Mystique in the House

The Droid Vulnerability Chain That Owns All Your Userspace

About us

- Focusing on Mobile System/iOT/Browser Research
- Pwn2Own winner & Pwnie Award Nominations
- Hundreds of CVEs from Google, Apple, Samsung etc
- Speaker at Blackhat, DEFCON, CanSecWest, MOSEC etc
- Our Twitter: @dawnseclab

Agenda

- Android sandbox review and typical historical vulnerabilities
- Introducing `Mystique` and how this chain is discovered
- Related vendor bugs in chain and static analysis technique
- Exploitation of the chain and how to detect it
- QA

Android Sandbox

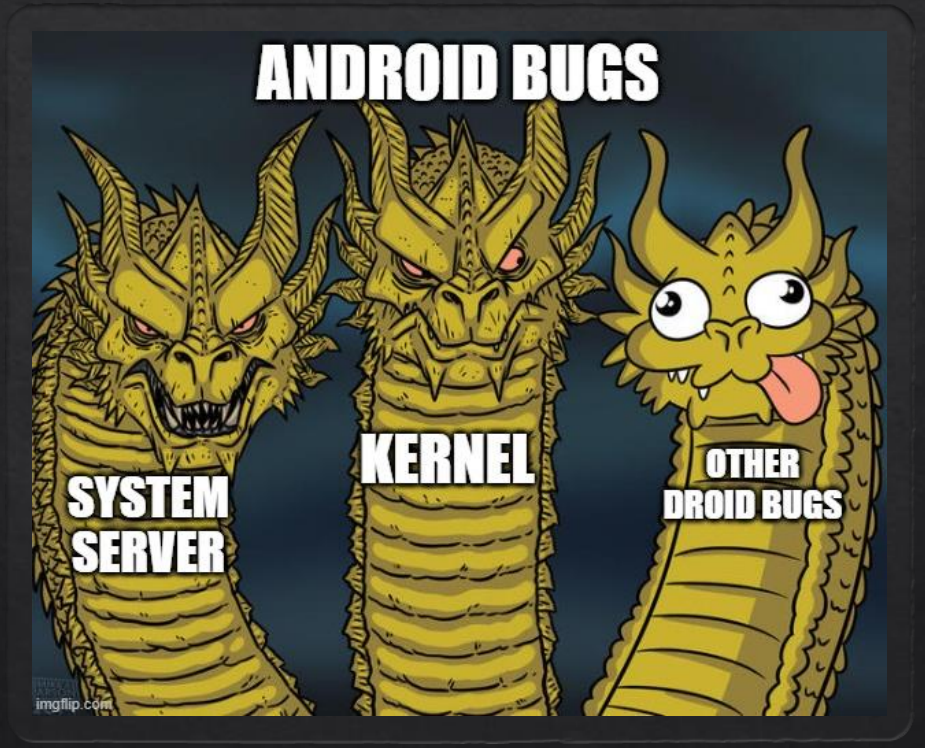
A Brief Overview

Android Process Sandbox

- DAC UID
  - isolated_app: u0_i1, …
  - Normal apps: u0_a10000 , …
  - radio, bluetooth, camera, nfc…
  - system
  - root
- MAC SELinux
  - isolated_app
  - untrusted_app (platform_app, priv_app)
  - radio, bluetooth, camera, nfc …
  - system_app, system_server, some binder service (keymaster,…)
  - Installd, ueventd, vold, kernel, …

Android Process Sandbox (cont.)

- System-Uid previously considered useless
  - Lots of bugs but not actually meaningful
    - From forensic/real world usage
  - Only system_server & kernel can cross process boundary
    - Last system_server exploitable bug is in 2016: *BitUnmap*
    - Kernel attacksurface heavily researched & mitigated: rare to seen/ difficult to exploit
- TODAY WE MAKE BUGS GREAT AGAIN



Introducing Mystique

Bug Chain Demystified

android / platform / system / sepolicy / 020e3ab0356e21a0013689cdb4899bb6ff13d398^! / .

```
   commit  020e3ab0356e21a0013689cdb4899bb6ff13d398        [log] [tgz]
   author  Songchun Fan <schfan@google.com>                Tue Feb 04 12:39:45 2020 -0800
committer  Songchun Fan <schfan@google.com>                Fri Feb 07 16:34:42 2020 +0000
     tree  b432d41adc9f61debf819cb46aa46626d9f84444
   parent  28d5e87d39b7a77b51e41c1d2ed61ceeb49501bc [diff]
```

```
selinux rules for apk files installed with Incremental

Apk files installed with Incremental are actually stored under the
/data/incremental directory.

Since files under /data/incremental are labeled as apk_file_data, we
need additional permissions to enable an apk installation.

Denial messages:
```

…this breaks with a single line of commit in Android 11

- https://android.googlesource.com/platform/system/sepolicy/+/020e3ab0356e21a0013689cdb4899bb6ff13d398%5E%21/#F1

```
Test: manual
BUG: 133435829
Change-Id: I70f25a6e63dd2be87ccbe9fb9e9d50fa64d88c36
```

```
diff --git a/private/system_app.te b/private/system_app.te
index e5d7d18..1432017 100644
--- a/private/system_app.te
+++ b/private/system_app.te
```

```
@@ -69,6 +69,9 @@
 # Settings need to access app name and icon from asec
 allow system_app asec_apk_file:file r_file_perms;

+# Allow system_app (adb data loader) to write data to /data/incremental
+allow system_app apk_data_file:file write;
+
 # Allow system apps (like Settings) to interact with statsd
 binder_call(system_app, statsd)
```

…this breaks with a single line of commit in Android 11

- https://android.googlesource.com/platform/system/sepolicy/+/020e3ab0356e21a0013689cdb4899bb6ff13d398%5E%21/#F1

Why is this change introduced?

- Android R (11) Preview 1 introduces adb data loader, as supplement for adb incremental install
  - AdbDataLoader as a system-app is allowed to modify apk code
  - Removed in Preview 4 and never made it into stable
  - However, the permission change is retained
- Who has apk_data_file label?
  - Prebundled applications in /system, by default readonly
  - New/updated apps in /data: app's base.apk and .so files, owner system:system

```
generic_x86:/system/priv-app/DownloadProvider # ls -lZ
total 204
-rw-r--r-- 1 root root u:object_r:system_file:s0 201300 2020-08-05 23:15 DownloadProvider.apk
drwxr-xr-x 3 root root u:object_r:system_file:s0   4096 2020-08-05 22:27 oat
generic_x86:/data/app/com.example.galaxystorepoc-hrCIiC7Ubu8nk1oHNOGT5w== # ls -lZ
total 3380
-rw-r--r-- 1 system system  u:object_r:apk_data_file:s0         3452587 2021-11-02 23:40 base.apk
drwxr-xr-x 2 system system  u:object_r:apk_data_file:s0            4096 2021-11-02 23:40 lib
drwxrwx--x 3 system install u:object_r:dalvikcache_data_file:s0    4096 2021-11-02 23:40 oat
-rwxr-xr-x 1 system system u:object_r:apk_data_file:s0   3349016 2021-07-02 19:49 libDownloadProxy.so
```

What's the consequence?

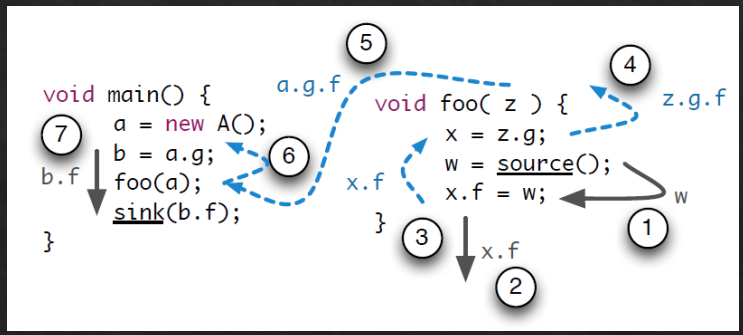
- This change in Android R(11) grants system_app the ability to write apk_data_file
  - Which greatly enlarges the power of system_app – and its vulnerabilities
    - To inject code into other apps by modifying their codes
  - Ability like system_server and root (in controlling other processes)
- Questions:
  - Will the file changes be rejected by certificate signature check?
    - For .so files, PackageManagerService won't recheck the integrity
      - Perfect persistent backdoor ^_^
    - For base.apk, rescan is not done until next reboot
    - Hijacked code is directly loaded and started by zygote, as if the original APK

The Mystique Chain

- This bug is reported to Google and assigned CVE-2021-0691 and fixed in Sep Bulletin
- We now need to find file overwrite bugs in system_app to chain it
  - We applied static analysis on major vendor ROMs to try to find them
  - And indeed found new ones
    - Samsung, Oppo, Xiaomi, etc…

Static analysis technique

- Classic source-sink taint analysis problem
  - Can be solved under IFDS dataflow framework on Soot
- Inter-procedure, finite, distributive subset (IFDS)
  - Context- and flow-, field-, object-sensitive
    - Insensitive overhead vs sensitive
  - IFDS works on Interprocedure-CFG
    - NormalFlowFuntions
    - CallFlowFunctions
    - ReturnFlowFunctions
    - CallToReturnFlowFunctions



Why not grantUriPermission ones?

