



# Entering the Android Rabbit Hole with Frida:

## Deep-Dive into Dynamic Analysis

# About me

*<whoami>*

🔎 **St fany Coura Coimbra**

🇧🇷 **Brazilian**

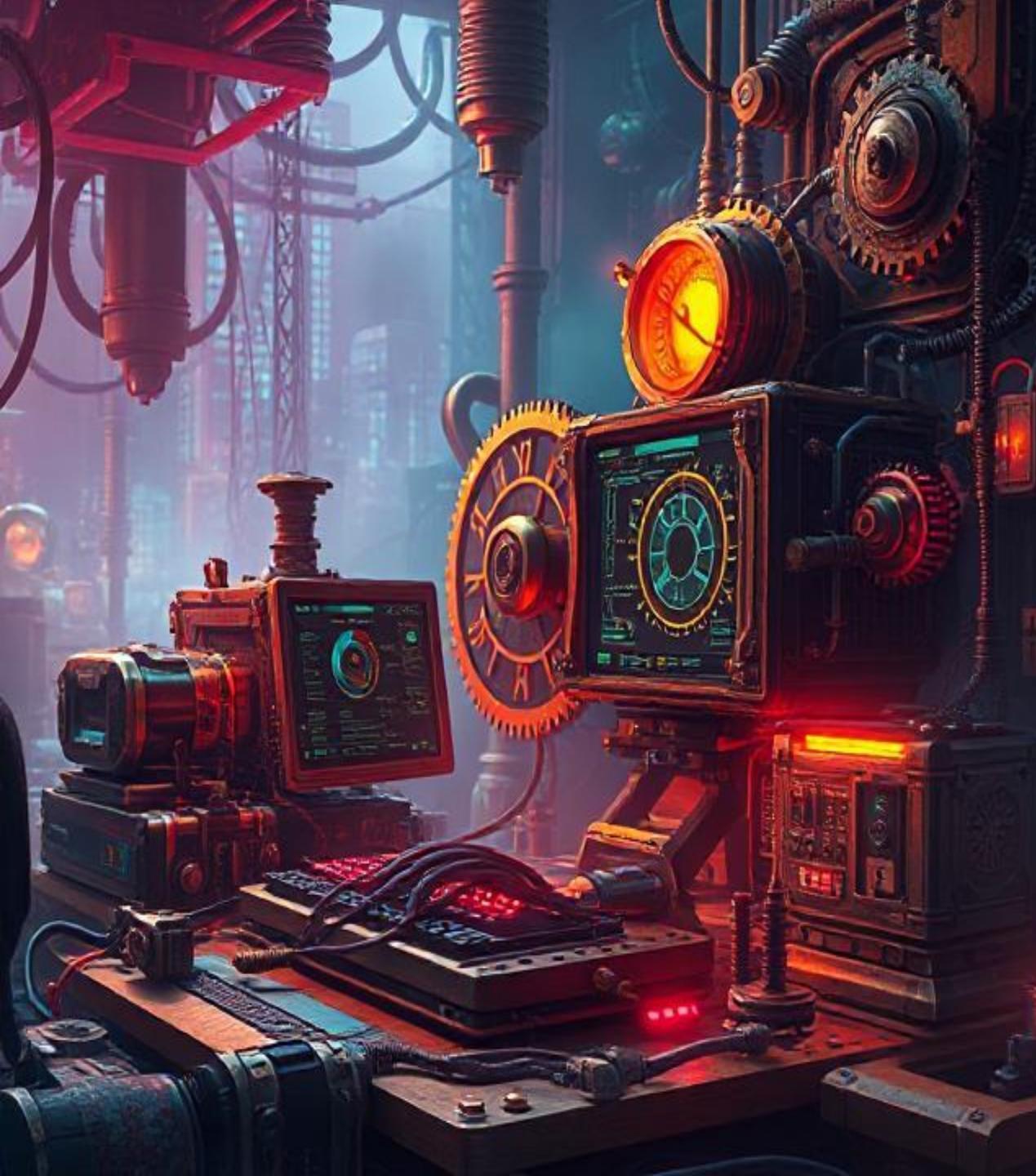
💻 **Offensive Security Analyst / Cybersecurity Researcher**



*<uname -a>*

🎓 **B.Sc. in Computer Engineering – Federal University of Itajub  (Unifei),  
Brazil – 2024**

🛡️ **Fields of interest: cryptography algorithms, web and android security**



# Introduction

## Understanding the **Android** Landscape

# The mind of the security analyst



## Android Security Framework

- Key security features: App sandboxing, permissions model, and security updates



## Common Vulnerabilities in Android Apps

- Exploits like Intent Redirection, Insecure Data Storage, Improper Implementation of Cryptography, and Insecure Communication



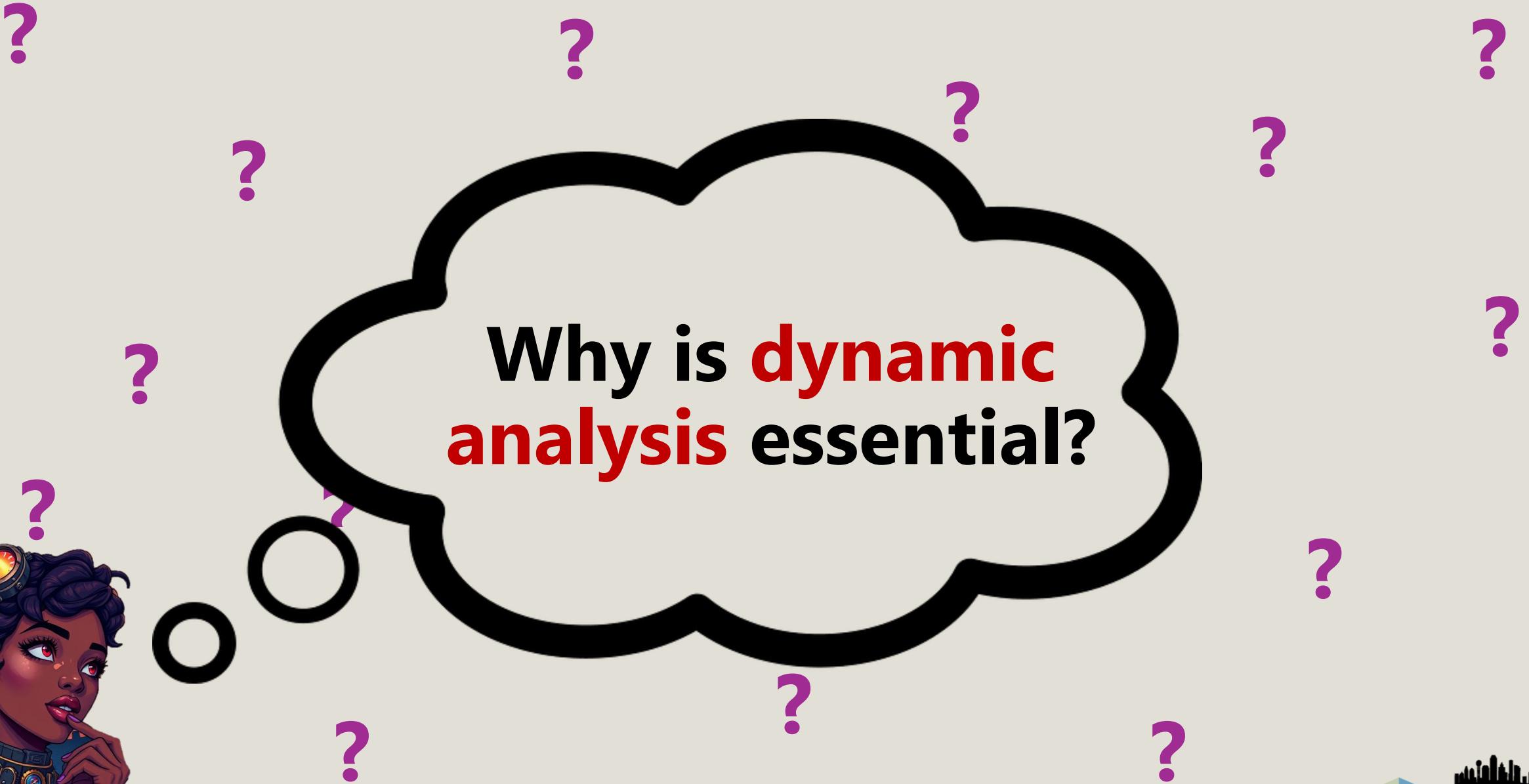
## Static and Dynamic Analysis Techniques

- Tools and methods for analyzing APKs: reverse engineering, code inspection, and runtime behavior monitoring





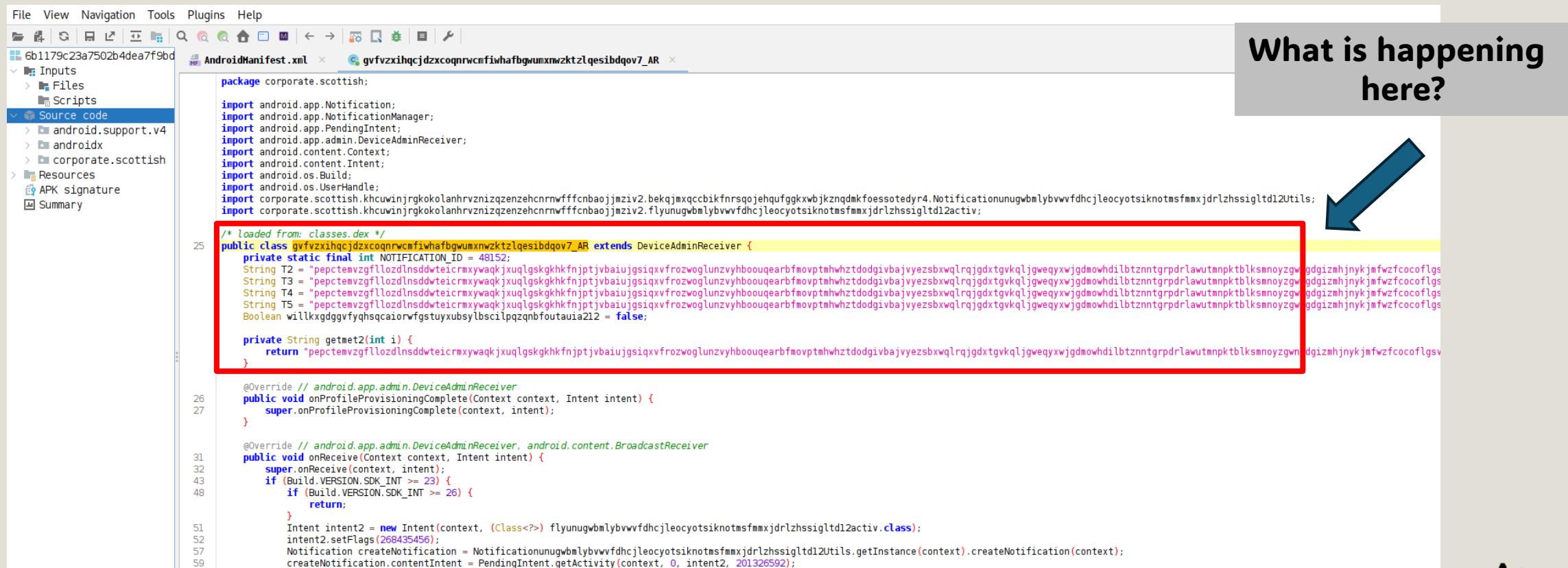
Why is **dynamic**  
**analysis** essential?



# Let's take a look at an example!

The reference sample (SHA-256):

