - Even in as a static property of the class
- repack, sign, install

# FRIDA – mode Embedded - Caveats

- Applications may search for the library name as an anti-debug technique
  - May need to change the library name
- Must use a version compatible with the target architecture
  - <https://github.com/frida/frida/releases>
- Agent may only be loaded after the JNI library is loaded or code is reached
  - After `System.loadLibrary("lib.so")`
- Agent may impose the need for permissions to access the INTERNET
  - Manifest may need to be updated

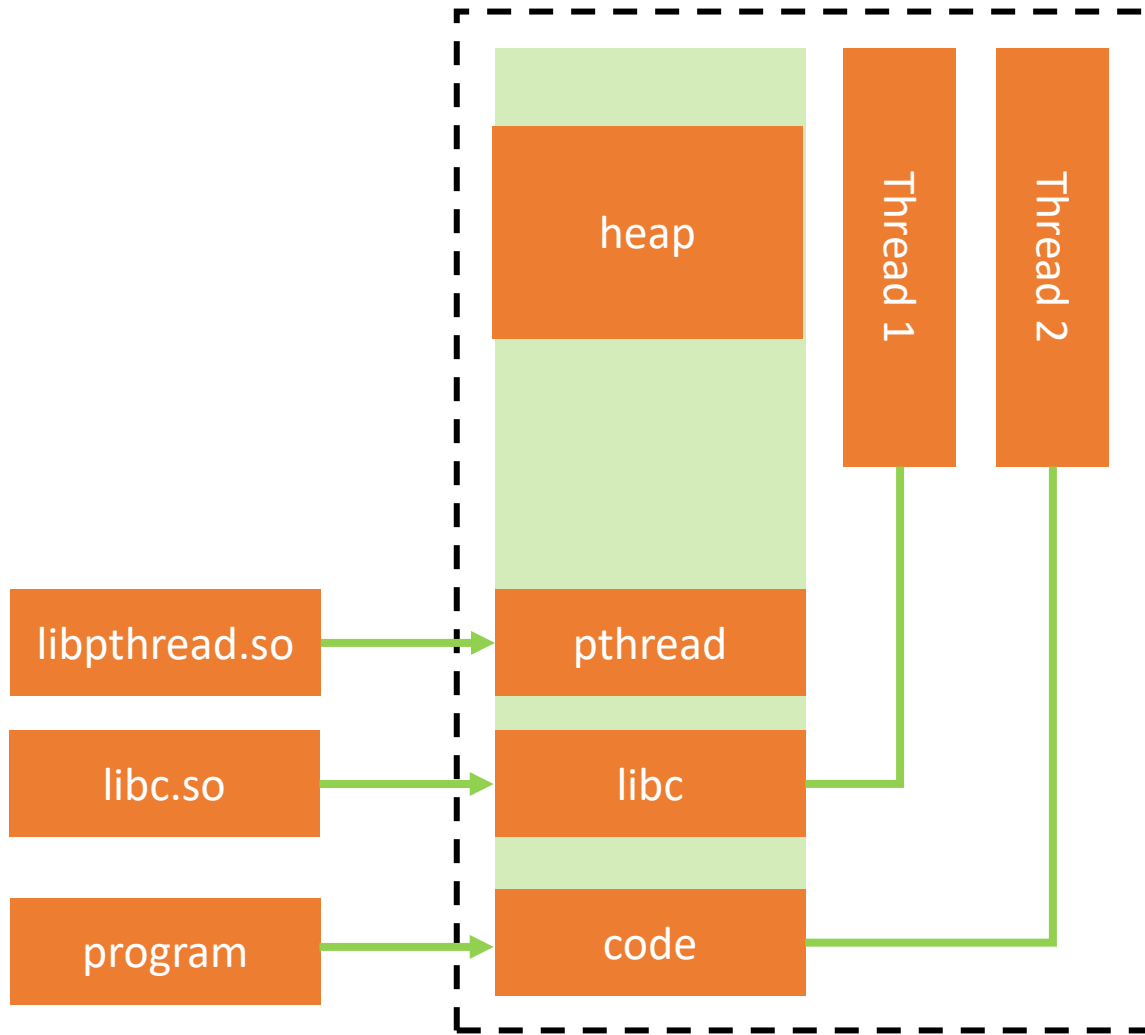
# FRIDA – mode Injected

- Run a Frida Server which injects the agent into the target process
  - Server provides an API for remote use
  - Server injects the agents into applications
  
- Requires the smartphone to be rooted or to be an emulator
  - In order to run the server and inject applications
  
- Cannot be used in production builds, only development
  - When in an Emulator, use a base without WITHOUT “Google Services”