- There exists a bug type for arbitrary file RW
  - grantUriPermission combined with FileProvider
- However, this cannot be used for Mystique
  - Write operation on FileDescriptors passed through binder is labelled writer's context

Static analysis technique

- Source
  - Intents and Bundles, Params from Entrypoints/Callbacks/Binder Transaction
  - Data Stream controllable such as Socket
  - Files/Resources from controllable locations such as package files and external storage
- Sink
  - Code loading
  - Command Execution
  - File IO

Static analysis technique: Difficulties Tackled

- Incomplete Callgraph
  - SPARK Pointer analysis cannot determine allocation node

```
public class Sample {
    public void target() {
        for(Helper helper: helpers)
            helper.handle();
    }

    Set<Helper> helpers;
    public Sample () {
        this.helpers = new HashSet<>();
        this.helpers.add(new HelperImplA());
    }
}

interface Helper {
    public void handle();
}

class HelperImplA implements Helper {

    @Override
    public void handle() {
        System.out.println("wtf");
    }
}

class HelperImplB implements Helper {

    @Override
    public void handle() {
    }
```

Static analysis technique: Difficulties Tackled

- Incomplete Callgraph
  - SPARK Pointer analysis cannot determine allocation node

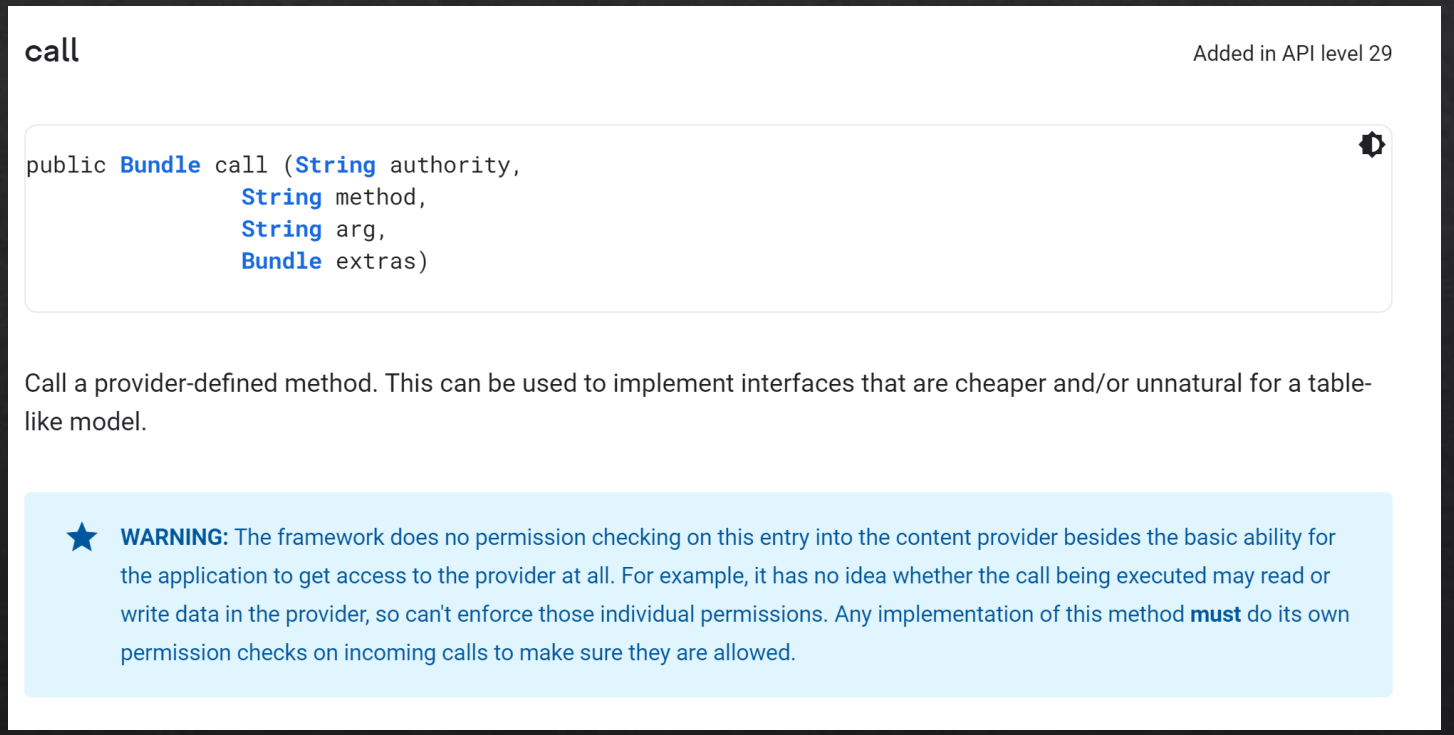
```
class AParcelable implements Parcelable {
    void func() {
    }
}

class MainActivity extends Activity {

    public void onResume() {
        AParcelable p = getIntent().getParcelableExtra("p");
        p.func();
    }
}
```

Static analysis technique: Several Difficulties Tackled

- Incomplete Callgraph
  - New Component Lifecycle



call

Added in API level 29

```
public Bundle call (String authority,
                    String method,
                    String arg,
                    Bundle extras)
```

Call a provider-defined method. This can be used to implement interfaces that are cheaper and/or unnatural for a table-like model.

★ **WARNING:** The framework does no permission checking on this entry into the content provider besides the basic ability for the application to get access to the provider at all. For example, it has no idea whether the call being executed may read or write data in the provider, so can't enforce those individual permissions. Any implementation of this method **must** do its own permission checks on incoming calls to make sure they are allowed.

Static analysis technique: Difficulties Tackled

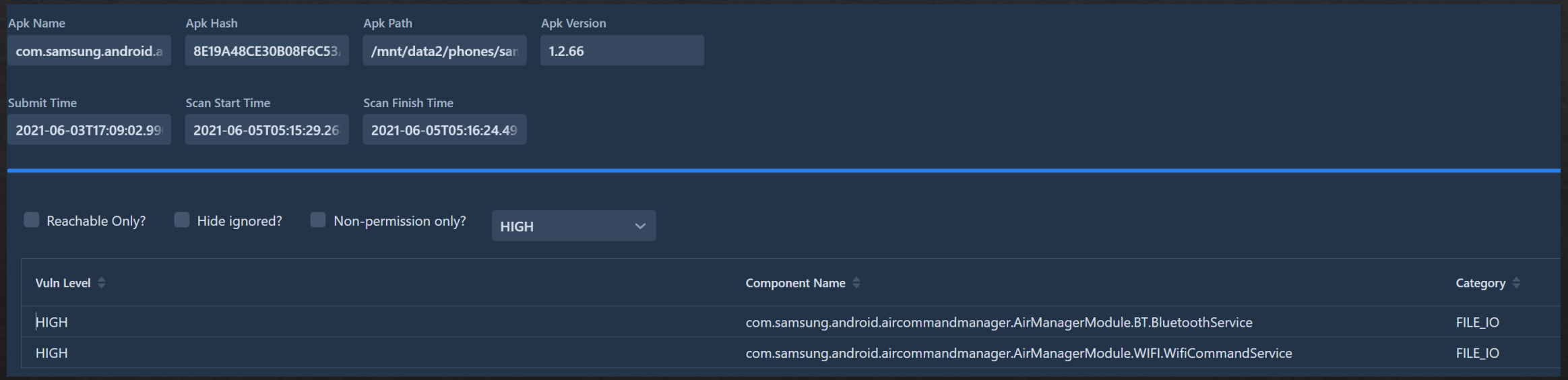
- Performance Tuning
  - Optimization for instance tainting
  - Drop taint fact for instance field if without sideeffect