6b1179c23a7502b4dea7f9bde7dde3d4b5b97c64f634ff3471a1d3d27390f3b1



File View Navigation Tools Plugins Help

6b1179c23a7502b4dea7f9bd

Inputs  
Source code  
Resources  
APK signature  
Summary

AndroidManifest.xml x gvfvxihqcjdzxcoqrwcmfiwhafbgwumxrnwzktzlqesibdqov7\_AR

```
package corporate.scottish;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.admin.DeviceAdminReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.os.UserHandle;
import corporate.scottish.khcuinjrgkokolanhrvznizqzenzechcnrnwfffcnbaojjmziv2.bekqjmxqccbikfnrsqojehqufgkxbjkznqdmkfoessotedyr4.Notificationunugbmlbybvvfdhcjleocyotsiknotmsfmmxjdrlzhsigltd12Utils;
import corporate.scottish.khcuinjrgkokolanhrvznizqzenzechcnrnwfffcnbaojjmziv2.flyunugbmlbybvvfdhcjleocyotsiknotmsfmmxjdrlzhsigltd12activ;

/* loaded from: classes.dex */
public class gvfvxihqcjdzxcoqrwcmfiwhafbgwumxrnwzktzlqesibdqov7_AR extends DeviceAdminReceiver {
    private static final int NOTIFICATION_ID = 48152;
    String T2 = "peptemvzgfllozdnsddweicrmxywaqkjxuqlgskgkhkfnjptvbaiujgsiqxvfrozoglnzvyhboouqearbfmovptmhwhztodgivbajvyezsbxwqlrqjgdxtgvkqljgweqyxwjgdmowhdilbtznntrpdrlawutmpktblksmnoyztwdgizmhjnykjmfwfzcocoflgs
    String T3 = "peptemvzgfllozdnsddweicrmxywaqkjxuqlgskgkhkfnjptvbaiujgsiqxvfrozoglnzvyhboouqearbfmovptmhwhztodgivbajvyezsbxwqlrqjgdxtgvkqljgweqyxwjgdmowhdilbtznntrpdrlawutmpktblksmnoyztwdgizmhjnykjmfwfzcocoflgs
    String T4 = "peptemvzgfllozdnsddweicrmxywaqkjxuqlgskgkhkfnjptvbaujgsiqxvfrozoglnzvyhboouqearbfmovptmhwhztodgivbajvyezsbxwqlrqjgdxtgvkqljgweqyxwjgdmowhdilbtznntrpdrlawutmpktblksmnoyztwdgizmhjnykjmfwfzcocoflgs
    String T5 = "peptemvzgfllozdnsddweicrmxywaqkjxuqlgskgkhkfnjptvbaujgsiqxvfrozoglnzvyhboouqearbfmovptmhwhztodgivbajvyezsbxwqlrqjgdxtgvkqljgweqyxwjgdmowhdilbtznntrpdrlawutmpktblksmnoyztwdgizmhjnykjmfwfzcocoflgs
    Boolean willkxgddgyfyqhsqcaiorwfgstuyxusbyslcipqzqnbfoutuaia212 = false;

    private String getmet2(int i) {
        return "peptemvzgfllozdnsddweicrmxywaqkjxuqlgskgkhkfnjptvbaiujgsiqxvfrozoglnzvyhboouqearbfmovptmhwhztodgivbajvyezsbxwqlrqjgdxtgvkqljgweqyxwjgdmowhdilbtznntrpdrlawutmpktblksmnoyztwdgizmhjnykjmfwfzcocoflgs
    }

    @Override // android.app.admin.DeviceAdminReceiver
    public void onProfileProvisioningComplete(Context context, Intent intent) {
        super.onProfileProvisioningComplete(context, intent);
    }

    @Override // android.app.admin.DeviceAdminReceiver, android.content.BroadcastReceiver
    public void onReceive(Context context, Intent intent) {
        super.onReceive(context, intent);
        if (Build.VERSION.SDK_INT >= 23) {
            if (Build.VERSION.SDK_INT >= 26) {
                return;
            }
            Intent intent2 = new Intent(context, (Class<?>) flyunugbmlbybvvfdhcjleocyotsiknotmsfmmxjdrlzhsigltd12activ.class);
            intent2.setFlags(268435456);
            Notification createNotification = Notificationunugbmlbybvvfdhcjleocyotsiknotmsfmmxjdrlzhsigltd12Utils.getInstance(context).createNotification(context);
            createNotification.contentIntent = PendingIntent.getActivity(context, 0, intent2, 201326592);
        }
    }
}
```

Source: <https://bazaar.abuse.ch/>



## The path to the unknown

Where the mage finds a powerful  
source of power

# Introducing Frida

- Frida is a powerful open-source tool which was built focusing on dynamic analysis of applications
- It can be used for **Android, iOS, Windows, Linux, macOS** and a couple of other types of applications



**FRIDA**

[OVERVIEW](#) [DOCS](#) [NEWS](#) [CODE](#) [CONTACT](#)

Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers.

New apps with encrypted network protocols

Black-box testing

Discover features of the app by interacting with it

Obfuscated code

# Playing hide and seek

- Making code hard to read -> still functional
- Protects network domains, prevents reverse engineering, and enhances security

## How exactly is this possible?

- Renaming variables and functions
- Control flow alteration
- String encryption
- Dead code insertion



### Original code

```
int i;

void write_char(char ch)
{
    printf("%c", ch);
}

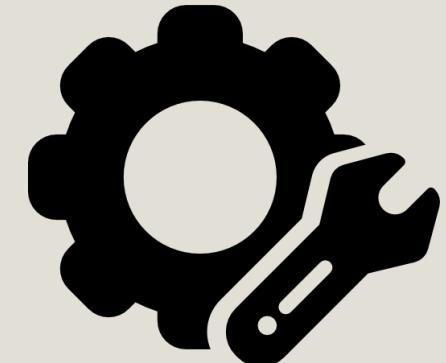
int main()
{
    for (i = 0; i < 15; i++) {
        write_char("hello, world!\n"[i]);
    }
    return 0;
}
```

### Code after obfuscation techniques

```
int i;main(){for(i=0;i["]<i;++i){--i;}"];
read('----',i++,"hell\
o,world!\n",'//'));}read(j,i,p){
write(j/p+p,i---j,i/i);}
```

# Setting up

- Download **Frida** for your OS: <https://frida.re/docs/installation/>
- You will also need to download **frida-server** for your device: <https://github.com/frida/frida/releases>
- Example on Linux with a **rooted** device:  
`pip install frida-tools  
unzip <path>/frida-server.zip  
adb root  
adb push <path>/frida-server.zip /data/local/tmp/frida-server  
adb shell chmod +x /data/local/tmp/frida-server  
adb shell /data/local/tmp/frida-server`



Done! *Start hacking!*

# Main features

## frida-ps

### List of processes of the device



| L-\$ frida-ps -Uai | PID  | Name            | Identifier                              |
|--------------------|------|-----------------|---|
|                    | 3212 | APK DONE        | corporate.scottish.km                   |
|                    | 4108 | Calendar        | com.google.android.calendar             |
|                    | 4244 | Clock           | com.google.android.deskclock            |
|                    | 4273 | Gmail           | com.google.android.gm                   |
|                    | 2600 | Google          | com.google.android.googlequicksearchbox |
|                    | 2600 | Google          | com.google.android.googlequicksearchbox |
|                    | 3823 | Maps            | com.google.android.apps.maps            |
|                    | 2421 | Messages        | com.google.android.apps.messaging       |
|                    | 2213 | Phone           | com.google.android.dialer               |
|                    | 3944 | Photos          | com.google.android.apps.photos          |
|                    | 1867 | SIM Toolkit     | com.android.stk                         |
|                    | 5194 | Settings        | com.android.settings                    |
|                    | 4742 | YouTube         | com.google.android.youtube              |
|                    | 5001 | YouTube Music   | com.google.android.apps.youtube.music   |
|                    | -    | - Camera        | com.android.camera2                     |
|                    | -    | - Chrome        | com.android.chrome                      |
|                    | -    | - Contacts      | com.android.contacts                    |
|                    | -    | - Drive         | com.google.android.apps.docs            |
|                    | -    | - Files         | com.google.android.documentsui          |
|                    | -    | - WebView Shell | org.chromium.webview_shell              |

# Main features

## frida-trace

### Trace function calls



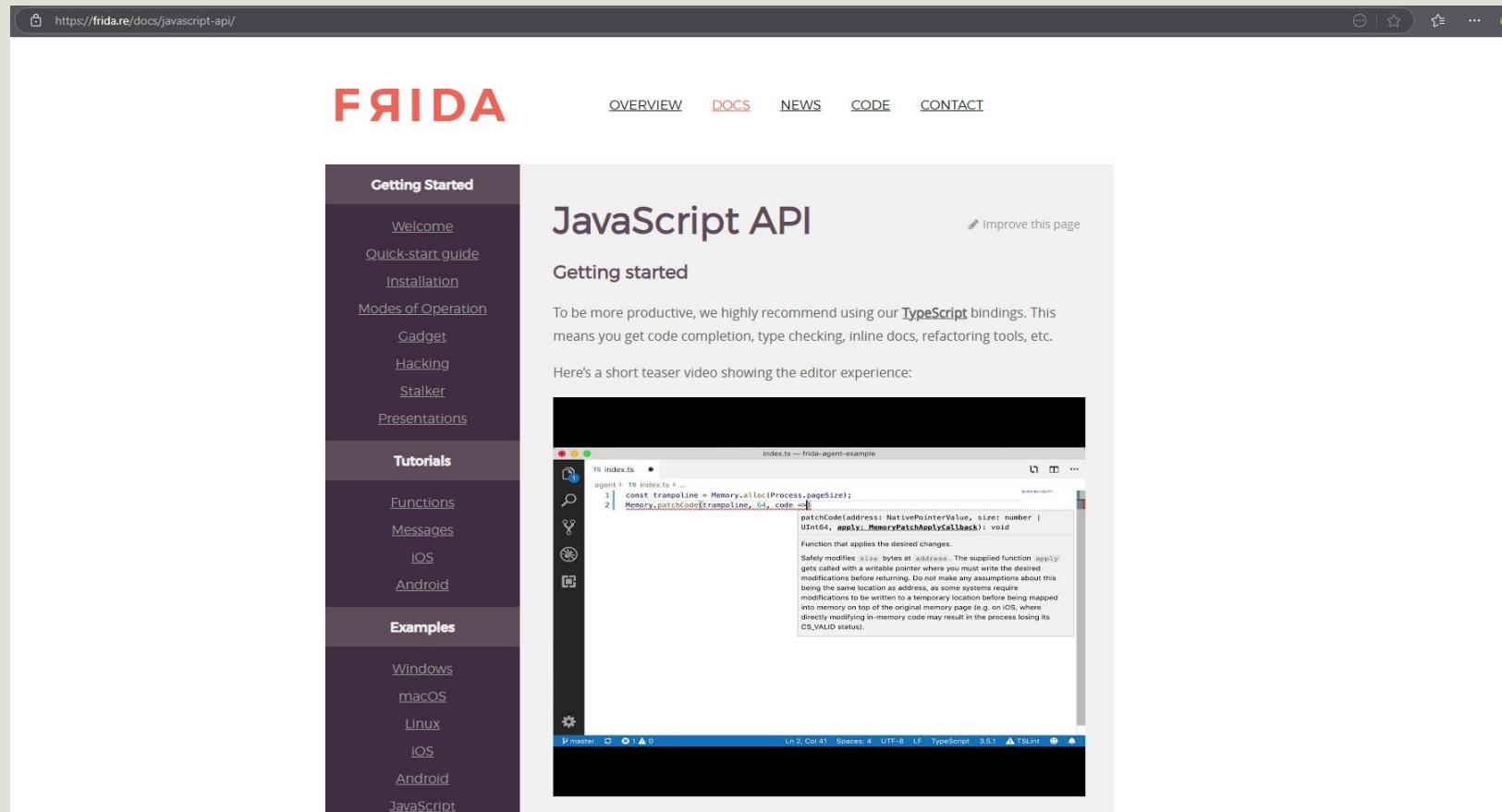
```
$ frida-trace -D emulator-5554 -f "corporate.scottish.km" -i "*decrypt*"  
Instrumenting ...  
_ZN7android9CryptoHal7decryptEPKhS2_NS_12CryptoPlugin4ModeERKNS3_7PatternERK  
_handlers__/_libmediadrm.so/_ZN7android9CryptoHal7decryptEPK_d1607552.js"  
_ZN7android16DrmMetricsLogger7decryptERKNS_6VectorIhEES4_S4_S4_RS2_: Auto-ge  
_ZN7android6DrmHal7decryptERKNS_6VectorIhEES4_S4_S4_RS2_: Auto-generated han  
_ZN7android10DrmHalAidl7decryptERKNS_6VectorIhEES4_S4_S4_RS2_: Auto-generate  
_ZN7android13CryptoHalAidl7decryptEPKhS2_NS_12CryptoPlugin4ModeERKNS3_7Patte  
bin/_handlers__/_libmediadrm.so/_ZN7android13CryptoHalAidl7decry_71789995.js"  
_ZN7android10DrmHalHidl7decryptERKNS_6VectorIhEES4_S4_S4_RS2_: Auto-generate  
_ZN7android13CryptoHalHidl7decryptEPKhS2_NS_12CryptoPlugin4ModeERKNS3_7Patte  
bin/_handlers__/_libmediadrm.so/_ZN7android13CryptoHalHidl7decry_2a6c01e5.js"  
_ZN4aidl7android8hardware3drm11BpDrmPlugin7decryptERKNSt3__16vectorIhNS4_9al  
3drm11B_a3e523e4.js"  
_ZN4aidl7android8hardware3drm17IDrmPluginDefault7decryptERKNSt3__16vectorIhN  
rdware3drm17I_7b8bade0.js"  
_ZN4aidl7android8hardware3drm20ICryptoPluginDefault7decryptERKNS2_11DecryptA  
_ZN4aidl7android8hardware3drm14BpCryptoPlugin7decryptERKNS2_11DecryptArgsEPi  
_ZN7android8hardware3drm4V1_113BpHwDrmPlugin7decryptERKNS0_8hidl_vecIhEES7_S
```

*More examples **HERE:** <https://codeshare.frida.re/>*

# Main features

## Javascript API

The important Javascript API's to do some hacking, interact with some methods and inspect what application classes do: <https://frida.re/docs/javascript-api>



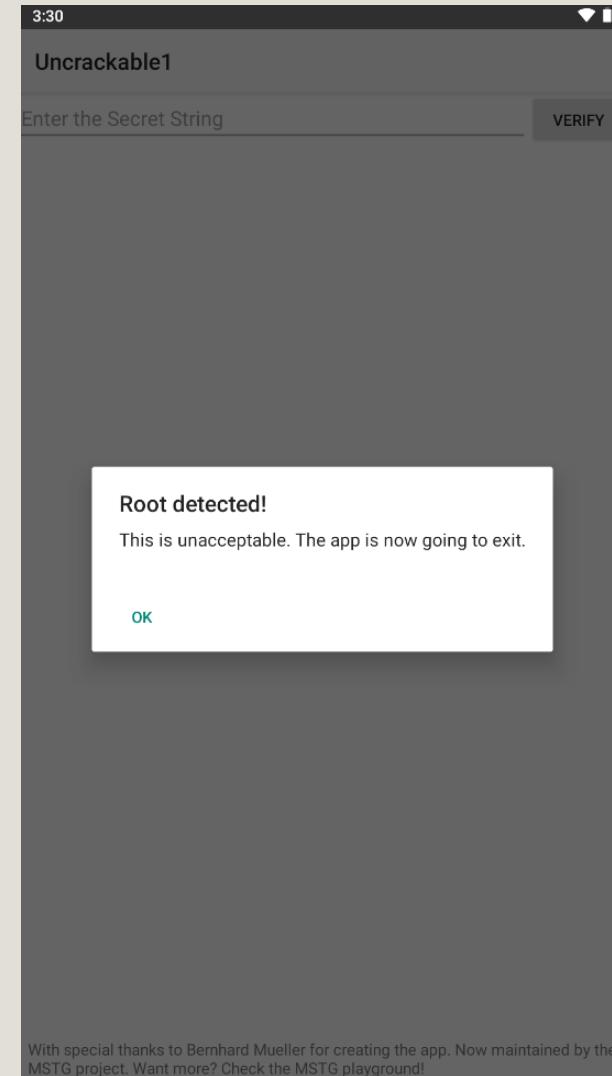
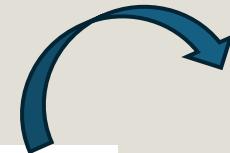
# A simple use scenario

