
download
FRIDA: Intro and Live Demo
Cesare Pizzi
Security Analyst
SORINT lab

WHO AM I

?

FRIDA: Introduction

What is FRIDA?

FRIDA is a DBI (Dynamic Binary Instrumentation toolkit) built to inject code into existing running processes.

In other words it allow to interact with live processes.

FRIDA: Introduction

FRIDA is not exactly a debugger

It allows you to make some "on-the-fly" modifications to running code.

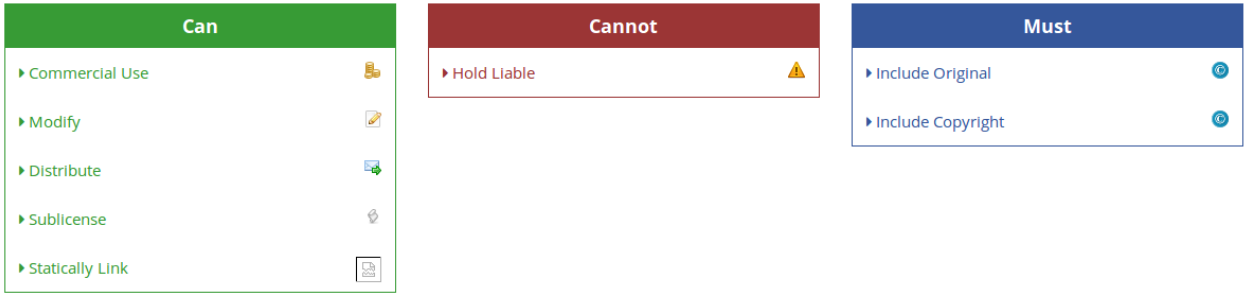
From FRIDA site:

"lets you inject snippets of Javascript or your own library into native apps on Windows, MacOS, Linux, iOS, Android and QNX"

FRIDA: Introduction

Some key features

- Not only Java, but can inject **libc** based apps
- Scriptable: execute your own code in the foreign process
- Python compatible: you can use the python library
- Multi-platform
- Open Source: wxWindows license



from https://tldrlegal.com

FRIDA: Introduction

Why should I use it?

- Reverse Engineering
- Live and "race-condition" debugging
- Create live and ephemeral patching

FRIDA: Introduction

Reverse engineering? Is that legal? Why should I do it?

We know we are in a gray area...

Article 6 (2009/24/EC)

Decompilation

1. The authorisation of the rightholder shall not be required where reproduction of the code and translation of its form within the meaning of points (a) and (b) of Article 4(1) are indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs, provided that the following conditions are met:

(a) those acts are performed by the licensee or by another person having a right to use a copy of a program, or on their behalf by a person authorised to do so;

(b) the information necessary to achieve interoperability has not previously been readily available to the persons referred to in point (a); and

(c) those acts are confined to the parts of the original program which are necessary in order to achieve interoperability.

FRIDA: Introduction

What about Italy?

From https://it.wikipedia.org/wiki/Reverse_engineering:

Nel caso specifico italiano, la reingegnerizzazione a scopo di interoperabilità con altri sistemi (e solo a questo scopo) è un atto pienamente lecito ai sensi dell'art. 64 della legge 633 del 22 aprile 1941, come modificata dall'art. 5 del D. Lgs. 518/1992, sia in senso "leggero" (qualora egli compia tali atti durante operazioni di caricamento, visualizzazione, esecuzione, trasmissione o memorizzazione del programma che egli ha il diritto di eseguire) che in senso di decompilazione vera e propria, ma solo al fine di permettere l'interoperabilità del software con altri programmi.
L'accezione di software è estesa per analogia a concetti informatici quali il formato di un file o la struttura interna di un protocollo.

FRIDA: Introduction

And so?

Well, we'll use a test app...

WhyShouldIPay.apk


DEMO TIME!
download
SORINT lab

FRIDA: live DEMO

What we need

Software needed for the DEMO:

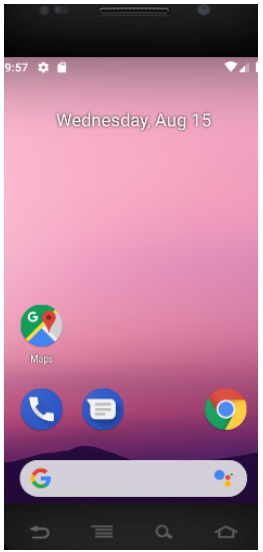
- FRIDA (# pip install frida)
- ByteCodeViewer (https://bytecodeviewer.com/)
- Android Phone or Android emulator (Android Studio)

FRIDA: live DEMO

Topology

FRIDA

FRIDA Core
(PC)



FRIDA server

FRIDA: live DEMO

Let’s start

```
cesare@linux:~$ adb devices
List of devices attached
emulator-5554 device
```