The Mystique Chain: Samsung Part

- Samsung is the leading vendor in Android market after Huawei disappeared
- We found at least 3 new bugs that can be used to serve Mystique in Samsung devices
  - From zero permission app

The Mystique Chain: Samsung Part

- CVE-2021-25450: Path Traversal in FactoryAirCommandManager
  - FactoryAirCommandManager is a system-uid diag agent



dmanager.AirManagerModule.BT.BluetoothService: nanager.AirManagerModule.BT.BluetoothSocketModule ircommandmanager.AirManagerModule.BT.BluetoothService,com.samsung.android.aircommandmanager.AirManagerModule.BT....	$r0.<com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothService: com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothSocketModule btSerialSocket> = $r1
dmanager.AirManagerModule.BT.BluetoothService: nanager.AirManagerModule.BT.BluetoothSocketModule ircommandmanager.AirManagerModule.BT.BluetoothService,com.samsung.android.aircommandmanager.AirManagerModule.BT....	$r0 := @parameter0: com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothService
dmanager.AirManagerModule.BT.BluetoothService$BluetoothTask: java.lang.Void doInBackground(java.lang.Void[])>	$r2 = r0.<com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothService$BluetoothTask: com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothService this$0>
dmanager.AirManagerModule.BT.BluetoothService$BluetoothTask: java.lang.Void doInBackground(java.lang.Void[])>	$r2 = r0.<com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothService$BluetoothTask: com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothService this$0>
dmanager.AirManagerModule.BT.BluetoothService$BluetoothTask: java.lang.Void doInBackground(java.lang.Void[])>	$r10 = staticinvoke <com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothService: com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothSocketModule access$000(com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothService)>($r2)
dmanager.AirManagerModule.BT.BluetoothService: nanager.AirManagerModule.BT.BluetoothSocketModule ircommandmanager.AirManagerModule.BT.BluetoothService)>	r1 = $r0.<com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothService: com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothSocketModule btSerialSocket>
dmanager.AirManagerModule.BT.BluetoothService: nanager.AirManagerModule.BT.BluetoothSocketModule ircommandmanager.AirManagerModule.BT.BluetoothService)>	return r1
dmanager.AirManagerModule.BT.BluetoothService$BluetoothTask: java.lang.Void doInBackground(java.lang.Void[])>	virtualinvoke $r10.<com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothSocketModule: void startSocket(androic
dmanager.AirManagerModule.BT.BluetoothClient: void startSocket(android.content.Context)>	specialinvoke r0.<com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothSocketModule: void startSocket(android.c
dmanager.AirManagerModule.BT.BluetoothSocketModule: void startSocket(android.content.Context)>	$r7 = r0.<com.samsung.android.aircommandmanager.AirManagerModule.BT.BluetoothSocketModule: java.io.InputStream is>
dmanager.AirManagerModule.BT.BluetoothSocketModule: void startSocket(android.content.Context)>	$i1 = virtualinvoke $r7.<java.io.InputStream: int read(byte[])>(r2)
dmanager.AirManagerModule.BT.BluetoothSocketModule: void startSocket(android.content.Context)>	virtualinvoke $r12.<java.io.FileOutputStream: void write(byte[],int,int)>(r2, 0, $i1)

Running Static Analysis on preload apps

Taint Propagation Path identified

The Mystique Chain: Samsung Part

- CVE-2021-25450
  - BluetoothClient and BluetoothServer provides mode for receiving files
  - File path and content controlled by incoming bluetooth socket data
  - Pairing needed or not?

```
if(v8_2.toUpperCase().contains("AT+FACMFILE")) {
                    this.isFileMode = true;
                    String subStr = v8_2.substring(v8_2.indexOf("=") + 1,
v8_2.length() - 2);
                    this.fileName = subStr.split(",")[0];
                    this.fileLength = Integer.parseInt(subStr.split(",")[1]);
                    ACUtil.log_i("BluetoothSocketModule", "Socket", "name: " +
this.fileName + " length: " + this.fileLength);
                    File file = new File(BluetoothSocketModule.savePath);
                    if(!file.exists()) {
                        file.mkdirs();
                    }

                    File v3_1 = new File(BluetoothSocketModule.savePath +
this.fileName);
                    ACUtil.log_d("BluetoothSocketModule", "Socket", "Save: " +
v3_1.getPath());
                    if(this.fileLength != 0) {
                        this.fos = new FileOutputStream(v3_1);
                        this.mHandler.sendEmptyMessage(0);
                    }

                    maxLength = 0;
                    Arrays.fill(newBuffer, 0);
                    goto label_28;
                }
```

The Mystique Chain: Samsung Part

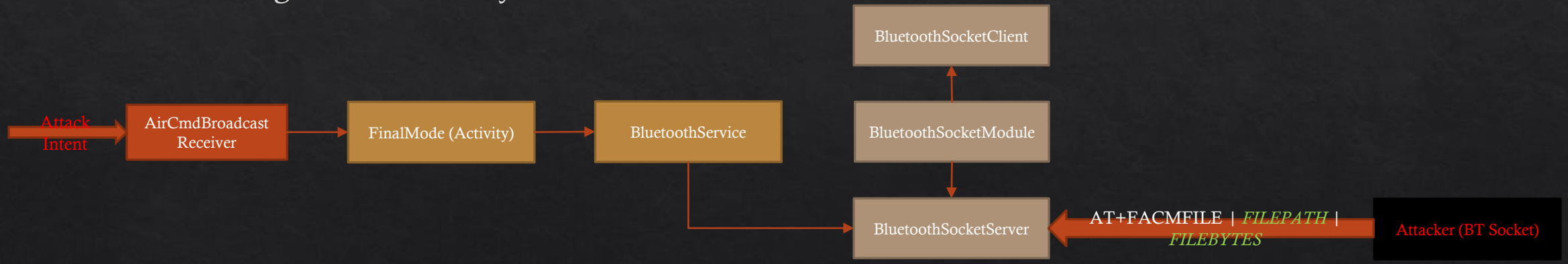
- CVE-2021-25450
  - BT Pairing is automatically done for us
  - Silently write out to controlled path with controlled content

```
if("android.bluetooth.device.action.PAIRING_REQUEST".equals(action)) {
                    try {
                        BluetoothDevice device =
(BluetoothDevice)intent.getParcelableExtra("android.bluetooth.device.extra.
DEVICE");
                        String v8 = device.getAddress();
                        BluetoothService.this.btClientAddress = v8;
                        int pin =
intent.getIntExtra("android.bluetooth.device.extra.PAIRING_KEY", 0x4D2);
                        ACUtil.log_d("BluetoothService", "Start Auto
Pairing. PIN = " +
intent.getIntExtra("android.bluetooth.device.extra.PAIRING_KEY", 0x4D2));
                        device.setPin("" + pin.getBytes("UTF-8"));
                        device.setPairingConfirmation(true);
                    }
                    catch(Exception e) {
                        ACUtil.log_d("BluetoothService", "Error occurs when
trying to auto pair");
                        e.printStackTrace();
                    }

                    return;
                }
```