- OWASP – OWASP Mobile Application Security

## MAS Crackmes

Welcome to the MAS Crackmes aka. UnCrackable Apps, a collection of mobile reverse engineering challenges. These challenges are used as examples throughout the OWASP MASTG. Of course, you can also solve them for fun.

|  |                          |  |                          |
|--|--------------------------|--|--------------------------|
|  Android UnCrackable L1<br>UnCrackable-Level1.apk   | <a href="#">Download</a> |  iOS UnCrackable L1<br>UnCrackable-Level1.ipa | <a href="#">Download</a> |
|  Android UnCrackable L2<br>UnCrackable-Level2.apk   | <a href="#">Download</a> |  iOS UnCrackable L2<br>UnCrackable-Level2.ipa | <a href="#">Download</a> |
|  Android UnCrackable L3<br>UnCrackable-Level3.apk   | <a href="#">Download</a> |  |                          |
|  Android UnCrackable L4<br>r2pay-v0.9.apk          | <a href="#">Download</a> |  |                          |
|  Android UnCrackable DRM<br>validate (ELF 32-bit) | <a href="#">Download</a> |  |                          |

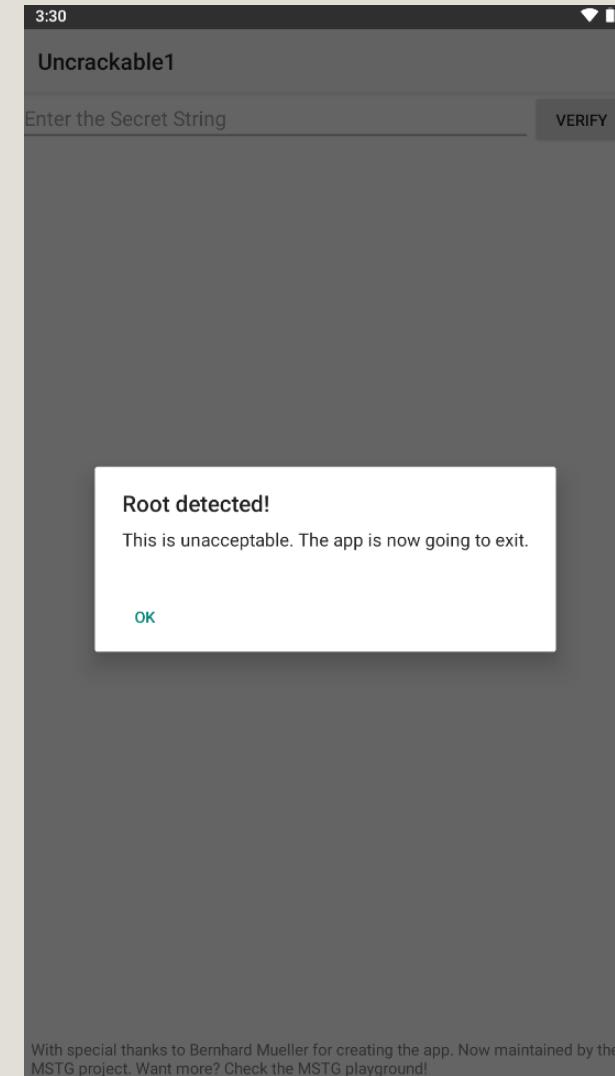


With special thanks to Bernhard Mueller for creating the app. Now maintained by the MSTG project. Want more? Check the MSTG playground!

# A simple use scenario

- OWASP – OWASP Mobile Application Security

Can we bypass  
a situation like this?

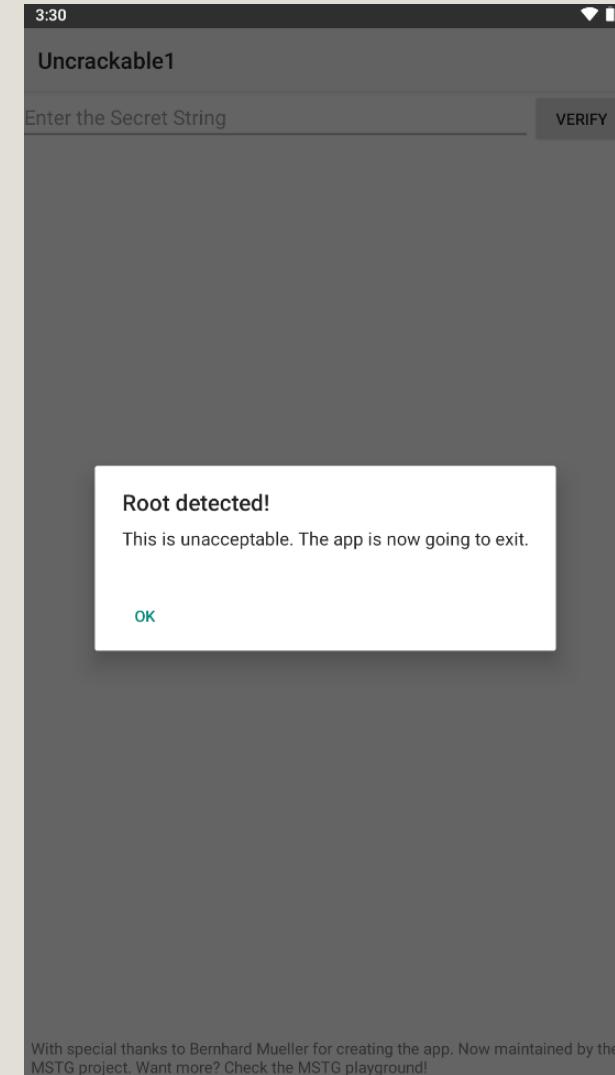


# A simple use scenario

- OWASP – OWASP Mobile Application Security

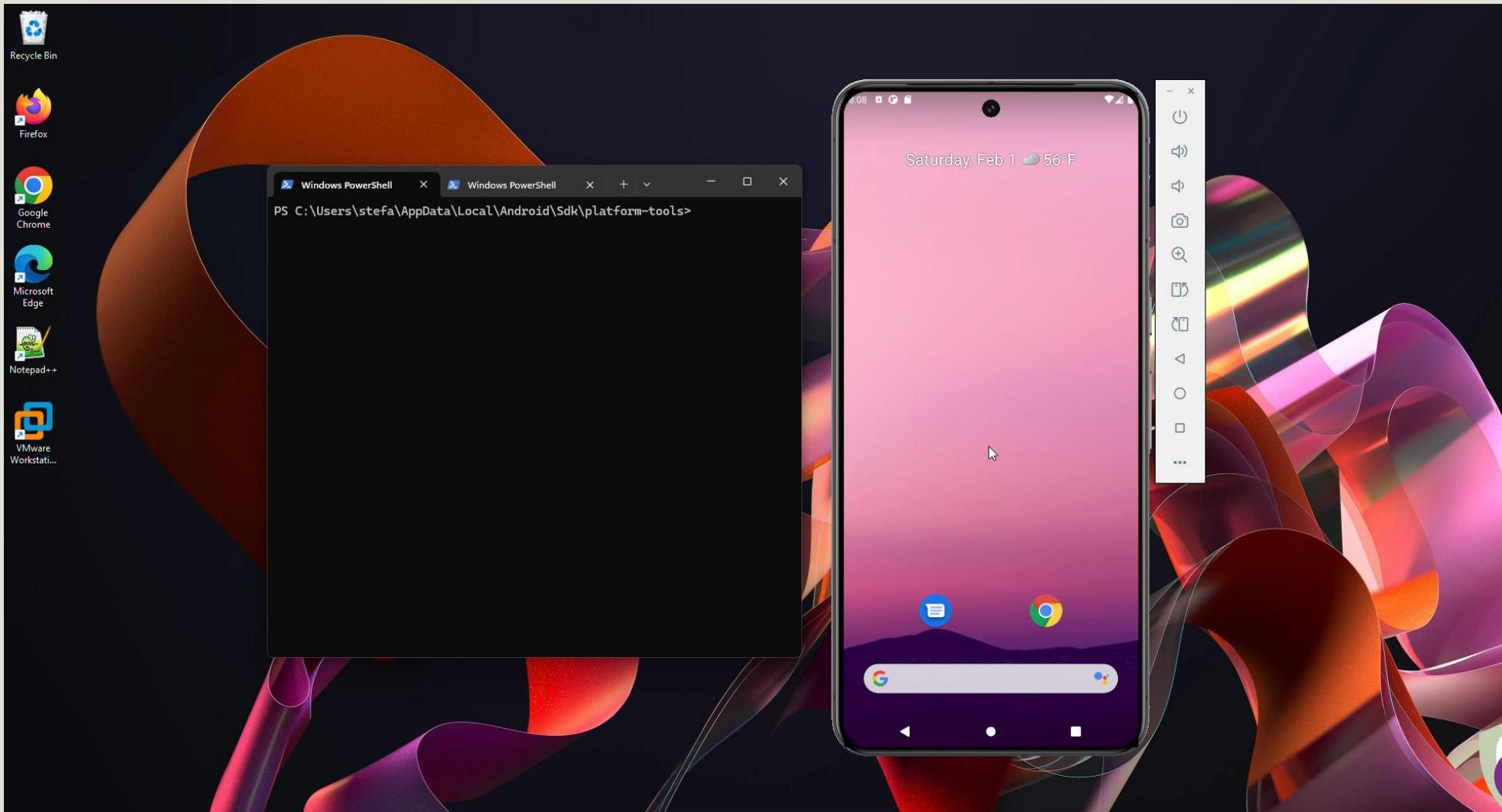
Can we bypass  
a situation like this?

YES! We can 😊



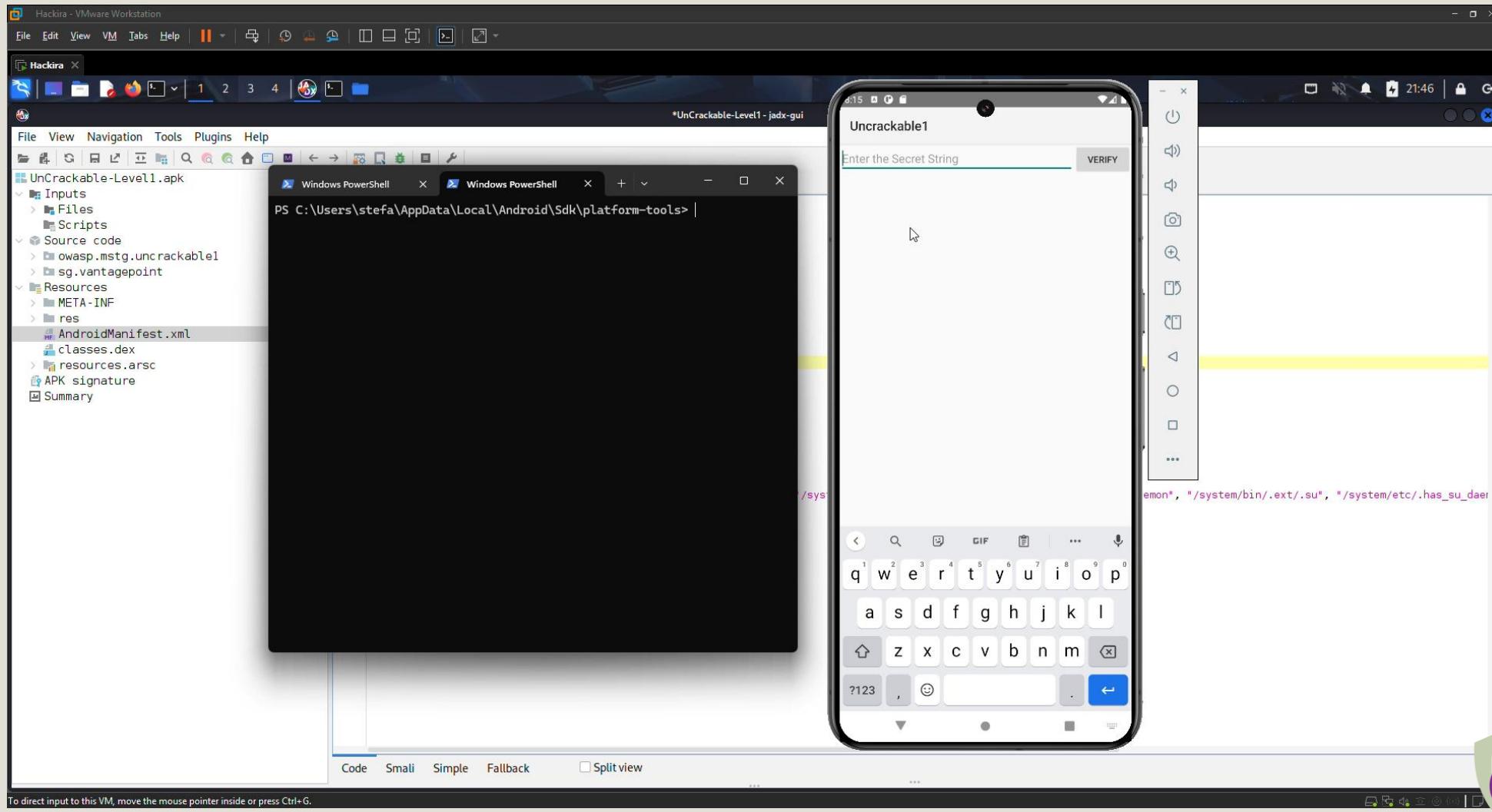
# A simple use scenario

- OWASP – OWASP Mobile Application Security: *Bypassing the first obstacle*



# A simple use scenario

- OWASP – OWASP Mobile Application Security: *Bypassing the second obstacle*



# Safe and sound?

When developing or researching...