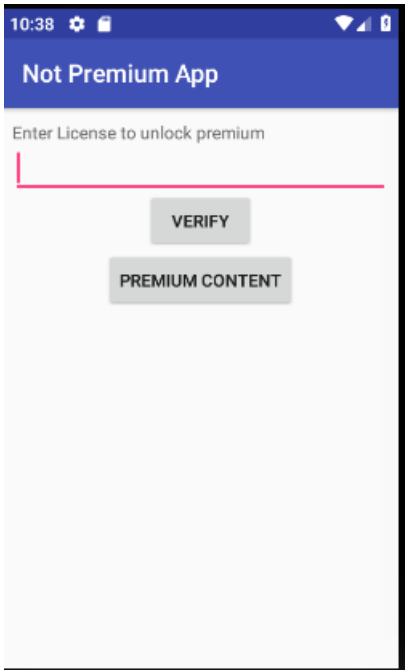
```
cesare@linux:~$ adb push frida-server-11.0.11-android-x86 /data/local/tmp/frida-server
7484 KB/s (24332004 bytes in 3.174s)
```

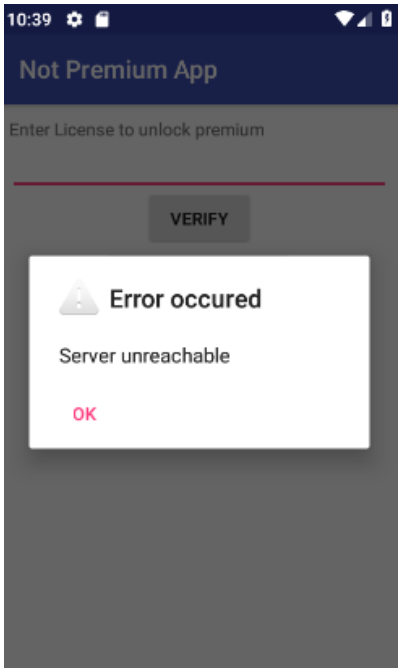
```
cesare@linux:~$ adb shell
generic_x86:/ $ su
generic_x86:/ # cd /data/local/tmp
generic_x86:/data/local/tmp # chmod 755 frida-server
generic_x86:/data/local/tmp # ./frida-server
```

FRIDA: live DEMO

Install the APP and have a look to it

```
cesare@dell:~$ adb install WhyShouldIPay.apk
7159 KB/s (1556472 bytes in 0.212s)
Success
```





FRIDA: live DEMO

Start ByteCodeViewer and look for “unreach”

```
/*
 * Enabled unnecessary exception pruning
 */
public void verifyClick(View view) {
    String string = ((EditText)this.findViewById(2131427421)).getText().toString();
    try {
        InputStream inputStream = new URL("http://broken.license.server.com/query?license=" +
                string).openConnection().getInputStream();
        StringBuilder stringBuilder = new StringBuilder();
        byte[] arrby = new byte[]{};
        while (inputStream.read(arrby) > 0) {
            stringBuilder.append(arrby);
        }
        String string2 = stringBuilder.toString();
        if (string2.equals("LICENSEKEYOK")) {
            String string3 = new String(MainActivity.xor((byte[])this.getMac().getBytes(), (byte[])string2.getBytes()));
            SharedPreferences.Editor editor = this.getApplicationContext().getSharedPreferences("preferences", 0).edit();
            editor.putString("KEY", string3);
            editor.commit();
            new AlertDialog.Builder((Context)this).setTitle((CharSequence)"Activation successful").
                    setMessage((CharSequence)"Activation successful").setIcon(17301543).show();
            return;
        }
    }
    catch (Exception exception) {
        new AlertDialog.Builder((Context)this).setTitle((CharSequence)"Error occured").
                setMessage((CharSequence)"Server unreachable").setNeutralButton((CharSequence)"OK", null).setIcon(17301543).show();
        return;
    }
    new AlertDialog.Builder((Context)this).setTitle((CharSequence)"Invalid license!").
            setMessage((CharSequence)"Invalid license!").setIcon(17301543).show();
}
```

FRIDA: live DEMO

That’s interesting...

```
InputStream inputStream = new URL("http://broken.license.server.com/query?license=" +
        string).openConnection().getInputStream();
StringBuilder stringBuilder = new StringBuilder();
byte[] arrby = new byte[]{};
while (inputStream.read(arrby) > 0) {
    stringBuilder.append(arrby);
}
String string2 = stringBuilder.toString();
if (string2.equals("LICENSEKEYOK")) {
    String string3 = new String(MainActivity.xor((byte[])this.getMac().getBytes(), (byte[])string2.getBytes()));
    SharedPreferences.Editor editor = this.getApplicationContext().getSharedPreferences("preferences", 0).edit();
    editor.putString("KEY", string3);
    editor.commit();
    new AlertDialog.Builder((Context)this).setTitle((CharSequence)"Activation successful").
            setMessage((CharSequence)"Activation successful").setIcon(17301543).show();
    return;
}
```