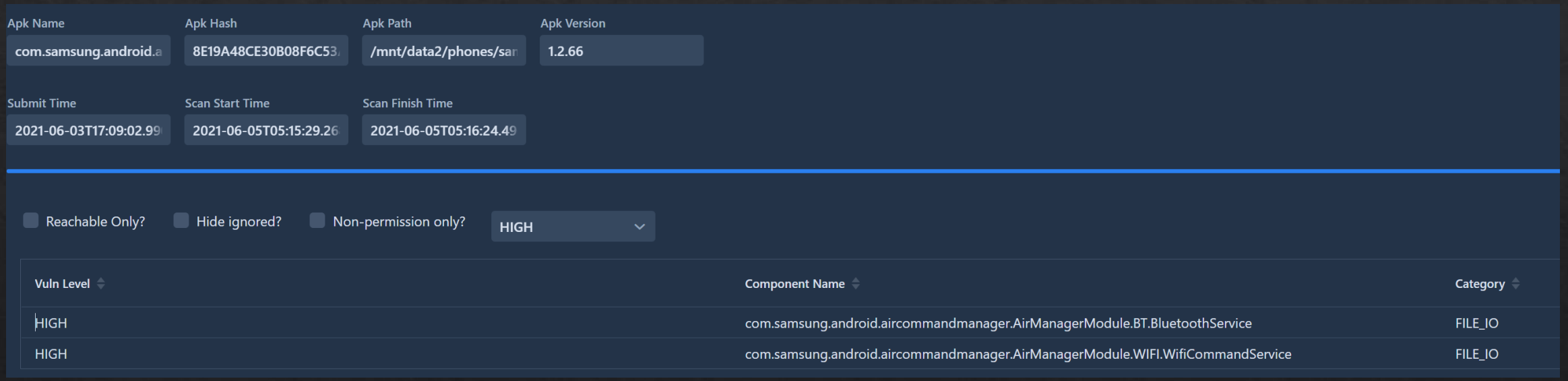
The Mystique Chain: Samsung Part

- Trigger Path
  - BluetoothSocketServer is created in BluetoothService (unexported)
  - BluetoothService created from FinalMode (unexported)
  - FinalMode can be redirected to from exported AirCmdBroadcastReceiver
  - Pairing is automatically done in BluetoothService



The Mystique Chain: Samsung Part

- CVE-2021-25485: Another Path Traversal in FactoryAirCommandManager



The Mystique Chain: Samsung Part 2

- CVE-2021-25510, CVE-2021-25511: Samsung Camera Privilege Escalation, Samsung FilterProvider Path Traversal
  - When a package with com.sec.android.camera.permission.USE_EFFECT_FILTER is installed, the system app FilterProvider is automatically notified through its registered broadcast receiver and recognize it as a CAMERA EFFECT FILTER.
  - The files in target package is extracted to /data/DownFilters/Lib64/

```
            v11_1 =
arg11.getPackageManager().getResourcesForApplication(arg12);

        AssetManager v12 = v11_1.getAssets();
        String[] v2 = new String[0];
        try {
            v2 = v12.list("so");
        }
        catch(IOException v3) {
            Log.e("FilterInstaller", "installLibFilter, asset lib list
exception : " + v3.toString());
        }

        if(v2.length <= 0) {
            Log.e("FilterInstaller", "asset lib list length is small than
0");
            return;
        }

        int v3_1 = v2.length;
        int v4;
        for(v4 = 0; v4 < v3_1; ++v4) {
            String v5 = v2[v4];
            Log.d("FilterInstaller", "filterFile : " + v5);
            if(this.storeLibFilters(v12, v11_1, v5, v5.split("\\.")[0]) == -
1L && (v5.endsWith(".so"))) {
                Log.e("FilterInstaller", "storeLibFilters fail");
            }
        }
```

The Mystique Chain: Samsung Part 2

- *Tex* and *so* files will be extracted
  - For *so* files, a sig file signed and verified by prebuilt Samsung RSA public key needs to be accompanied, or *content://com.samsung.android.provider.filterprovider/filters* will refused to insert it
  - Intended for use by Samsung Galaxy Store only
  - However, it does not verify *SO* suffix!

```
private long storeLibFilters(AssetManager arg18, Resources arg19, String arg20,
String arg21) {
  //...
          if((arg20.endsWith(".so")) && !this.checkSoSignature(v6)) {
            Log.e("FilterInstaller", "Signature check failed.");
            return -1L;
        }

        if(!arg20.endsWith(".sig")) {

            ContentResolver v10 = this.mServiceContext.getContentResolver();

            ContentValues v11 = new ContentValues();

            try {

//…

                if(v0_2 != null) {

                    v11.put("texture", v13);

                }

            }
```

The Mystique Chain: Samsung Part 2

- Camera Effect Filter is loaded when user opens Camera and uses magic effect
  - Preloaded in Camera process space by libcamera_effect_processor_jni.so
  - Got code execution in Camera process with *WRITE_EFFECT_FILTER* permission
    - Opens up ways to access *MyFilterService* in *FilterProvider*

```
__int64 __fastcall sub_13A3A4(__int64 a1, char *a2, char *s1)
{
  if ( strcmp(s1, "com.samsung.android.provider.filterprovider") && *(_DWORD
*)(a1 + 36) != 2 )
  {
    asprintf(ptr, "%s%s", "/data/DownFilters/Lib64/", a2);
    v7 = "/data/DownFilters/Tex/";
LABEL_11:
    ptr[1] = v7;
    goto LABEL_12;
  }
//...
      if ( v6 != 400 )
      {
          asprintf(ptr, "%s%s", "/system/lib64/", a2);
        v7 = "/system/cameradata/preloadfilters/Tex/";
        goto LABEL_11;
      }
    }
  }
LABEL_12:
  __android_log_print(3, "SECIMAGING", "Try to open external effect (%s)",
ptr[0]);
  *(_QWORD *)(a1 + 56) = dlopen(ptr[0], 2);
  free(ptr[0]);
  v8 = *(void **)(a1 + 56);
  if ( !v8 )
  {
    v19 = dlerror();
    __android_log_print(6, "SECIMAGING", "Filter %s dlopen failed", v19);
    return 0LL;
  }
  v9 = (__int64 (*)(void))dlsym(v8, "Create");
```

The Mystique Chain: Samsung Part 2

- MyFilterService, protected by signature-level WRITE_EFFECT_FILTER permission, has a typical path traversal
  - *And runs as system_app*

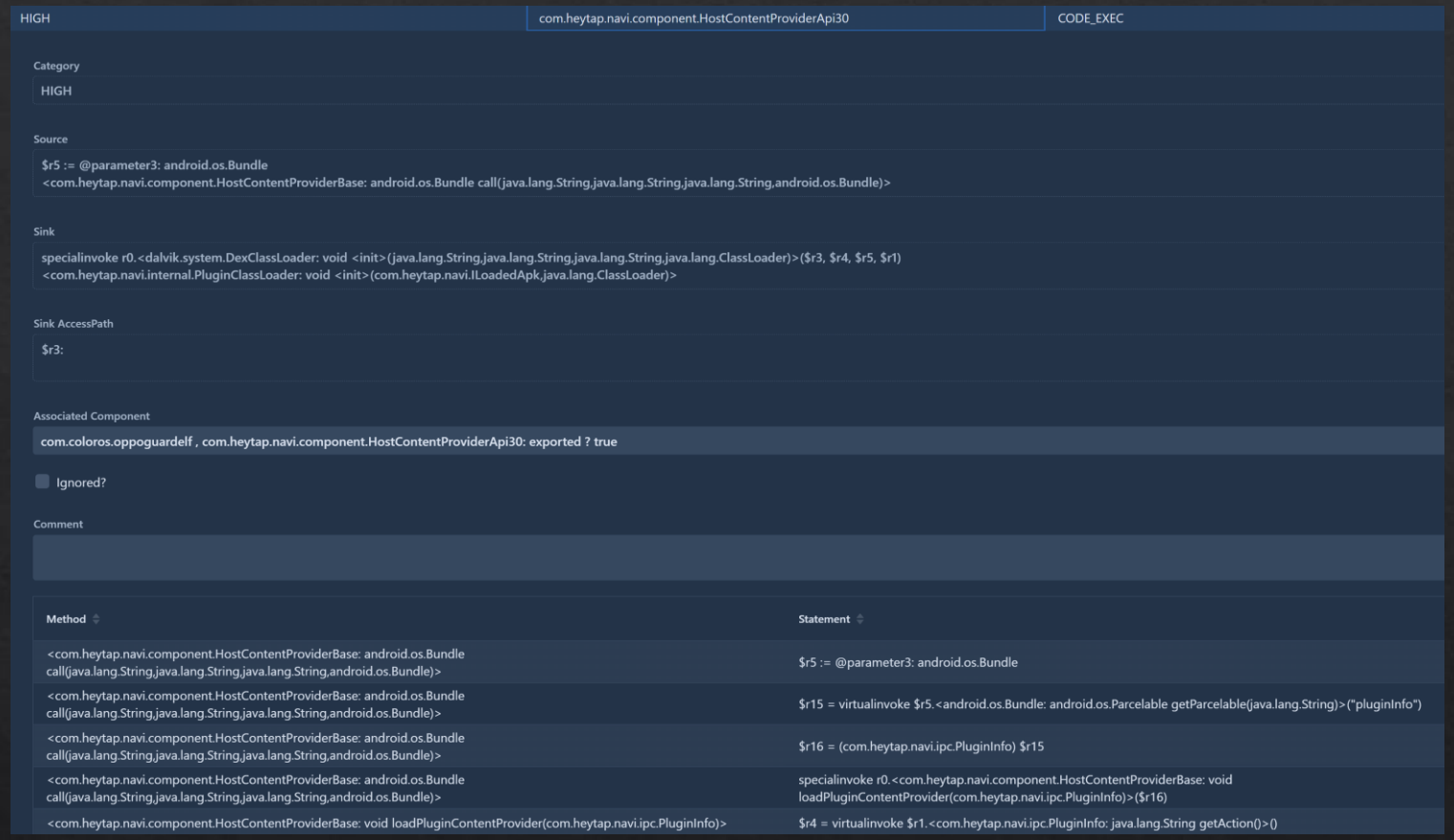
```
package com.samsung.android.provider.filterprovider;

    void install(Bundle arg15) {
//...
    String v4 = arg15.getString("filename");
    String v8 =
this.copySelFileFromIntent(MyFilterInstaller.FILTER_STORAGE_MY_FILTER + "/" +
v4, arg15);

//.. /data/xxx/ + "../../../../../"
 private String copySelFileFromIntent(String arg9, Bundle arg10) {
        //...
            v5 = new File(arg9);
        //...
                v9_1 = arg10.getByteArray("filter_data");
                v10 = new FileOutputStream(v5);
                //...
                    v10.write(v9_1);
```