- ***Build your arsenal.***  
Static and dynamic analysis tools  
(decompilers, automated scripts, simulated environments)
- ***Have a strategy plan.***  
Order of analysis (based on number and types of components, time and complexity)
- ***Dive into the analysis.***  
Check how the used protocols work, features flows of the application, cryptography characteristics
- ***Take notes.***  
Every discover of a flow or feature of a path, document the process
- ***Take another path.***  
Try to search for a new perspective in the analysis to cover more possibilities



 Semgrep

 drozer





## Level up

An **obstacle** lies in the mage's path  
to the **golden treasure**

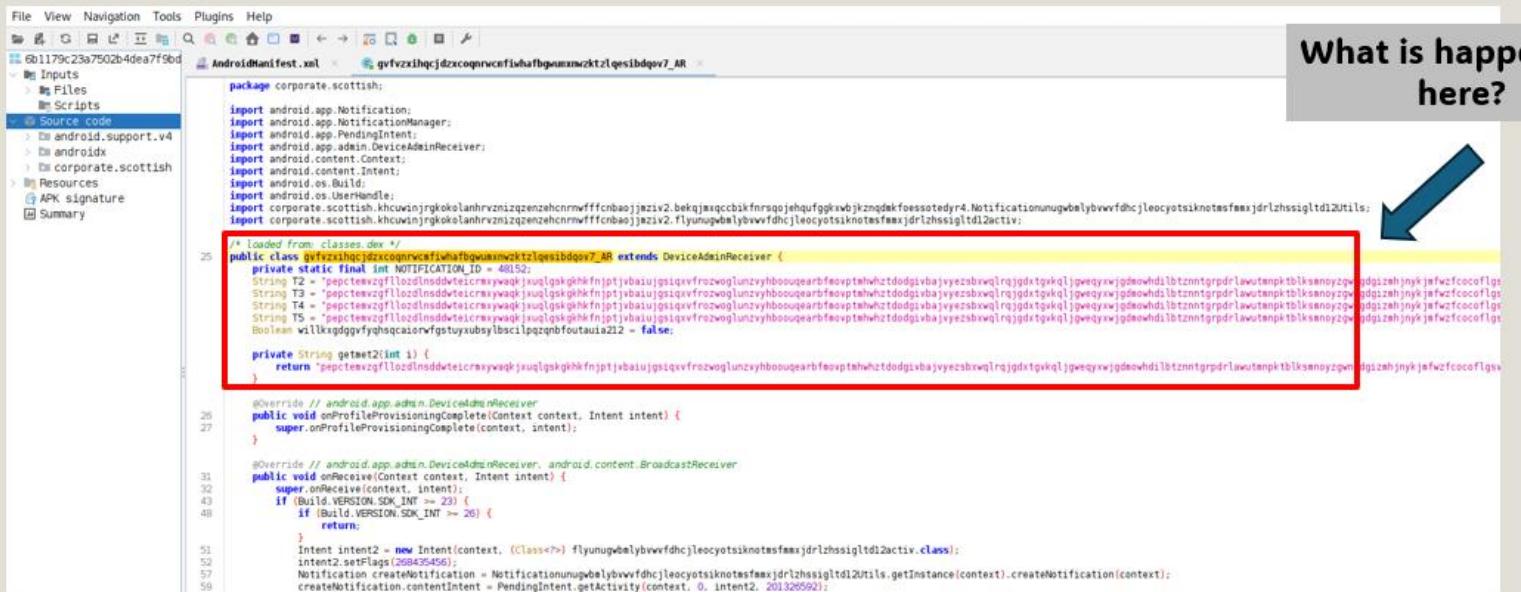
# Let's analyze something interesting...



Previously on this presentation...

## Let's take a look at an example!

The reference sample (SHA-256):



What is happening here?

```
File View Navigation Tools Plugins Help
AndroidManifest.xml gfvzxihqjdzxcopnrcefisahfbgnmmwktz1qesibdqv7_AR
Source code Inputs Files Scripts
Source code
android.support.v4
androidx
corporate.scottish
APK signature Summary
File View Navigation Tools Plugins Help
AndroidManifest.xml gfvzxihqjdzxcopnrcefisahfbgnmmwktz1qesibdqv7_AR
Source code Inputs Files Scripts
Source code
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.admin.DeviceAdminReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import corporate.scottish.khcuinjrgrkokolanhrvznizqzenzeahnrmwffcnbaejjzziv2.bekgjmxcccbikfnrsqqjeahufgkwbjkzndkfessotedyr4.Notificationunugblybvvfdhcjleocytisknotesfmxjdrlhssigltd12activ;
import corporate.scottish.khcuinjrgrkokolanhrvznizqzenzeahnrmwffcnbaejjzziv2.flyunugblybvvfdhcjleocytisknotesfmxjdrlhssigltd12activ;
*/
public class gfvzxihqjdzxcopnrcefisahfbgnmmwktz1qesibdqv7_AR extends DeviceAdminReceiver {
    private static final int NOTIFICATION_ID = 48125;
    String T2 = "peptevszfllozdlnsdwteicraxywakjxuqlgskgkhfnjptjbaiujsqsvfrzoglnvhboosuerbarfeuptehctddogisbjyezbwqlnsjgdtpkqljgveoyxjpdewhdilbtznntrprdlawutmpktblksanoyzwgjdgzahjnjkjfwtfcocofigs";
    String T3 = "peptevszfllozdlnsdwteicraxywakjxuqlgskgkhfnjptjbaiujsqsvfrzoglnvhboosuerbarfeuptehctddogisbjyezbwqlnsjgdtpkqljgveoyxjpdewhdilbtznntrprdlawutmpktblksanoyzwgjdgzahjnjkjfwtfcocofigs";
    String T4 = "peptevszfllozdlnsdwteicraxywakjxuqlgskgkhfnjptjbaiujsqsvfrzoglnvhboosuerbarfeuptehctddogisbjyezbwqlnsjgdtpkqljgveoyxjpdewhdilbtznntrprdlawutmpktblksanoyzwgjdgzahjnjkjfwtfcocofigs";
    String T5 = "peptevszfllozdlnsdwteicraxywakjxuqlgskgkhfnjptjbaiujsqsvfrzoglnvhboosuerbarfeuptehctddogisbjyezbwqlnsjgdtpkqljgveoyxjpdewhdilbtznntrprdlawutmpktblksanoyzwgjdgzahjnjkjfwtfcocofigs";
    Boolean willkgdggfyjhqcaierwfgtuyubscilpgzqnfoutaus212 = false;
    private String getmet2(int i) {
        return "peptevszfllozdlnsdwteicraxywakjxuqlgskgkhfnjptjbaiujsqsvfrzoglnvhboosuerbarfeuptehctddogisbjyezbwqlnsjgdtpkqljgveoyxjpdewhdilbtznntrprdlawutmpktblksanoyzwgjdgzahjnjkjfwtfcocofigs";
    }
    @Override // android.app.admin.DeviceAdminReceiver
    public void onProfileProvisioningCompleted(Context context, Intent intent) {
        super.onProfileProvisioningCompleted(context, intent);
    }
    @Override // android.app.admin.DeviceAdminReceiver, android.content.BroadcastReceiver
    public void onReceive(Context context, Intent intent) {
        super.onReceive(context, intent);
        if (Build.VERSION.SDK_INT >= 23) {
            if (Build.VERSION.SDK_INT >= 26) {
                return;
            }
            Intent intent2 = new Intent(context, (Class<?>) flyunugblybvvfdhcjleocytisknotesfmxjdrlhssigltd12activ.class);
            intent2.putExtra("208425456");
            Notification createNotification = Notificationunugblybvvfdhcjleocytisknotesfmxjdrlhssigltd12activ.getInstance(context).createNotification(context);
            createNotification.setContentIntent = PendingIntent.getActivity(context, 0, intent2, 20320552);
        }
    }
}
```

Source: <https://bazaar.abuse.ch/>



# *Let's analyze something interesting...*



**Visa Secure campaign  
using *fake VISA*  
*app Distribution***

[https://visasecurity\[.\]net](https://visasecurity[.]net)  
(already reported and taken down)

Alberto Segura  
@alberto\_segura

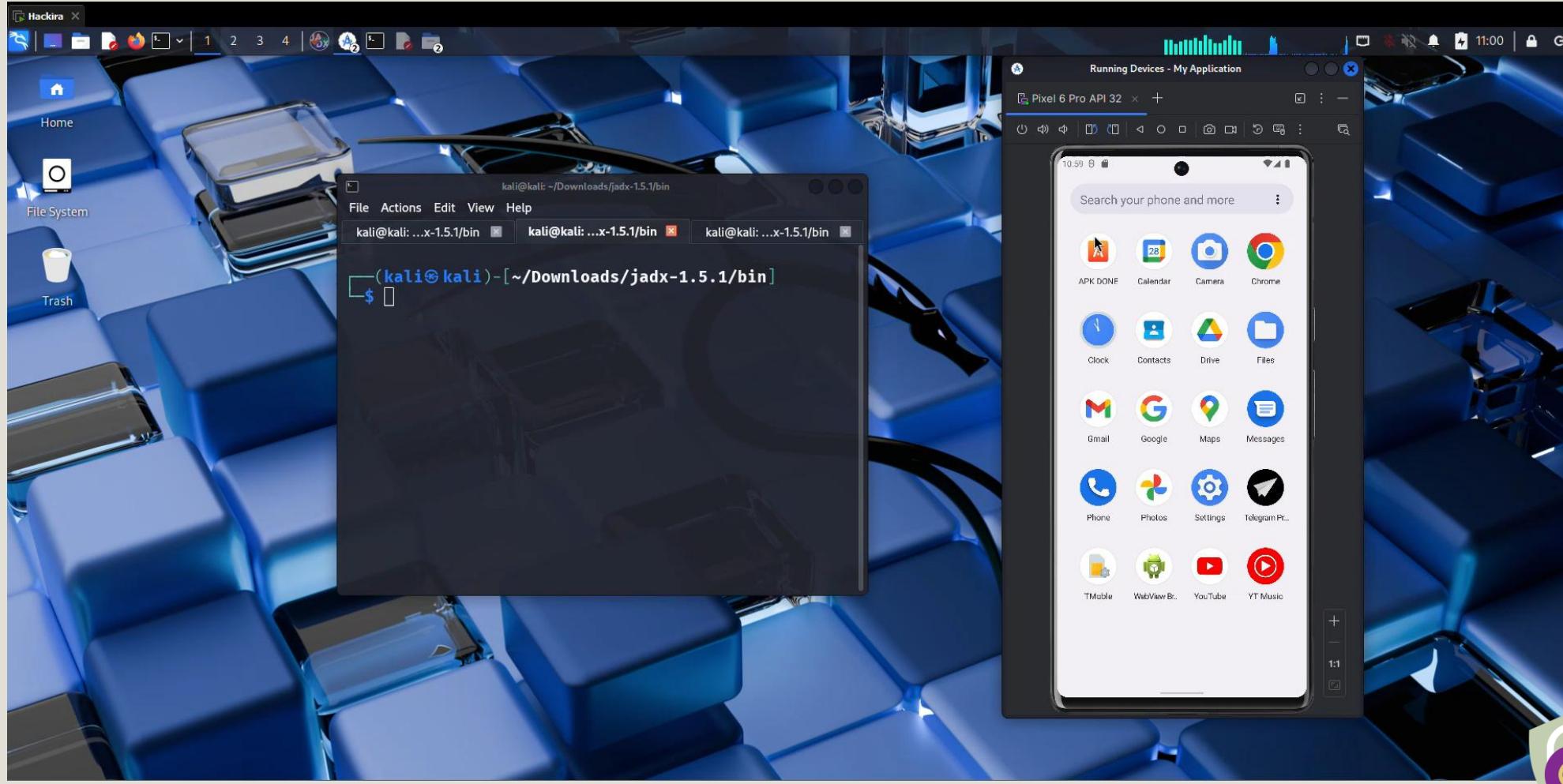
#Spynote campaign using fake VISA app  
Distribution: [https://visasecurity\[.\]net/](https://visasecurity[.]net/)

C2: 172.86.93.104:7771  
hash:  
6b1179c23a7502b4dea7f9bde7dde3d4b5b97c64f634ff3471a1d3d27390  
f3b1  
(The PC download button does nothing)

A screenshot of a web browser showing a fake VISA app distribution page. The page features the VISA logo and the text 'Protect your device from fraud with VISA SECURE APP'. It includes a 'Download For Android' button and a list of features: 'Block malware - Stop viruses, ransomware, and spyware from attacking your device.', 'Real - time protection - Get instant defense against threats while browsing or downloading.', 'Guard your data - Keep hackers from stealing passwords, bank info, and private files.', 'Boost performance - Prevent slowdowns by removing harmful software.', and 'Browse safely - Avoid phishing scams and dangerous websites.' At the bottom, it says 'Please contact VISA SECURITY support immediately to confirm your recent activity.'

# *Let's analyze something interesting...*

- Malware (Spyware): Visa Secure.apk
- Sample analyzed in: 03/26/2025



# *Let's analyze something interesting...*



Since we are in an emulated device, **LET'S JUST CHANGE IT FOR A REAL DEVICE** and continue our analysis!

# *Let's analyze something interesting...*



Since we are in an emulated device, **LET'S JUST CHANGE IT FOR A REAL DEVICE** and continue our analysis!