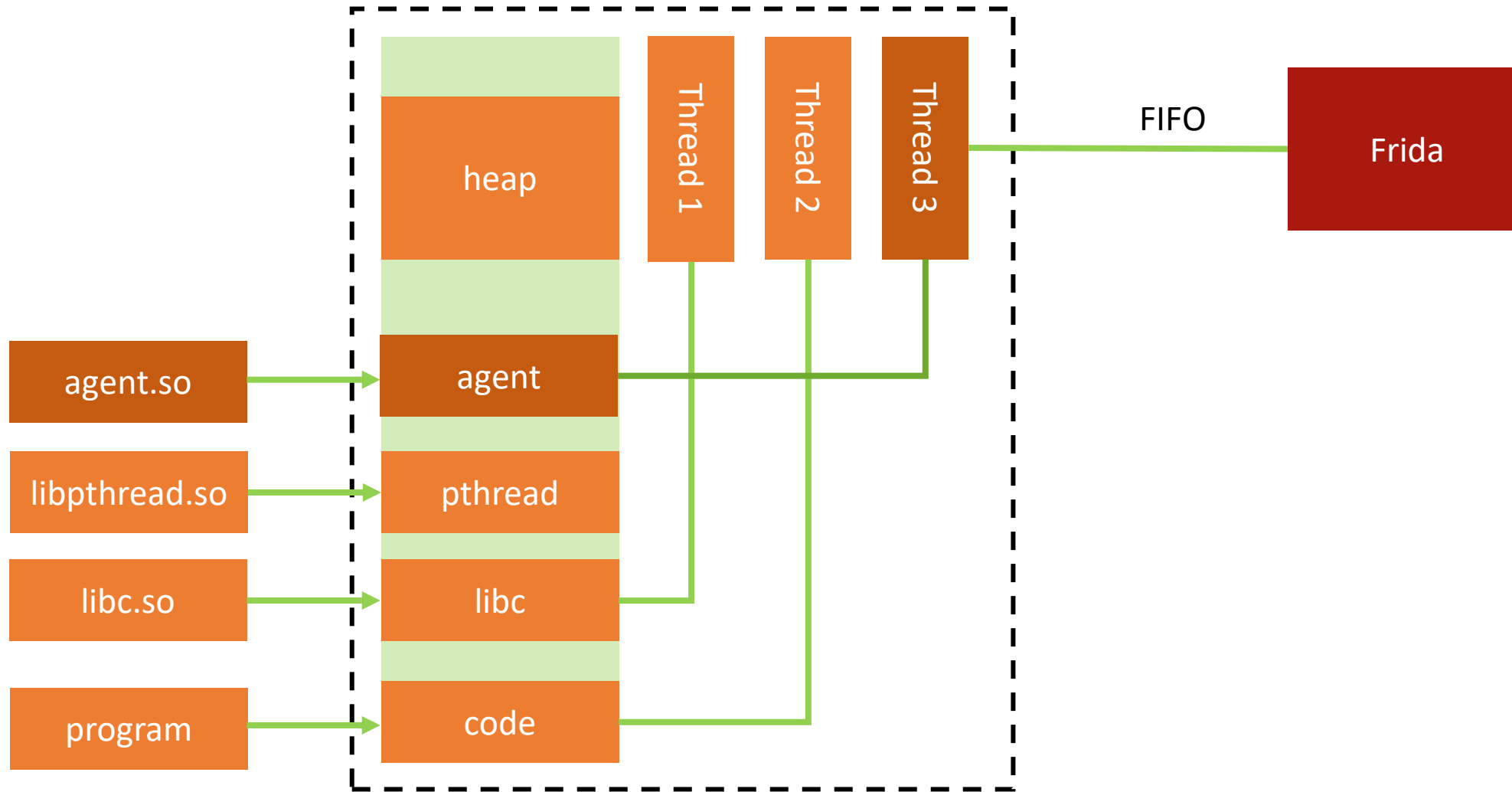
# FRIDA – mode Injected – How it works?

- Create an Agent: it's an `.so` with some custom code
- Start a server that will be ready to attach to processes
- Injection:
  - Create thread in the remote process using `ptrace`
    - `PTRACE_ATTACH`, `PTRACE_GETREGS`
  - Allocate memory for a bootstrapping code
    - Minimal amount of code required for pulling the agent
  - Write bootstrapper to memory
  - Execute bootstrapper in remote process
    - Open communication channel to server (FIFO)
    - Loads the agent from a shared library (`.so`)
    - Executes the agent
    - Closes communication channel (agent will expose an API)

# FRIDA – mode Injected



# FRIDA – mode Injected



# FRIDA – mode Injected

- Required functionality:
- ptrace
  - Process tracing
- mmap
  - Map files to memory. In particular, the agent .so
- dlopen
  - Open the .so with the agent
- dlsym
  - Retrieves addresses of loaded symbols
- signal
  - To handle system signals