The Mystique Chain: Oppo Part

- LPE for Mystique in Oppo was also identified: CVE-2021-20243
  - ContentProvider is exported with `call` entrance
  - Which loads external plugin but without sufficient validation
  - Attacker can fake a plugin to be loaded in host process



HIGH | com.heytap.navi.component.HostContentProviderApi30 | CODE_EXEC

Category

HIGH

Source

$r5 := @parameter3: android.os.Bundle
<com.heytap.navi.component.HostContentProviderBase: android.os.Bundle call(java.lang.String,java.lang.String,java.lang.String,android.os.Bundle)>

Sink

specialinvoke r0.<dalvik.system.DexClassLoader: void <init>(java.lang.String,java.lang.String,java.lang.String,java.lang.ClassLoader)>($r3, $r4, $r5, $r1)
<com.heytap.navi.internal.PluginClassLoader: void <init>(com.heytap.navi.ILoadedApk,java.lang.ClassLoader)>

Sink AccessPath

$r3:

Associated Component

com.coloros.oppoguardelf , com.heytap.navi.component.HostContentProviderApi30: exported ? true

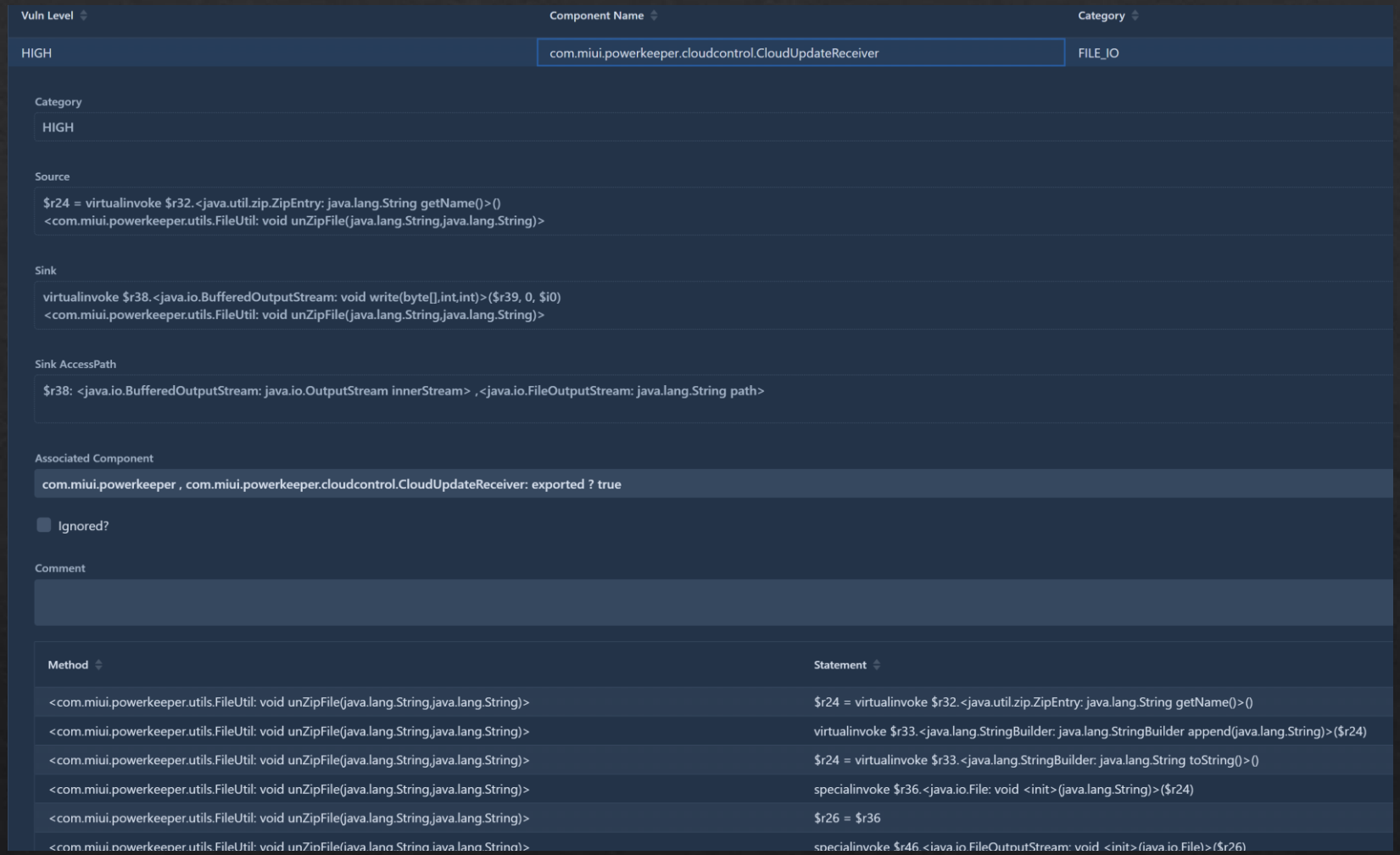
Ignored?

Comment

Method	Statement
<com.heytap.navi.component.HostContentProviderBase: android.os.Bundle call(java.lang.String,java.lang.String,java.lang.String,android.os.Bundle)>	$r5 := @parameter3: android.os.Bundle
<com.heytap.navi.component.HostContentProviderBase: android.os.Bundle call(java.lang.String,java.lang.String,java.lang.String,android.os.Bundle)>	$r15 = virtualinvoke $r5.<android.os.Bundle: android.os.Parcelable getParcelable(java.lang.String)>("pluginInfo")
<com.heytap.navi.component.HostContentProviderBase: android.os.Bundle call(java.lang.String,java.lang.String,java.lang.String,android.os.Bundle)>	$r16 = (com.heytap.navi.ipc.PluginInfo) $r15
<com.heytap.navi.component.HostContentProviderBase: android.os.Bundle call(java.lang.String,java.lang.String,java.lang.String,android.os.Bundle)>	specialinvoke r0.<com.heytap.navi.component.HostContentProviderBase: void loadPluginContentProvider(com.heytap.navi.ipc.PluginInfo)>($r16)
<com.heytap.navi.component.HostContentProviderBase: void loadPluginContentProvider(com.heytap.navi.ipc.PluginInfo)>	$r4 = virtualinvoke $r1.<com.heytap.navi.ipc.PluginInfo: java.lang.String getAction()>()

The Mystique Chain: Xiaomi Part

- LPE for Mystique in Xiaomi is also possible:
  - Unfortunately duplicate with Google AVPI program: CVE-2021-0501
  - Insufficient validation in unzipping



Vuln Level	Component Name	Category
HIGH	com.miui.powerkeeper.cloudcontrol.CloudUpdateReceiver	FILE_IO