# *Let's analyze something interesting...*



**NEVER** download malware on your device. If you suspect it is one and want to make sure, analyze it on an emulator!



# Let's analyze something interesting...

- Malware (Spyware): Visa Secure.apk
- Sample analyzed in: 03/26/2025

The screenshot shows a desktop environment with a dark-themed window manager. In the center, there is a Java decompiler window titled "Hackira" with the file path "-/Downloads/bypassEmulator.js - Mousepad". The code is as follows:

```
1 Java.perform({
2     var isEmulator = true;
3     if (!isEmulator) {
4         isEmulator = false;
5     }
6     if (isEmulator) {
7         //var isEmulator = true;
8         isEmulator = false;
9     }
10    /*var isEmulator = true;
11    isEmulator = false;
12 */
13    if (isEmulator) {
14        //var isEmulator = true;
15        isEmulator = false;
16    }
17    if (isEmulator) {
18        //var isEmulator = true;
19        isEmulator = false;
20    }*/
21 });
22
```

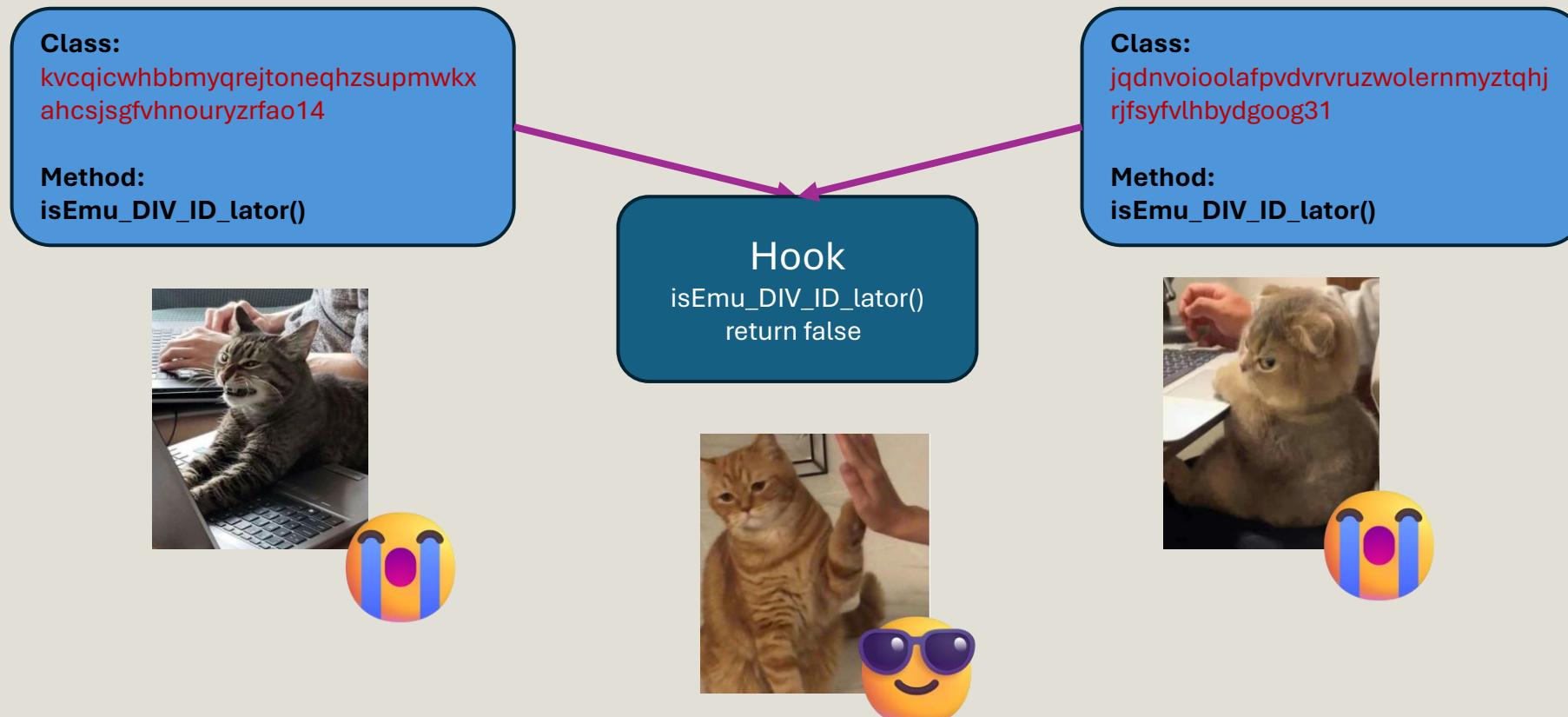
Below the decompiler, there are tabs for "Code", "Smali", "Simple", "Fallback", and "Split view". The "Code" tab is selected.

To the right of the decompiler is a mobile application window titled "Running Devices - My Application" showing a "Pixel 6 Pro API 32" device. The device screen displays a standard Android home screen with various app icons like APK DRAFT, Calendar, Camera, Chrome, Clock, Contacts, Drive, Files, Gmail, Google, Maps, Messages, Phone, Photos, Settings, Telegram, TMobile, WebView B..., YouTube, and YT Music.

At the bottom left of the desktop, there is a small note: "To return to your computer, move the mouse pointer outside or press Ctrl+Alt."

# *Let's analyze something interesting...*

- Malware (Spyware): Visa Secure.apk
- Sample analyzed in: 03/26/2025



# *Let's analyze something interesting...*

- **Malware (Spyware): Visa Secure.apk**
- **Sample analyzed in: 03/26/2025**

| Aspect              | Value                               |
|---------------------|-------------------------------------|
| <b>Fingerprint</b>  | Generic                             |
| <b>Model</b>        | Emulator, Android SDK built for x86 |
| <b>Brand</b>        | generic_x86                         |
| <b>Hardware</b>     | Goldfish                            |
| <b>Manufacturer</b> | Unknown, Genymotion                 |
| <b>Contains</b>     | Emulator, simulator                 |

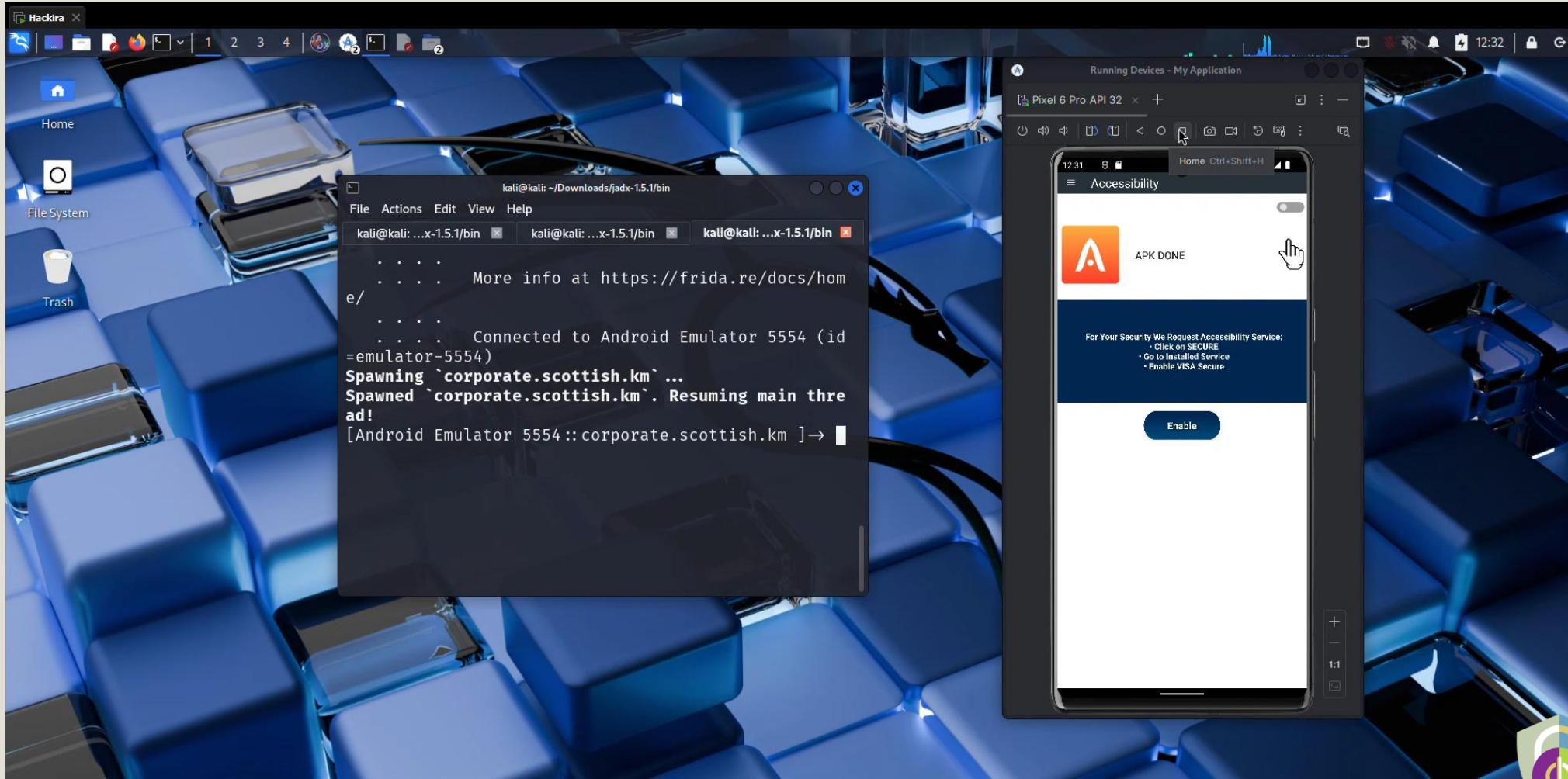
# *Let's analyze something interesting...*

- **Malware (Spyware): Visa Secure.apk**
- **Sample analyzed in: 03/26/2025**

```
1 Java.perform(function() {
2     var isEmulator =
3         Java.use("corporate.scottish.khcuwinjrgkokolanhrvznizqzenzehcnrnwfffcbaojjmziv2.kvcqicwhbbmyqrejtoneqhzsupmwkxahcsjsgfvhnouryzrfa014
4 ");
5     isEmulator["isEmu_DIV_ID_lator"].implementation = function() {
6         return false;
7     };
8     isEmulator["AsknoEmu"].implementation = function() {
9         console.log("I AM HERE!!!");
10    }
11
12    var isEmulator2 =
13        Java.use("corporate.scottish.khcuwinjrgkokolanhrvznizqzenzehcnrnwfffcbaojjmziv2.jqdnvoioolafpvdrvruzwlernmyztqhjrjfsyfvlhbydgoog31
14 ");
15    isEmulator2["isEmu_DIV_ID_lator"].implementation = function() {
16        return false;
17    };
18    isEmulator2["AsknoEmu"].implementation = function() {
19        console.log("I AM HERE!!!");
20    }
21});
```

# *Let's analyze something interesting...*

- But what the app seems to be doing anyway?



# Why does it want to access *accessibilities*?

- Features that make an app usable by people with disabilities (e.g., screen readers, high-contrast modes, touch accommodations)
- Examples: Voice Access, TalkBack, magnification gestures)

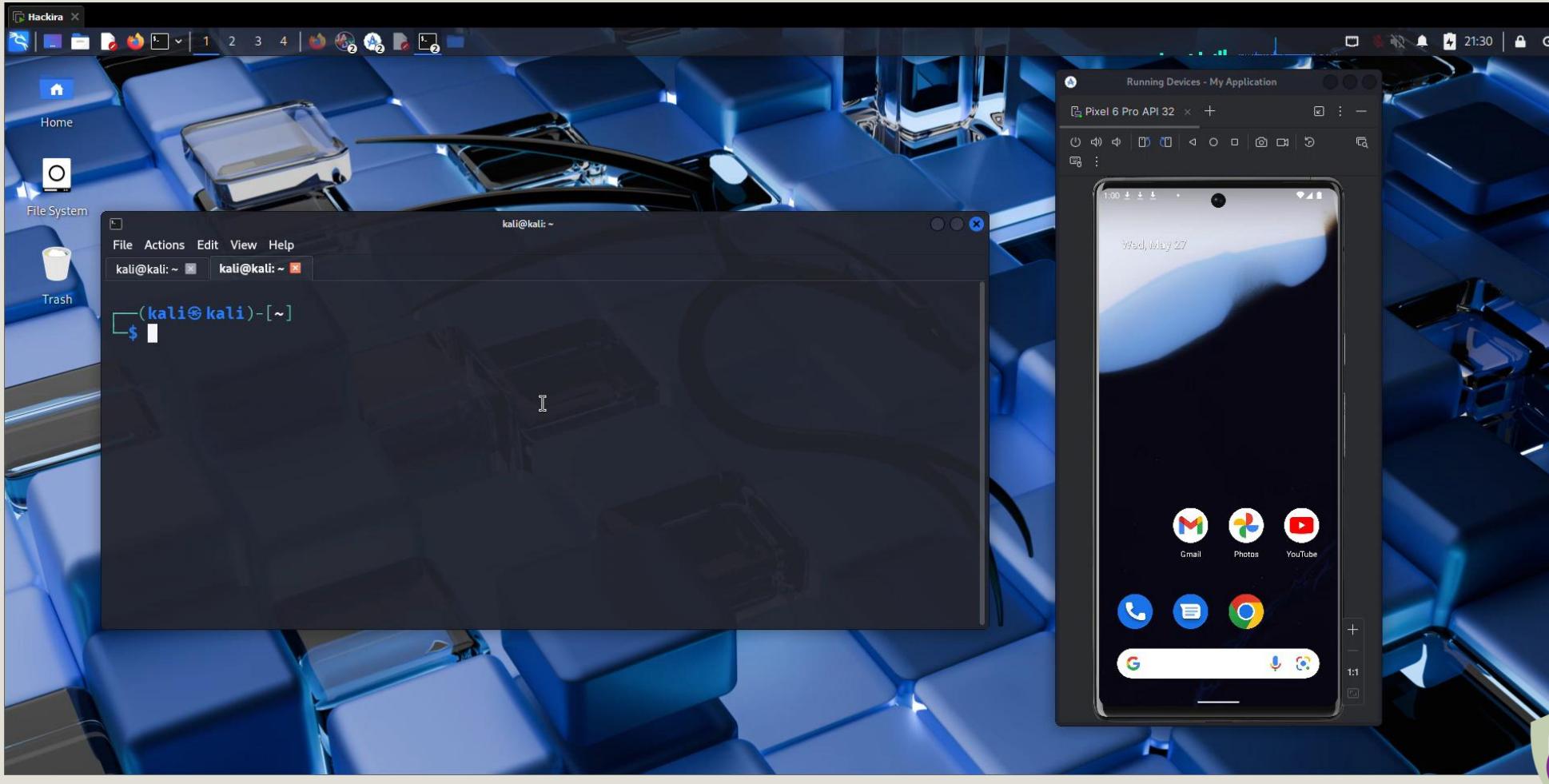
## Why would a **malware** want that then?