# FRIDA – mode Injected - Howto

```
### On the first PC terminal

# wget frida-server from github

adb push frida-server /data/local/tmp
adb shell
su
cd /data/local/tmp
chmod +x frida-server
./frida-server

### On the second PC terminal
# List processes
$ frida-ps -U
```



# FRIDA – How to use

- Command line tools: frida, frida-trace, frida-ps, frida-discover..
- Python interface
  - Provides a more advanced, programmatic interface
  - Allows predictable and repeatable instrumentation
- How to instrument code: using JS that overload existing functions
  - Large repository at: [Frida CodeShare](#)

# Example: com.re.lab1

- Application requires a pin to unlock the flag
  - Pin is created dynamically and stored to an encrypted database
  - Application cannot be tampered as it checks the signature
- Static analysis will yield little as the value is created on real time
- Approaches to dynamic analysis:
  - Insert a function to access the correct pin and log it to the terminal

**Let's break the check. Just to test.  
Objective: make b.checkAppSignature return false**

```
int d1 = b.checkAppSignature(this);
if(d1 < 1){
    Toast.makeText(context, "Application Tampered", Toast.LENGTH_LONG).show();
    this.finishAffinity();
}
try{
```

## com.re.lab1 - Java.perform: executes the given payload

Snippet provides an alternative implementation of the method.

```
Java.perform(function(){
    Java.use("com.re.lab1.b").checkAppSignature.implementation = function(a) {
        console.log("Signature will fail");
        return 0;
    };
});
```

```
[jpbarraca@wintermute] frida -U -f com.re.lab1 -l signatureFail.js
```

```
┌───┐   Frida 14.2.13 - A world-class dynamic instrumentation toolkit  
│───┘  
│  (  │  
│  >  │   Commands:  
│  /_  │   help      -> Displays the help system  
│  /_  │   object?   -> Display information about 'object'  
│  /_  │   exit/quit -> Exit  
│  /_  │  
│  /_  │   More info at https://www.frida.re/docs/home/  
└───┘  
Spawned `com.re.lab1`. Use %resume to let the main thread start executing!  
[Android Emulator 5554::com.re.lab1]-> %resume  
[Android Emulator 5554::com.re.lab1]-> Signature will fail  
█
```



pp has the pin provided

Cursor has the value obtained form the DB

```
cursor = secureDB.rawQuery("SELECT * FROM a;",null);
cursor.moveToFirst();

if(pp.equalsIgnoreCase(cursor.getString(0))){
    Toast.makeText(MainActivity.this, "Right Pin, Congratulations", Toast.LENGTH_SHORT).show();
    pin1.removeAllViews();
    String xo = getResources().getString(R.string.google_api_key);
    a mo = new a();
    xo = mo.func1(xo,xo.substring(4));
    xo = a.func2(xo);
    xo = a.func3(xo.substring(1),xo);
    xo = a.func4(xo,xo,xo.substring(2));
    tv1.setText("Flag: "+xo);
}else{
    Toast.makeText(MainActivity.this, "Incorrect Pin, "+(max_tries+1-i)+" attempts remaining",
}
}
```

**Objective: reimplement `Java.lang.String.equalsIgnoreCase` so that it return true, and prints the correct ping**

```
Java.perform(function(){
  Java.use("java.lang.String").equalsIgnoreCase.implementation = function(a) {
    console.log("Real PIN: " + a);
    return true;
  };
});
```

```
[jpbarraca@wintermute] frida -U -f com.re.lab1 -l equalsIgnoreCase.js
```

```
┌───┐
│   │ Frida 14.2.13 - A world-class dynamic instrumentation toolkit
├───┘
│   │
│   │ Commands:
│   │ help      -> Displays the help system
│   │ object?   -> Display information about 'object'
│   │ exit/quit -> Exit
│   │
│   │ More info at https://www.frida.re/docs/home/
```

```
Spawned `com.re.lab1`. Use %resume to let the main thread start executing!
```

```
[Android Emulator 5554::com.re.lab1]-> %resume
[Android Emulator 5554::com.re.lab1]-> Real PIN: INSERT
Real PIN: INSERT
Real PIN: INSERT
Real PIN: INSERT
Real PIN: 4597
```

Other uses of the method

Finally, the PIN





# Interceptors: Intercepts calls to a function

- Define two events where code can be executed
  - **OnEnter**: When the function is called
  - **OnLeave**: After the function returns
- Can be used as generic logger, or to trigger other actions
  - Can intercept calls on lower layers of the application stack
    - Data that is to be written, sql queries, etc...

```
function foo(){
  Interceptor.attach(Module.findExportByName("libc.so", "open"), {
    onEnter: function(args){
      console.log("Entering the function");
    },
    onLeave: function(args){
      console.log("Leaving the function");
    },
  });
}
```