Category

HIGH

Source

$r24 = virtualinvoke $r32.<java.util.zip.ZipEntry: java.lang.String getName()>()
<com.miui.powerkeeper.utils.FileUtil: void unZipFile(java.lang.String,java.lang.String)>

Sink

virtualinvoke $r38.<java.io.BufferedOutputStream: void write(byte[],int,int)>($r39, 0, $i0)
<com.miui.powerkeeper.utils.FileUtil: void unZipFile(java.lang.String,java.lang.String)>

Sink AccessPath

$r38: <java.io.BufferedOutputStream: java.io.OutputStream innerStream> ,<java.io.FileOutputStream: java.lang.String path>

Associated Component

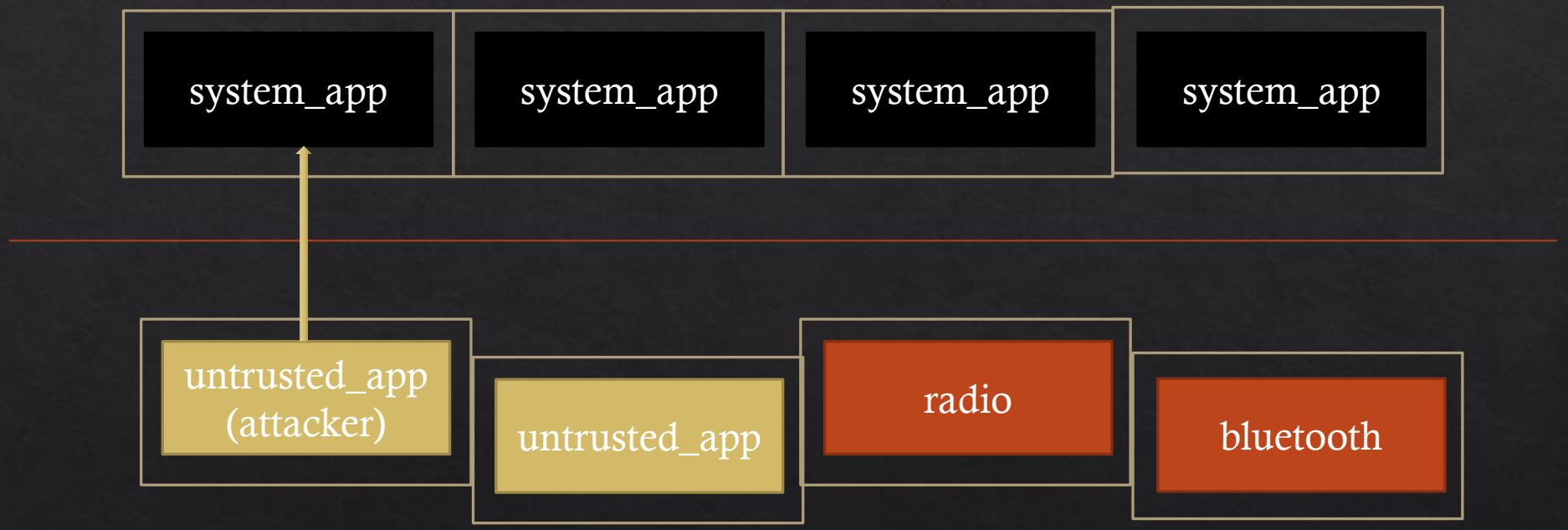
com.miui.powerkeeper , com.miui.powerkeeper.cloudcontrol.CloudUpdateReceiver: exported ? true

Ignored?

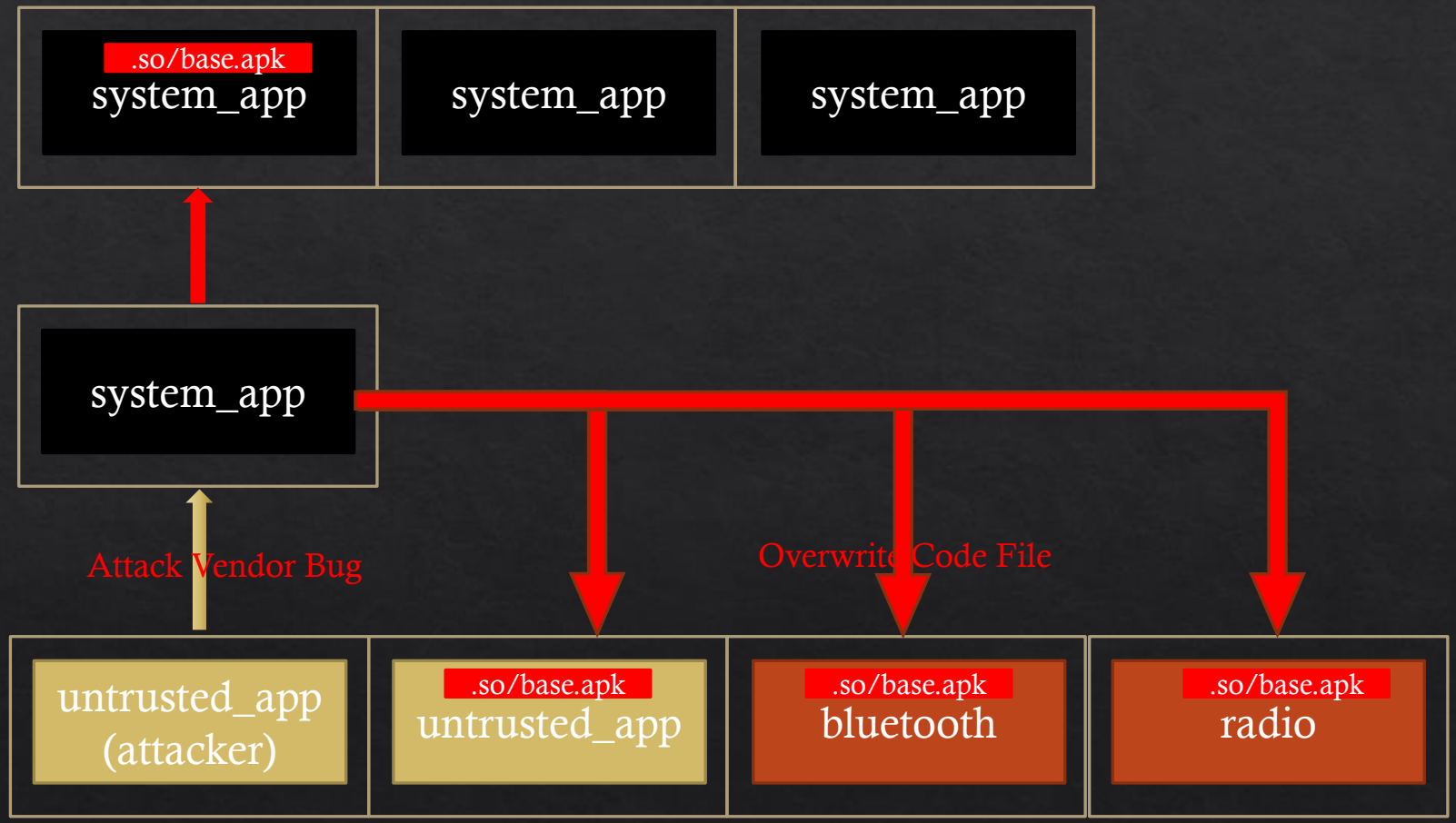
Comment

Method	Statement
<com.miui.powerkeeper.utils.FileUtil: void unZipFile(java.lang.String,java.lang.String)>	$r24 = virtualinvoke $r32.<java.util.zip.ZipEntry: java.lang.String getName()>()
<com.miui.powerkeeper.utils.FileUtil: void unZipFile(java.lang.String,java.lang.String)>	virtualinvoke $r33.<java.lang.StringBuilder: java.lang.StringBuilder append(java.lang.String)>($r24)
<com.miui.powerkeeper.utils.FileUtil: void unZipFile(java.lang.String,java.lang.String)>	$r24 = virtualinvoke $r33.<java.lang.StringBuilder: java.lang.String toString()>()
<com.miui.powerkeeper.utils.FileUtil: void unZipFile(java.lang.String,java.lang.String)>	specialinvoke $r36.<java.io.File: void <init>(java.lang.String)>($r24)
<com.miui.powerkeeper.utils.FileUtil: void unZipFile(java.lang.String,java.lang.String)>	$r26 = $r36
<com.miui.powerkeeper.utils.FileUtil: void unZipFile(java.lang.String,java.lang.String)>	specialinvoke $r46.<java.io.FileOutputStream: void <init>(java.io.File)>($r26)