- It can allow applications to click buttons and enter text without user interaction
- Capture keystrokes and intercept sensitive information (e.g., passwords, OTPs)
- Malware can create fake overlays on legitimate apps (e.g., banking apps) to steal user credentials and other sensitive data and monitor all user's activities



# *Let's analyze something interesting...*

- Since it seems to be an apk to download other files, let's try to see if there are any sockets trying to be accessed by the application...



# Let's analyze something interesting...

- The **MAGIC** happens when you combine **DYNAMIC + STATIC** analysis!

The screenshot shows a web-based static analysis interface. On the left, the navigation bar includes links to Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. The main interface has two main panes: one for file analysis and one for code review.

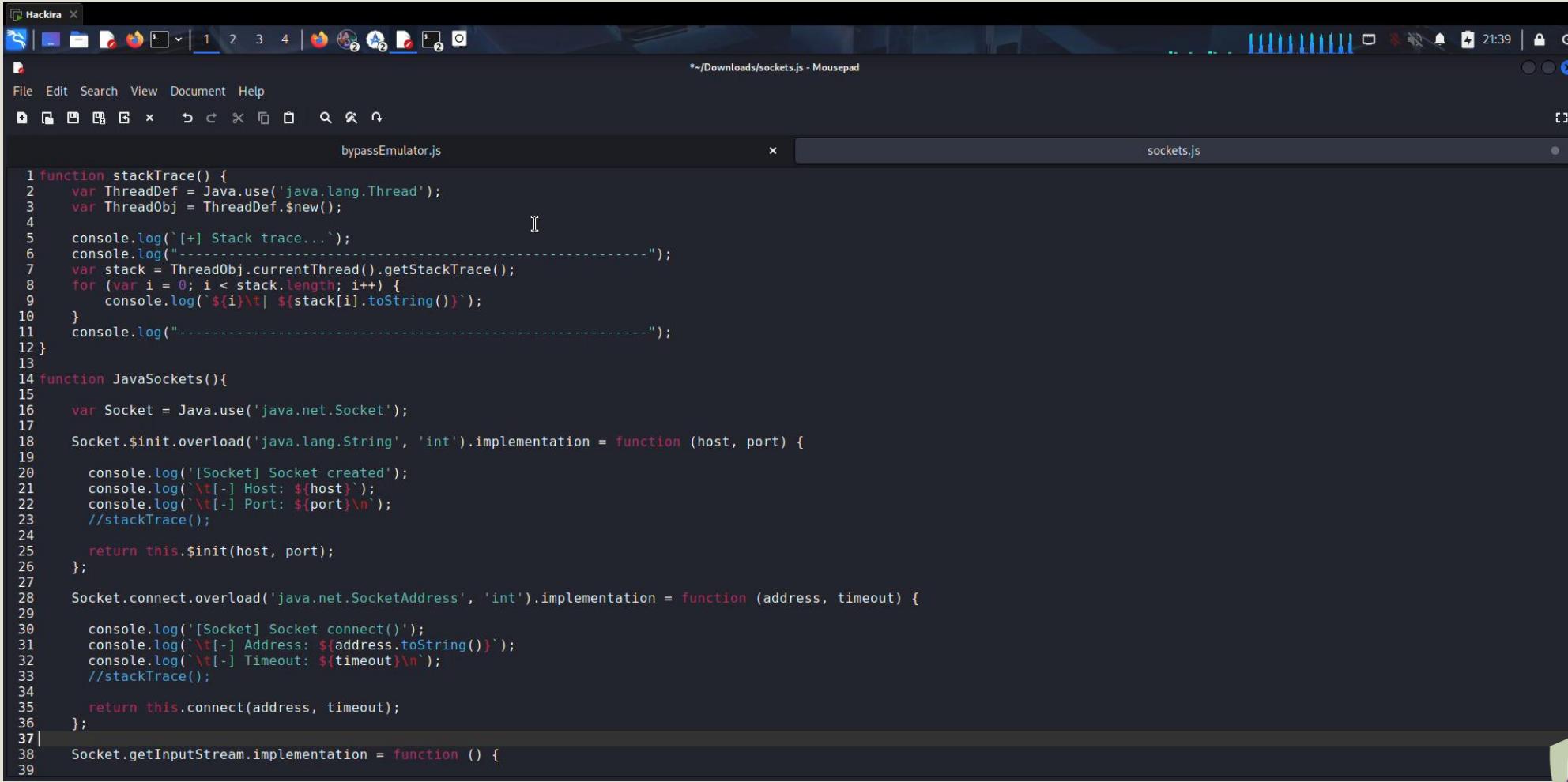
**File Analysis:** Shows a file tree for a Java application. Key files include KEY.RSA, KEY.SF, MANIFEST.MF, and several XML files under res/values (arrays.xml, attrs.xml, colors.xml, dimens.xml, drawables.xml, integers.xml, strings.xml). The strings.xml file contains the string "mrzalmcysowcmcm1tmjfgiyuseqgctxzctwidlniquktuvfdmh38.oconqfyqqmuqv71Service".

**Code Review:** Displays the Java code for a service named VpnService.Builder. The code includes methods for getting local IP addresses and adding routes to a VPN builder. A specific line of code is highlighted: `VPNbuilder.addAddress(getApplicationContext(), 24);`. The code also checks for loopback addresses and adds DNS servers.

**Bottom Navigation:** Includes buttons for socket, Highlight All, Match Case, Match Diacritics, Whole Words, and Phrase not found.