FRIDA: live DEMO

Our first FRIDA script

```
setImmediate(function() { //prevent timeout
  console.log("[*] Starting script");

  Java.perform(function () {
    var cls_launcher_activity = Java.use("de.fraunhofer.sit.premiumapp.LauncherActivity");

    cls_launcher_activity.verifyClick.implementation = function () {
      console.log("[*] verifyClick got called!\n");
    };
  });
});
```

```
cesare@dell:~$ frida -U -l apk1.js de.fraunhofer.sit.premiumapp
```

FRIDA: live DEMO

Our first FRIDA script results

```
  ____
 / _  |   Frida 11.0.11 - A world-class dynamic instrumentation toolkit
| (_| |
 > _  |   Commands:
/_/ |_|       help      -> Displays the help system
. . . .       object?   -> Display information about 'object'
. . . .       exit/quit -> Exit
. . . .
. . . .   More info at http://www.frida.re/docs/home/

[*] Starting script
[Android Emulator 5554::de.fraunhofer.sit.premiumapp]->
 [*] verifyClick got called!
[Android Emulator 5554::de.fraunhofer.sit.premiumapp]->
```

FRIDA: live DEMO

One step further...

```
setImmediate(function() { //prevent timeout
  console.log("[*] Starting script");

  Java.perform(function () {
    var cls_launcher_activity = Java.use("de.fraunhofer.sit.premiumapp.LauncherActivity");

      cls_launcher_activity.verifyClick.implementation = function () {
      console.log("[*] verifyClick got called!\n");
      console.log("[*] Read MAC: " + this.getMac());

      editor = this.getApplicationContext().getSharedPreferences("preferences", 0).edit();
      editor.putString("KEY", "Trythisone");
      editor.commit();
      console.log("[*] Exiting verifyCLick\n");
    };
  });
});
```

FRIDA: live DEMO

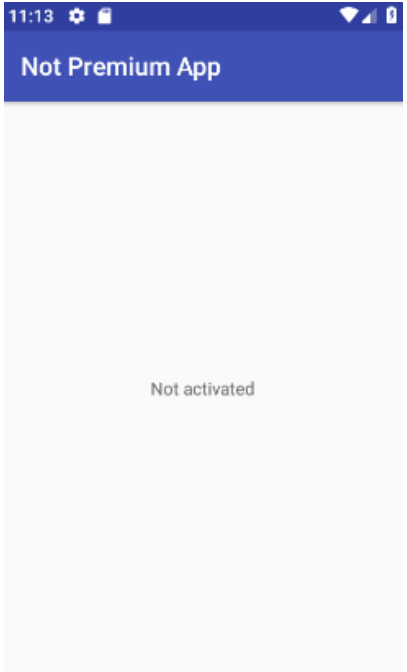
Mmmmmm.....

```
   ____
  / _  |   Frida 11.0.11 - A world-class dynamic instrumentation toolkit
 | (_| |
  > _  |   Commands:
 /_/ |_|       help      -> Displays the help system
 . . . .       object?   -> Display information about 'object'
 . . . .       exit/quit -> Exit
 . . . .
 . . . .   More info at http://www.frida.re/docs/home/

[*] Starting script
[Android Emulator 5554::de.fraunhofer.sit.premiumapp]-> [*] verifyClick got called!

[*] Read MAC: 02:00:00:00:00:00
[*] Exiting verifyCLick

[Android Emulator 5554::de.fraunhofer.sit.premiumapp]->
```



FRIDA: live DEMO

Another try?

```
cls_launcher_activity.verifyClick.implementation = function () {
  console.log("[*] verifyClick got called!\n");
  console.log("[*] Read MAC: " + this.getMac());
  strMAC = strToBytes(this.getMac());
  strLIC = strToBytes("LICENSEKEYOK");
  strKEY = byteToStr(cls_main_activity.xor(strMAC,strLIC));
  console.log("[*] Computed KEY: " + strKEY);

  editor = this.getApplicationContext().getSharedPreferences("preferences", 0).edit();
  editor.putString("KEY", strKEY);
  editor.commit();
  console.log("[*] Exiting verifyCLick\n");
};
```

FRIDA: live DEMO

:-)

```
   ____
  / _  |   Frida 11.0.11 - A world-class dynamic instrumentation toolkit
 | (_| |
  > _  |   Commands:
 /_/ |_|       help      -> Displays the help system
 . . . .       object?   -> Display information about 'object'
 . . . .       exit/quit -> Exit
 . . . .
 . . . .   More info at http://www.frida.re/docs/home/

[*] Starting script
[Android Emulator 5554::de.fraunhofer.sit.premiumapp]->
[*] verifyClick got called!
[*] Read MAC: 02:00:00:00:00:00
[*] Computed KEY: |{yu~iu{iq|yyu~
[*] Exiting verifyCLick

[Android Emulator 5554::de.fraunhofer.sit.premiumapp]->
```

FRIDA: References

Where to find additional infos

https://www.frida.re/
https://github.com/frida/frida
https://github.com/cecio (scripts of this exercise)
https://team-sik.org/wp-content/uploads/2017/06/WhyShouldIPay.apk_.zip
https://developer.android.com/studio/


download
Thanks!
SORINT lab