The Mystique Chain: Conclusion



The Mystique Chain: Conclusion



Post Exploitation

Towards a real-world practical attack

Post Exploitation: Unleash the full power of Mystique

- Given the presented chain, how can we actually turn it into practical control over other apps?
- Recall we can overwrite two kinds of files under `/data/app/xxx/`: dynamic library and base.apk
  - Change on dynamic library is persistent until next uninstall
  - Change on base.apk is persist until next reboot: Smali repackaging is suffice
- Inject Frida-gadget for our convenience

Post Exploitation: Unleash the full power of Mystique

- For dynamic library injection, ELF patching is needed to preserve the original function of the target
  - Take Telegram as example
  - We use LIEF to inject NEEDED entry in the dynamic section of libtmessages.38.so
  - Frida-gadget loaded automatically

Post Exploitation: Unleash the full power of Mystique

```
Dynamic section at offset 0x1348300 contains 34 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libjnigraphics.so]
 0x0000000000000001 (NEEDED)             Shared library: [liblog.so]
 0x0000000000000001 (NEEDED)             Shared library: [libz.so]
 0x0000000000000001 (NEEDED)             Shared library: [libGLESv2.so]
 0x0000000000000001 (NEEDED)             Shared library: [libandroid.so]
 0x0000000000000001 (NEEDED)             Shared library: [libOpenSLES.so]
 0x0000000000000001 (NEEDED)             Shared library: [libdl.so]
 0x0000000000000001 (NEEDED)             Shared library: [libm.so]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so]
 0x000000000000000e (SONAME)             Library soname: [libtmessages.38.so]
 0x0000000000000019 (INIT_ARRAY)         0x12de638
 0x000000000000001b (INIT_ARRAYSZ)       64 (bytes)
 0x000000000000001a (FINI_ARRAY)         0x12de678
 0x000000000000001c (FINI_ARRAYSZ)       16 (bytes)
 0x0000000000000004 (HASH)               0x228
 0x000000006ffffef5 (GNU_HASH)           0x41e00
 0x0000000000000005 (STRTAB)             0x155a28
 0x0000000000000006 (SYMTAB)             0x8b3a8
 0x000000000000000a (STRSZ)              2121242 (bytes)
 0x000000000000000b (SYMENT)             24 (bytes)
 0x0000000000000003 (PLTGOT)             0x1349560
 0x0000000000000002 (PLTRELSZ)           293064 (bytes)
 0x0000000000000014 (PLTREL)             RELA
 0x0000000000000017 (JMPREL)             0x464980
 0x0000000000000007 (RELA)               0x36c688
 0x0000000000000008 (RELASZ)             1016568 (bytes)
 0x0000000000000009 (RELAENT)            24 (bytes)
 0x000000000000001e (FLAGS)              BIND_NOW
 0x000000006ffffffb (FLAGS_1)            Flags: NOW
 0x000000006ffffffe (VERNEED)            0x36c628
 0x000000006fffffff (VERNEEDNUM)         3
 0x000000006ffffff0 (VERSYM)             0x35b842
 0x000000006ffffff9 (RELACOUNT)          17576
 0x0000000000000000 (NULL)               0x0
```

```
Dynamic section at offset 0x14ac000 contains 35 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [/data/app/~~JlOalurLK-eqrdNB1QP5WA==/com.example.xx
xxxxxxxxx-h8v78MdAw3uZAepyluxiLA==/lib/arm64/intentionally-reserved-space-dir/libfrida-gadget.so]
 0x0000000000000001 (NEEDED)             Shared library: [libjnigraphics.so]
 0x0000000000000001 (NEEDED)             Shared library: [liblog.so]
 0x0000000000000001 (NEEDED)             Shared library: [libz.so]
 0x0000000000000001 (NEEDED)             Shared library: [libGLESv2.so]
 0x0000000000000001 (NEEDED)             Shared library: [libandroid.so]
 0x0000000000000001 (NEEDED)             Shared library: [libOpenSLES.so]
 0x0000000000000001 (NEEDED)             Shared library: [libdl.so]
 0x0000000000000001 (NEEDED)             Shared library: [libm.so]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so]
 0x000000000000000e (SONAME)             Library soname: [libtmessages.38.so]
 0x0000000000000019 (INIT_ARRAY)         0x12e1638
 0x000000000000001b (INIT_ARRAYSZ)       64 (bytes)
 0x000000000000001a (FINI_ARRAY)         0x12e1678
 0x000000000000001c (FINI_ARRAYSZ)       16 (bytes)
 0x0000000000000004 (HASH)               0x3228
 0x000000006ffffef5 (GNU_HASH)           0x44e00
 0x0000000000000005 (STRTAB)             0x96b3000
 0x0000000000000006 (SYMTAB)             0x8e3a8
 0x000000000000000a (STRSZ)              4243456 (bytes)
 0x000000000000000b (SYMENT)             24 (bytes)
 0x0000000000000003 (PLTGOT)             0x134c560
 0x0000000000000002 (PLTRELSZ)           293064 (bytes)
 0x0000000000000014 (PLTREL)             RELA
 0x0000000000000017 (JMPREL)             0x467980
 0x0000000000000007 (RELA)               0x36f688
 0x0000000000000008 (RELASZ)             1016568 (bytes)
 0x0000000000000009 (RELAENT)            24 (bytes)
 0x000000000000001e (FLAGS)              BIND_NOW
 0x000000006ffffffb (FLAGS_1)            Flags: NOW
 0x000000006ffffffe (VERNEED)            0x36f628
 0x000000006fffffff (VERNEEDNUM)         3
 0x000000006ffffff0 (VERSYM)             0x35e842
 0x000000006ffffff9 (RELACOUNT)          17576
 0x0000000000000000 (NULL)               0x0
```

Unleash the full power of Mystique

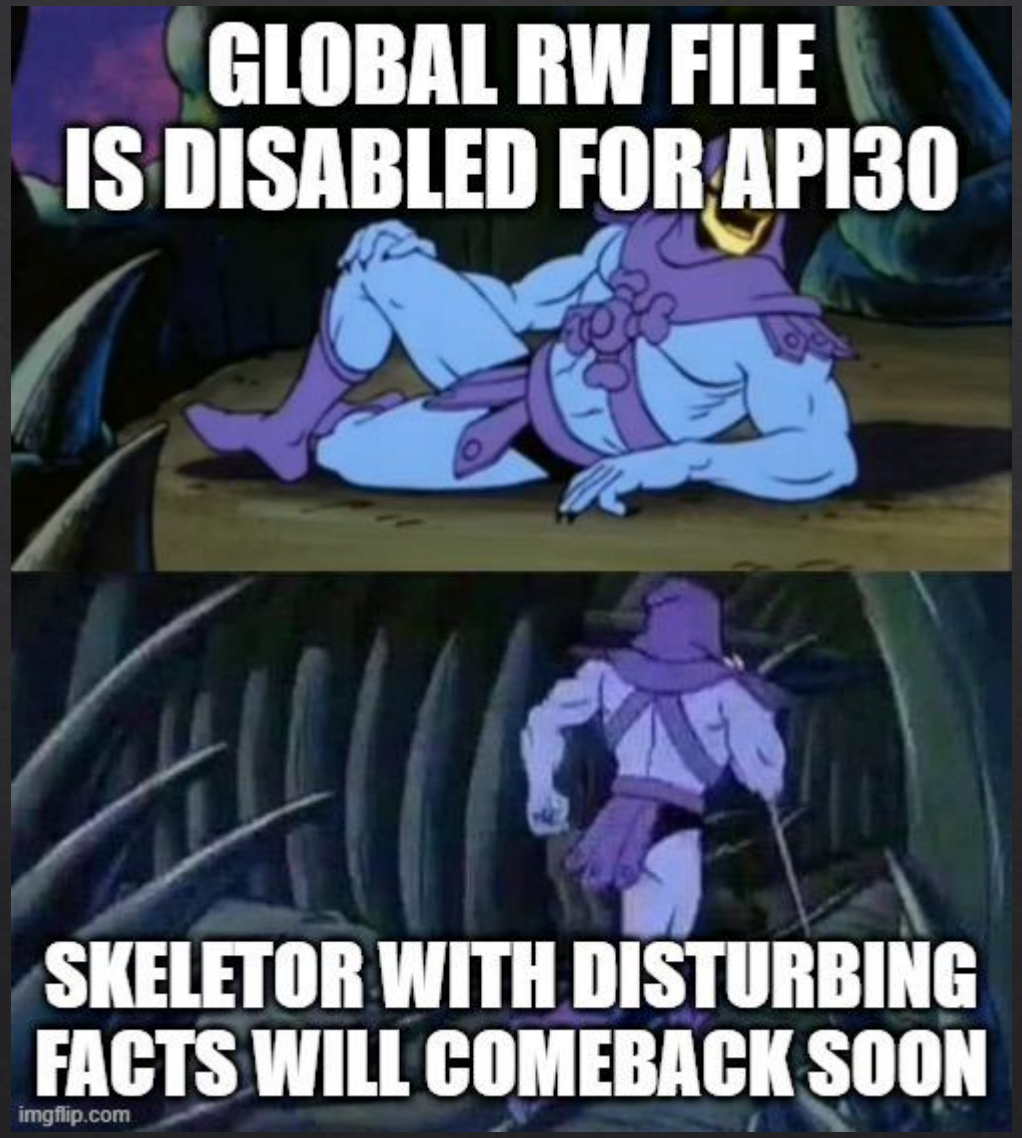
- Frida-gadget mode needs a configuration file and an instruction script
  - Libfrida-gadget-config: instruction script path
  - Frida-script.js: hooking code
  - Libfrida-gadget need to be executable/readable for victim
  - Process running libfrida-gadget.so must be able to read config/script files
- Challenge:
  - How to achieve this stealthily and stably?