# Let's analyze something interesting...

- Show me the code!



```
bypassEmulator.js
1 function stackTrace() {
2     var ThreadDef = Java.use('java.lang.Thread');
3     var ThreadObj = ThreadDef.$new();
4
5     console.log(`[+] Stack trace...`);
6     console.log(`-----`);
7     var stack = ThreadObj.currentThread().getStackTrace();
8     for (var i = 0; i < stack.length; i++) {
9         console.log(`${i}\t| ${stack[i].toString()}`);
10    }
11    console.log(`-----`);
12 }
13
14 function JavaSockets(){
15
16     var Socket = Java.use('java.net.Socket');
17
18     Socket.$init.overload('java.lang.String', 'int').implementation = function (host, port) {
19
20         console.log('[Socket] Socket created');
21         console.log(`\t[-] Host: ${host}`);
22         console.log(`\t[-] Port: ${port}\n`);
23         //stackTrace();
24
25         return this.$init(host, port);
26     };
27
28     Socket.connect.overload('java.netSocketAddress', 'int').implementation = function (address, timeout) {
29
30         console.log('[Socket] Socket connect()');
31         console.log(`\t[-] Address: ${address.toString()}`);
32         console.log(`\t[-] Timeout: ${timeout}\n`);
33         //stackTrace();
34
35         return this.connect(address, timeout);
36     };
37
38     Socket.getInputStream.implementation = function () {
39 }
```

```
sockets.js
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

Source: TFG-Frida-Scripts/Network and Traffic/Sockets.js at main · ineesdv/TFG-Frida-Scripts · GitHub

# What about a *Brazilian flavor*?

The reference sample (SHA-256):

b14412435de7ebe13c434eef79a5c24dc2c48e5483d9db2aef30b569738f4b32

| MALWARE bazaar   |  |   |                  |          |                           |
|--|--|---|------------------|----------|---------------------------|
| <a href="#">from ABUSE.ch</a>    SPAMHAUS |  |   |                  |          |                           |
| Intelligence 8   | IOCs   | YARA  | File information | Comments | <a href="#">Actions ▾</a> |
| SHA256 hash:   | <a href="#"> b14412435de7ebe13c434eef79a5c24dc2c48e5483d9db2aef30b569738f4b32</a>                                 |   |                  |          |                           |
| SHA3-384 hash:   | <a href="#"> 67963e94655d19f48901381c9d9491d7702ac81420bc98b47877fabca443595acbc79fb31e8501d1afc0ed8b24af4576</a> |   |                  |          |                           |
| SHA1 hash:   | <a href="#"> e9d243692c386f9a7e311479acd59dc29f9cdb7</a>  |   |                  |          |                           |
| MD5 hash:  | <a href="#"> 24b78178404f1819c958b89e20c3cf5d</a>   |   |                  |          |                           |
| humanhash:   | <a href="#"> berlin-mississippi-lemon-massachusetts</a>   |   |                  |          |                           |
| File name:   | google_docs.apk  |   |                  |          |                           |
| Download:  | <a href="#"> download sample</a>  |   |                  |          |                           |
| Signature                                 | <a href="#"> BrasDex</a>  | <a href="#"> Alert ▾</a> |                  |          |                           |
| File size:   | 1'160'423 bytes  |   |                  |          |                           |
| First seen:  | 2023-01-20 21:10:08 UTC  |   |                  |          |                           |
| Last seen:   | Never  |   |                  |          |                           |
| File type:   |  apk  |   |                  |          |                           |
| MIME type:   | application/zip  |   |                  |          |                           |
| ssdeep                                  | <a href="#"> 24576:KyJuovoOyEyMr3pL/acLII2K0YBdSuPA/J:KquFyEp/aGI2XYBYus</a>                                    |   |                  |          |                           |
| TLSH                                    | <a href="#"> T14D358BC6BA95FD93C835C271C586C56B33ABE6240B4397B761048738286FB678F416CB</a>                       |   |                  |          |                           |
| TrID                                    | 67.5% (.APK) Android Package (38500/1/9)   |   |                  |          |                           |

# *What about a **Brazilian flavor?***

**About the campaign:**

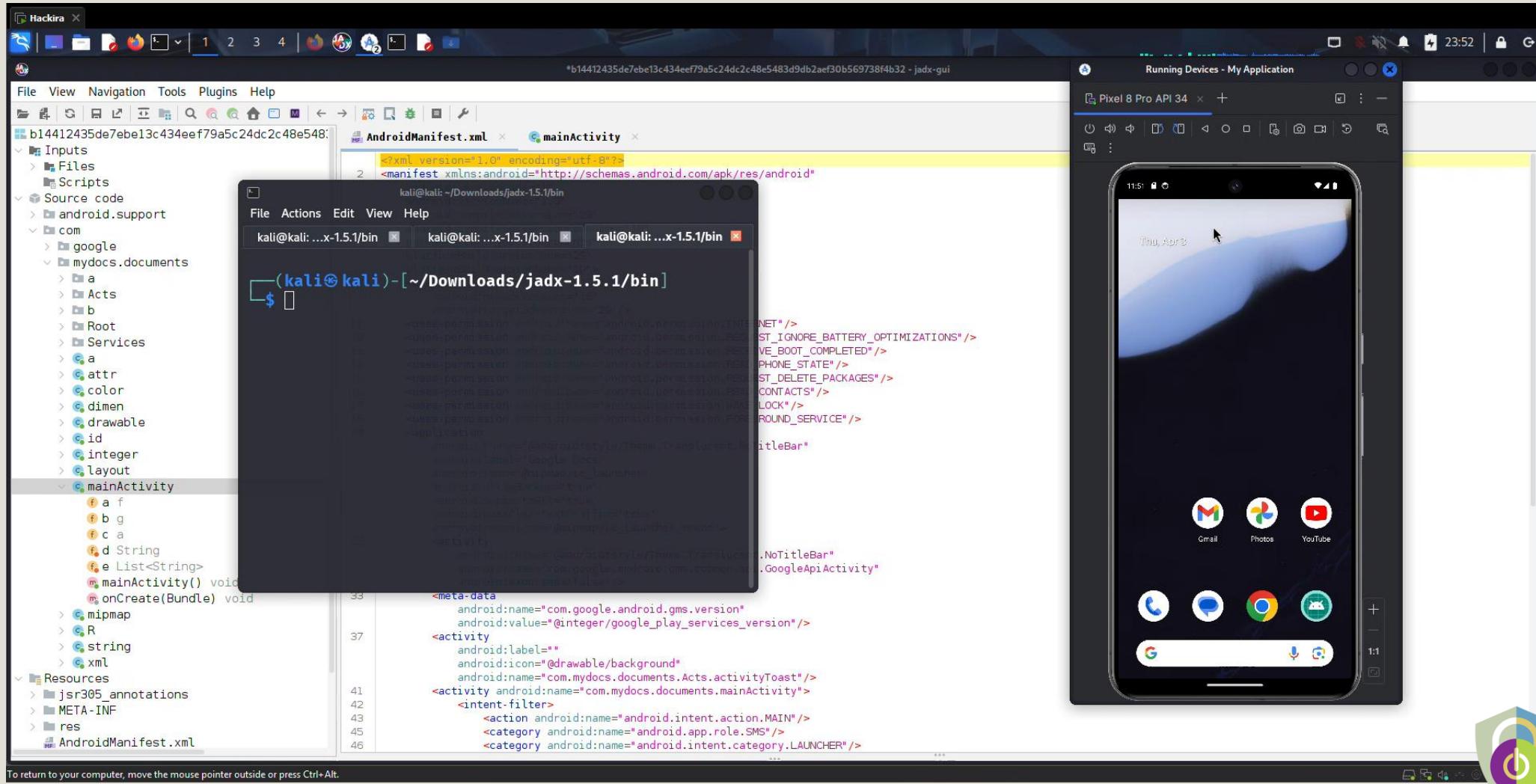
<https://www.threatfabric.com/blogs/double-trouble-in-latam>

The ThreatFabric blog post features a dark blue header with the ThreatFabric logo and a green 'BLOG' button. The main title is 'BrasDex: A new Brazilian ATS Android Banker with ties to Desktop malware'. Below the title is the date '15 December 2022'. The main content area contains a large image of a green, cybernetic-looking Android head wearing a gas mask, set against a background of the Brazilian flag and various electronic components like circuit boards and phones.

The Pix Payment System advertisement has a white header with the text 'Pix Payment System' and 'Developed by Banco Central do Brasil'. The main body is dark blue with the Pix logo (a teal diamond shape) and the text 'Allows users to send or receive payment transfers in few seconds at any time'. Below the text is the text 'powered by Banco Central'.

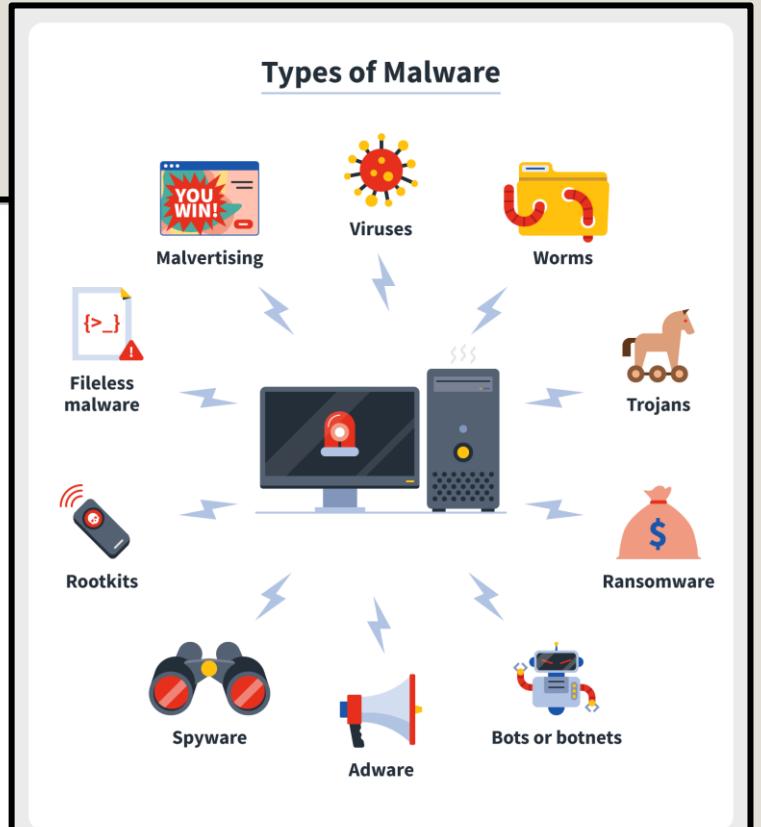
# *What about a Brazilian flavor?*

Family Type: BrasDex (Banking trojan)



# But all existing malwares are not just the same?

| Nexus   | Godfather   | Pixpirate  | Saderat  | Hook   | PixBankBot   | Xenomorph v3   | Vultur   | BrasDex   |
|---|---|--|--|--|--|--|--|---|
|  498 Known Variants |  1,171 Known Variants        |  123 Known Variants |  300 Known Variants |  14 Known Variants |  4 Known Variants |  6 Known Variants |  9 Known Variants |  1 Known Variant |
| 39 Banking Apps Targeted  | 237 Banking Apps Targeted   | 10 Banking Apps Targeted   | 8 Banking Apps Targeted  | 468 Banking Apps Targeted  | 4 Banking Apps Targeted  | 83 Banking Apps Targeted   | 122 Banking Apps Targeted  | 8 Banking Apps Targeted   |
| 9 Countries Targeted  | 57 Countries Targeted   | 1 Countries Targeted   | 23 Countries Targeted  | 43 Countries Targeted  | 1 Countries Targeted   | 14 Countries Targeted  | 15 Countries Targeted  | 1 Countries Targeted  |
| Offered as MaaS   | Offered as MaaS   | Not offered as MaaS  | Not offered as MaaS  | Offered as MaaS  | Not offered as MaaS  | Offered as MaaS  | Not offered as MaaS  | Not offered as MaaS   |
| Stolen Data Exfiltrated to:<br>USA<br>Netherlands<br>Turkey<br>Spain                                | Stolen Data Exfiltrated to:<br>USA<br>Turkey<br>Spain<br>Canada<br>France<br>Germany<br>UK<br>Italy<br>Poland | Stolen Data Exfiltrated to:<br>Brazil  | Stolen Data Exfiltrated to:<br>Thailand<br>Philippines<br>Peru                                       | Stolen Data Exfiltrated to:<br>Russia  | Stolen Data Exfiltrated to:<br>Brazil  | Stolen Data Exfiltrated to:<br>USA   | Stolen Data Exfiltrated to:<br>USA   | Stolen Data Exfiltrated to:<br>Australia<br>Poland  |



# No, let's *bridge* into another tool!



# No, let's *bridge* into another tool!



by *federicodotta and mustafairan*

- **Burp Suite Extension**
- **Born to turn some complex analysis of applications less time consuming**
- **Integrates function's hooks, native libraries' inspection with the amazing Burp Suite environment, limiting the reversing effort and by completely removing (in most situations) the developing one**
- **Provides an engine to visually create custom plugins for inspecting and editing HTTP requests/responses using the mobile app's own functions via Frida**
- **AND all that is included in *Burp Suite Community!***

# No, let's *bridge* into another tool!



by **federicodotta and mustafairan**

Burp Suite Community Edition v2025.1.1 - Temporary Project

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn **Brida** Settings

Configurations JS Editor Hooks and functions Graphical analysis Graphical hooks Custom plugins Generate stubs Debug export

Server status: **running**  
Application status: **running**  
Use virtual env:

Python binary path: `/usr/bin/python` Select file  
Pyro host: `localhost`  
Pyro port: `9999`  
frida-compile path: `/home/kali/node_modules/frida-compile/bin/frida-compile.js` Select file  
Use old version of frida-compile (< 10):   
Frida JS files folder: `/home/kali/Brida` Select folder Create default JS files  
Application ID (spawn) / PID (attach): `com.example.app`  
○ Frida Remote ○ Frida USB ○ Frida Local ○ Frida Host ○ Frida Device  
Host:port (Frida Host) / Device ID (Frida Device):

\*\*\*\* Console cleared successfully \*\*\*\*

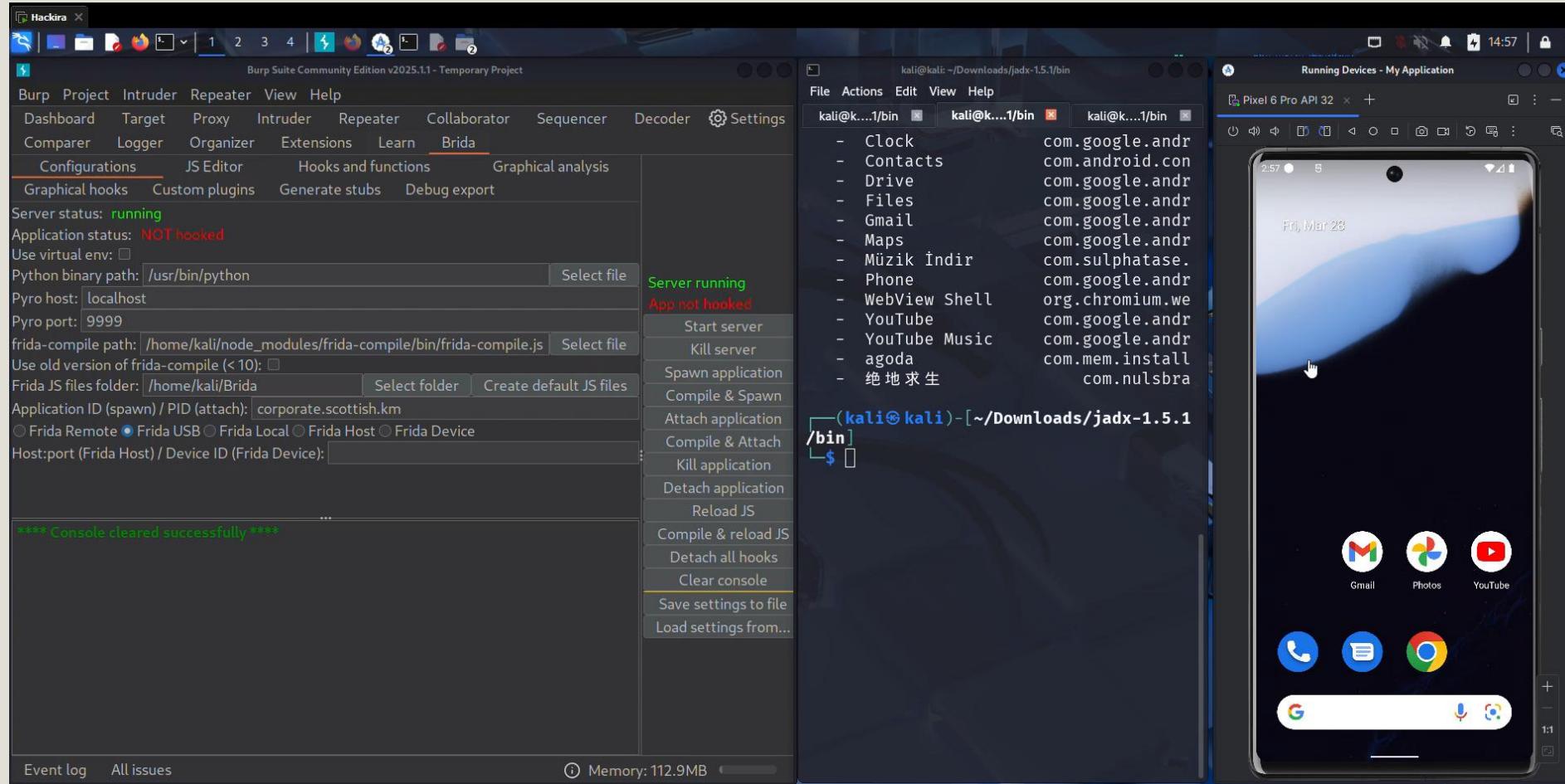
Event log All issues Memory: 140.5MB

Server running  
App hooked

- Start server
- Kill server
- Spawn application
- Compile & Spawn
- Attach application
- Compile & Attach
- Kill application
- Detach application
- Reload JS
- Compile & reload JS
- Detach all hooks
- Clear console
- Save settings to file
- Load settings from file

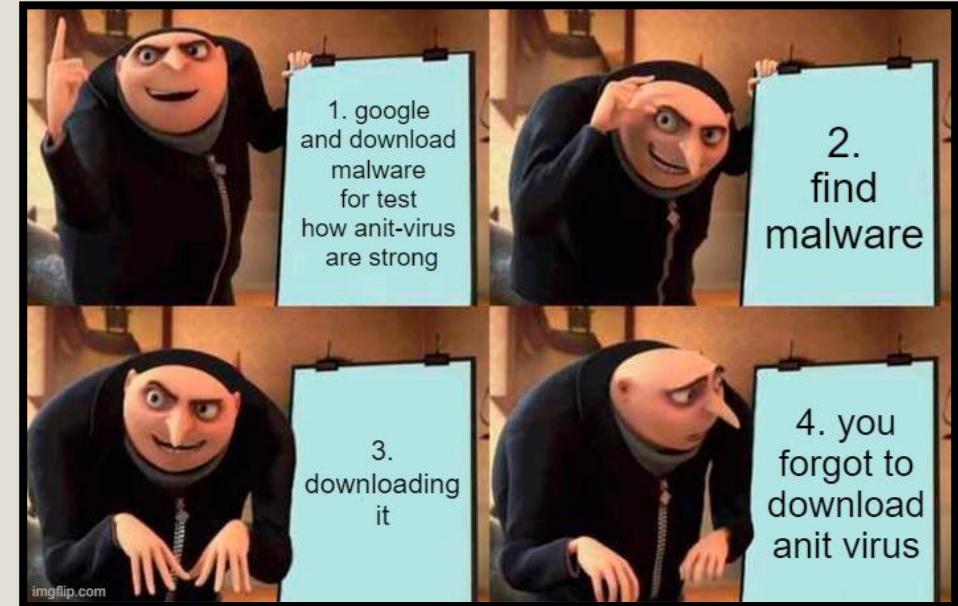
# No, let's *bridge* into another tool!

- Nothing is better than a good EXAMPLE USING 



# Download and analyze *malware samples!*

- *MalwareBazaar*
- *vx-underground*
- *ANY.RUN*: Registration required
- Hybrid Analysis: Registration required
- InQuest Malware Samples on GitHub
- KernelMode.info: Registration required
- MalShare: Registration required
- MalwareSamples Malware-Feed: Curated
- VirusBay: Registration required
- VirusShare: Registration required
- VirusSign: Registration required



# Conclusion and some *insights*

- Analyze applications is **HARD!**
- **API Interception:** Allows monitoring and altering of API calls to understand better the application
- **Real-time script injection** to modify app behavior
- It is **not** just spent time looking at the code
- **Careful with the time, complexity and your goals**
- **ALWAYS** be aware of the news and new patches



# References

- <https://www.fortinet.com/blog/threat-research/fortinet-identifies-malicious-packages-in-the-wild-insights-and-trends>
- <https://www.fortinet.com/blog/threat-research/defeating-an-android-packer-with-frida>
- <https://medium.com/@urban.vidergar/attacking-android-malware-with-frida-93c7cc0e1f50>
- <https://github.com/ineesdv/TFG-Frida-Scripts>
- <https://frida.re/docs/javascript-api/>
- <https://source.android.com/docs/security/bulletin/2025-03-01?hl=pt-br>
- <https://www.forbes.com/sites/zakdoffman/2024/03/15/samsung-s24-ultra-s23-free-update-warning-for-galaxy-android-users/>
- <https://thehackernews.com/2025/04/triada-malware-preloaded-on-counterfeit.html>



# Questions?

Let's connect!



[linkedin.com/in/stefany-coimbra/](https://linkedin.com/in/stefany-coimbra/)





# Entering the Android Rabbit Hole with Frida:

## Deep-Dive into Dynamic Analysis

# Frida scripts

## Bypass of Uncrackable.apk (OWASP Level 01)

```
1 // First Part
2 Java.perform(function(){
3   var c_function = Java.use("sg.vantagepoint.a.c");
4
5   c_function["a"].implementation = function () {
6     return false;
7   };
8
9   c_function["b"].implementation = function () {
10    return false;
11  };
12
13  c_function["c"].implementation = function () {
14    return false;
15  };
16
17 // Second Part
18 function convertToString(arg) {
19   var buffer = Java.array('byte', arg);
20   var result = "";
21   //Converting each byte to an individual char
22   for(var i = 0; i < buffer.length; ++i){
23     result += (String.fromCharCode(buffer[i] & 0xff));
24   }
25   return result;
26 }
27
28 Java.use("sg.vantagepoint.a.a").a.implementation = function(ba1, ba2) {
29   const retval = this.a(ba1, ba2);
30   console.log("secret code is: " + convertToString(retval));
31   return retval;
32 };
33});
```

# Frida scripts

## Bypassing Root (Malware: Visa Secure.apk)

```
1 Java.perform(function() {
2     var isEmulator =
3         Java.use("corporate.scottish.khcuwinjrgkokolanhrvznizqzenzehcnrnwfffcnbaojjmziv2.kvcqicw-
4             hbbmyqrejtoneqhzsupmwkxahcsjsgfvhnouryzrfaol4");
5
6     isEmulator["isEmu_DIV_ID_lator"].implementation = function() {
7         return false;
8     };
9
10    isEmulator["AsknoEmu"].implementation = function() {
11        console.log("I AM HERE!!!");
12    };
13
14    var isEmulator2 =
15        Java.use("corporate.scottish.khcuwinjrgkokolanhrvznizqzenzehcnrnwfffcnbaojjmziv2.jqdnvoi-
16            oolafpvdrvruzwlernmyztqhjrjfsyfvlhbydgoog31");
17
18    isEmulator2["isEmu_DIV_ID_lator"].implementation = function() {
19        return false;
20    };
21
22    isEmulator2["AsknoEmu"].implementation = function() {
23        console.log("I AM HERE!!!");
24    }
25});
```

# Frida scripts

## Searching for sockets (Malware: Visa Secure.apk)

```
1 function stackTrace() {
2     var ThreadDef = Java.use('java.lang.Thread');
3     var ThreadObj = ThreadDef.$new();
4
5     console.log(`[+] Stack trace...`);
6     console.log("-----");
7     var stack = ThreadObj.currentThread().getStackTrace();
8     for (var i = 0; i < stack.length; i++) {
9         console.log(` ${i}\t| ${stack[i].toString()}`);
10    }
11    console.log("-----");
12 }
13
14 function JavaSockets(){
15
16     var Socket = Java.use('java.net.Socket');
17
18     Socket.$init.overload('java.lang.String', 'int').implementation = function (host, port) {
19
20         console.log('[Socket] Socket created');
21         console.log(`\t[-] Host: ${host}`);
22         console.log(`\t[-] Port: ${port}\n`);
23         //stackTrace();
24
25         return this.$init(host, port);
26     };
27
28     Socket.connect.overload('java.net.SocketAddress', 'int').implementation = function (address,
29     timeout) {
30
31         console.log('[Socket] Socket connect());
32         console.log(`\t[-] Address: ${address.toString()}`);
33         console.log(`\t[-] Timeout: ${timeout}\n`);
34         //stackTrace();
35
36         return this.connect(address, timeout);
37     };
38     Socket.getInputStream.implementation = function () {
39 }
```

# Frida scripts

**Searching for  
sockets  
(Malware:  
Visa Secure.apk)**

```
40      console.log('[Socket] Socket getInputStream()');
41      console.log(`\t[-] Local Address: ${this.getLocalAddress().getHostAddress()}`);
42      console.log(`\t[-] Local Port: ${this.getLocalPort()}`);
43      console.log(`\t[-] Remote Address: ${this.getInetAddress().getHostAddress()}`);
44      console.log(`\t[-] Remote Port: ${this.getPort()}\\n`);
45      //stackTrace();
46
47      var inputStream = this.getInputStream();
48      return inputStream;
49  };
50
51  Socket.getOutputStream.implementation = function () {
52
53      console.log('[Socket] Socket getOutputStream()');
54      console.log(`\t[-] Local Address: ${this.getLocalAddress().getHostAddress()}`);
55      console.log(`\t[-] Local Port: ${this.getLocalPort()}`);
56      console.log(`\t[-] Remote Address: ${this.getInetAddress().getHostAddress()}`);
57      console.log(`\t[-] Remote Port: ${this.getPort()}\\n`);
58      //stackTrace();
59
60      var outputStream = this.getOutputStream();
61      return outputStream;
62  };
63
64  Socket.close.implementation = function () {
65      console.log('[Socket] close()');
66      return this.close();
67  };
68
69
70
71  var LocalServerSocket = Java.use('android.net.LocalServerSocket');
72
73  LocalServerSocket.$init.overload('java.lang.String').implementation = function (name) {
74      console.log(`[LocalServerSocket] Socket '${name}' created\\n`);
75      //stackTrace();
```

# Frida scripts

## Searching for sockets (Malware: Visa Secure.apk)

```
76      return this.$init(name);
77  };
78
79  LocalServerSocket.accept.implementation = function () {
80    console.log('[LocalServerSocket] Socket.accept()\n');
81    //stackTrace();
82
83    return this.accept();
84  };
85
86  LocalServerSocket.close.implementation = function () {
87    console.log('[LocalServerSocket] Socket.close()\n');
88    //stackTrace();
89
90    return this.close();
91  };
92
93
94
95
96  var ServerSocket = Java.use('java.net.ServerSocket');
97
98
99  ServerSocket.$init.overload('int').implementation = function (port) {
100    console.log('[ServerSocket] Socket created');
101    console.log(`\t[-] Port: ${port}\n`);
102    //stackTrace();
103
104    return this.$init(port);
105  };
106
107  ServerSocket.accept.implementation = function () {
108    console.log('[ServerSocket] Socket.accept()\n');
109    //stackTrace();
110
111    return this.accept();
112  };
113
114  ServerSocket.close.implementation = function () {
115    console.log('[ServerSocket] Socket.close()\n');
```

# Frida scripts

## Searching for sockets (Malware: Visa Secure.apk)

```
116         //stackTrace();
117
118     return this.close();
119 }
120
121
122
123 var SSLSocket = Java.use('javax.net.ssl.SSLSocket');
124
125 SSLSocket.$init.overload('java.lang.String', 'int').implementation = function (host, port) {
126
127     console.log('[SSLSocket] Socket created');
128     console.log(`\t[-] Host: ${host}`);
129     console.log(`\t[-] Port: ${port}\n`);
130     //stackTrace();
131
132     this.$init(host, port);
133 }
134
135 SSLSocket.getOutputStream.implementation = function () {
136
137     console.log('[SSLSocket] Socket.getOutputStream()\n');
138     // stackTrace();
139
140     var outputStream = this.getOutputStream();
141     // To log the data OutputStream.write needs to be intercepted
142
143     return outputStream;
144 }
145
146 SSLSocket.getInputStream.implementation = function () {
147
148     console.log('[SSLSocket] Socket.getInputStream()\n');
149     //stackTrace();
150
151     var inputStream = this.getInputStream();
152     // To log the data InputStream.read needs to be intercepted
153
154     return inputStream;
155 }
```

# Frida scripts

## Searching for sockets (Malware: Visa Secure.apk)

```
156
157 }
158
159
160 function NativeSockets(){
161     // Ref: https://gist.github.com/zihadmahiuddin
162
163     var getaddrinfoPtr = Module.findExportByName(null, 'getaddrinfo')
164     var connectPtr = Module.findExportByName(null, 'connect')
165     var sendPtr = Module.findExportByName(null, 'send')
166     var recvPtr = Module.findExportByName(null, 'recv')
167
168     var getaddrinfoFunction = new NativeFunction(getaddrinfoPtr, 'int', ['pointer', 'pointer',
169     'pointer', 'pointer'])
170     var connectFunction = new NativeFunction(connectPtr, 'int', ['int', 'pointer', 'int'])
171     var sendFunction = new NativeFunction(sendPtr, 'int', ['int', 'pointer', 'int', 'int'])
172     var recvFunction = new NativeFunction(recvPtr, 'int', ['int', 'pointer', 'int', 'int'])
173
174 /**
175 * Returns hex from an ArrayBuffer object
176 * @param {ArrayBuffer} array Array to work with
177 * @param {Boolean} hex Whether to convert to hex or plain string
178 */
179 function getReadable(array, hex) {
180     var result = new Uint8Array(array.byteLength)
181     result.set(array, 0)
182     if (hex === false) {
183         var str = ''
184         for (var i = 0; i < result.length; i++) {
185             str += String.fromCharCode(result[i])
186         }
187         return str
188     } else {
189         var hexStr = ''
190         for (var j = 0; j < result.length; j++) {
191             hexStr += result[j].toString(16)
192         }
193         return hexStr
194     }
}
```

# Frida scripts

## Searching for sockets (Malware: Visa Secure.apk)

```
195     }
196
197     /**
198      * Returns a nice formatting of a function with parameters
199      * @param {string} functionName The name of the function to format
200      * @param {string[]} params The function parameters as strings
201      */
202     function formatFunction(functionName, params, retval) {
203
204         var result = ` [NATIVE SOCKET] Function ${functionName} called\n`;
205         for (var i = 0; i < params.length; i++) {
206             result += `\t[-] Parameter ${i+1}: ${params[i]}\n`;
207         }
208         if (retval !== undefined) {
209             result += `\t[-] Return value: ${retval}\n`;
210         }
211         return result;
212     }
213
214
215     function replaceGadp() {
216         Interceptor.replace(getaddrinfoPtr, new NativeCallback(function(name, service, req, pai) {
217             var nameStr = Memory.readUtf8String(name)
218             console.log(formatFunction('getaddrinfo', [nameStr, service, req, pai]))
219             return getaddrinfoFunction(name, service, req, pai)
220         }, 'int', ['pointer', 'pointer', 'pointer', 'pointer']))
221     }
222
223     function replaceConnect() {
224         Interceptor.replace(connectPtr, new NativeCallback(function(socket, address, addressLen) {
225             var endpoint = {
226                 ip: '',
227                 port: 0
228             }
229             var portPtr = ptr(parseInt(address) + 2)
230             var portHigh = Memory.readU8(portPtr)
231             var portLow = Memory.readU8(ptr(parseInt(portPtr) + 1))
232             endpoint.port = (portHigh & 0xFF) << 8 | (portLow & 0xFF)
233
234             var ipPtr = ptr(parseInt(address) + 4)
```

# Frida scripts

## Searching for sockets (Malware: Visa Secure.apk)

```
235     var ip = []
236
237     ip.push(Memory.readU8(ipPtr))
238     ip.push(Memory.readU8(ptr(parseInt(ipPtr) + 1)))
239     ip.push(Memory.readU8(ptr(parseInt(ipPtr) + 2)))
240     ip.push(Memory.readU8(ptr(parseInt(ipPtr) + 3)))
241
242     endpoint.ip = ip.join('.')
243
244     var result = connectFunction(socket, address, addressLen)
245
246     console.log(formatFunction('connect', [socket, JSON.stringify(endpoint), addressLen],
  result));
247     return result
248   }, 'int', ['int', 'pointer', 'int']));
249 }
250
251 function replaceSend() {
252   Interceptor.replace(sendPtr, new NativeCallback(function(fd, buf, len, flags) {
253     var buffer = Memory.readByteArray(buf, len)
254     var result = sendFunction(fd, buf, len, flags)
255     console.log(formatFunction('send', [fd, getReadable(buffer, false), len, flags],
  result))
256     return result
257   }, 'int', ['int', 'pointer', 'int', 'int']));
258 }
```

# Frida scripts

## Searching for sockets (Malware: Visa Secure.apk)

```
259
260     function replaceRecv() {
261         Interceptor.replace(recvPtr, new NativeCallback(function(fd, buf, len, flags) {
262             var result = recvFunction(fd, buf, len, flags)
263             if (result > -1) {
264                 var buffer = Memory.readByteArray(buf, result)
265                 console.log(formatFunction('recv', [fd, getReadable(buffer, false), len, flags],
266                     result))
267             } else {
268                 console.log(formatFunction('recv', [fd, getReadable(buffer, false), len, flags],
269                     result))
270             }
271             return result
272         }, 'int', ['int', 'pointer', 'int', 'int']));
273     }
274     replaceGadp();
275     replaceConnect();
276     replaceSend();
277     replaceRecv();
278 }
```

# Frida scripts

## Searching for sockets (Malware: Visa Secure.apk)

```
279
280
281 Java.perform(function () {
282
283
284     console.log('');
285     console.log('[*] Hooking Java sockets ...');
286     console.log('');
287     JavaSockets();
288     console.log('[*] Hooking Native sockets ...');
289     console.log('');
290     NativeSockets();
291
292 });
293
```