Dancing in shackles

Use external storage?

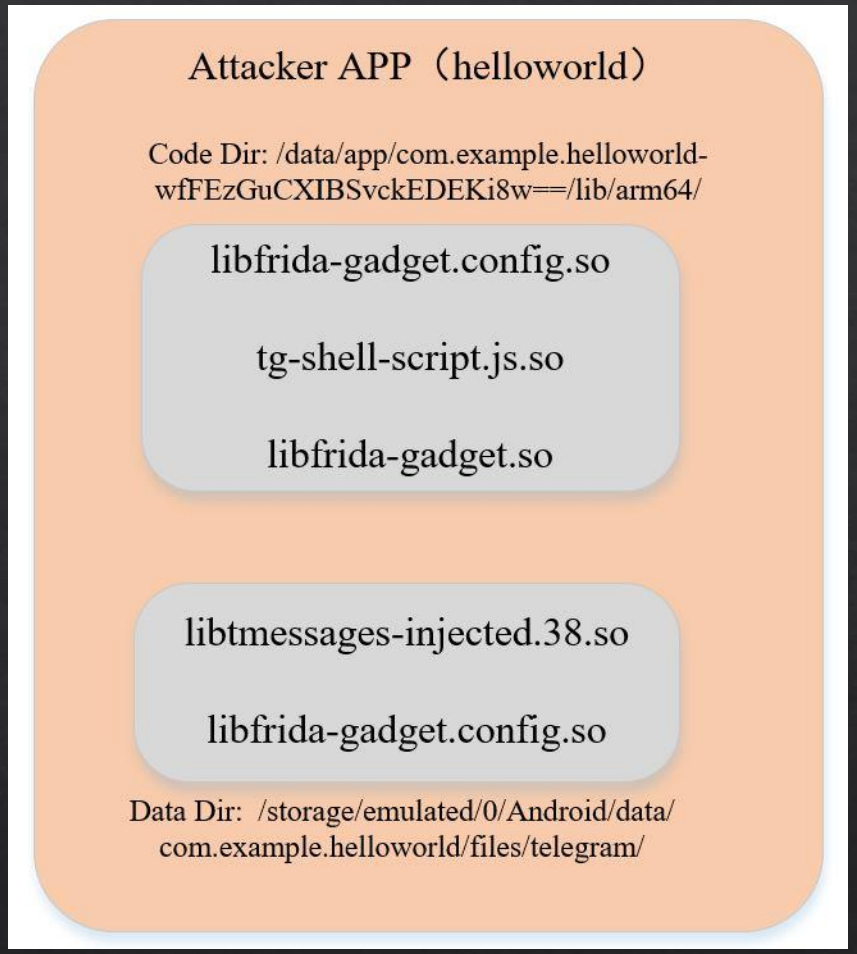
Dancing in shackles

Place world readable/writable files in my own app data directory?



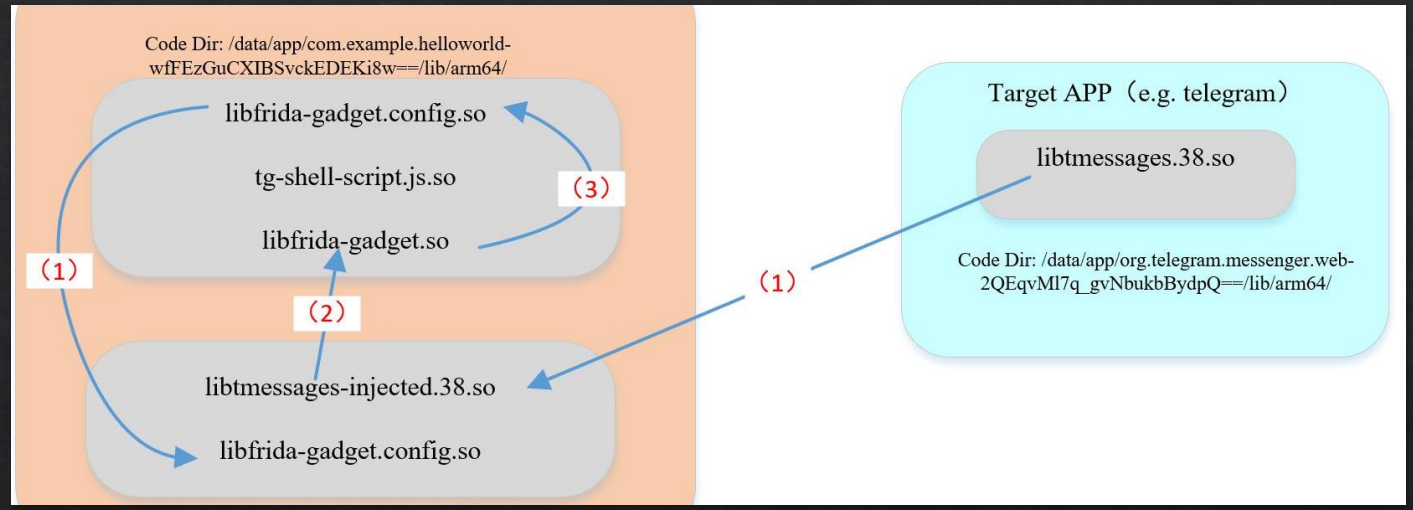
Unleash the full power of Mystique: Dancing in shackles

- Solution: pack them in our app's library folder!
  - App-lib files are globally readable/executable by everyone
  - For path changes required in config file, use Mystique to attack ourself to update our config
    - Android randomize installation path for each app upon each install
  - Make the config file points back to frida script in our app-lib



Final steps

- Release and patch the frida-gadget config file and victim's so based on obtained app path
  - Point victim entry back to frida-gadget in attacker's path
  - Point config entry back to script file
- Copy them back to attacker's lib folder
- Copy patched target so and overwrite victim
- BOOM



DEMO

Detection

Detection of Mystique

- Unlike traditional rooting exploits, Mystique does not have kernel or su stuff traces
  - Traditional rooting detection mechanism will not work
  - SafetyNet and other risk detection SDKs may fail
- Library timestamp is a clue
  - Scan all last modification timestamps of files in app code directory
  - File with abnormal timestamp show the app might be attacked
  - We have released detection tool and SDK under our website

Conclusion

- Mystique is a new type of vulnerability chain that can obtain arbitrary app's privilege from zero-permission app, whose ability somehow similar to kernel and system_server exploits
- It may evade traditional detection mechanism and we proposed way to find possible attacks

Credits

- Google and Samsung Mobile Security
- Soot Community

Questions?