Breaking into Android IPC Mechanisms through Advanced AIDL Fuzzing

Rajanish Pathak & Hardik Mehta

AHMEDABAD
BSIDES

About Us!

Hardik Mehta

@hardw00t

Head of Security Services

Rajanish Pathak

@h4ckologic

Manager Security Services

Our Motivation

INCREASING POPULARITY OF ANDROID PLATFORMS

COMPLEX MECHANISMS

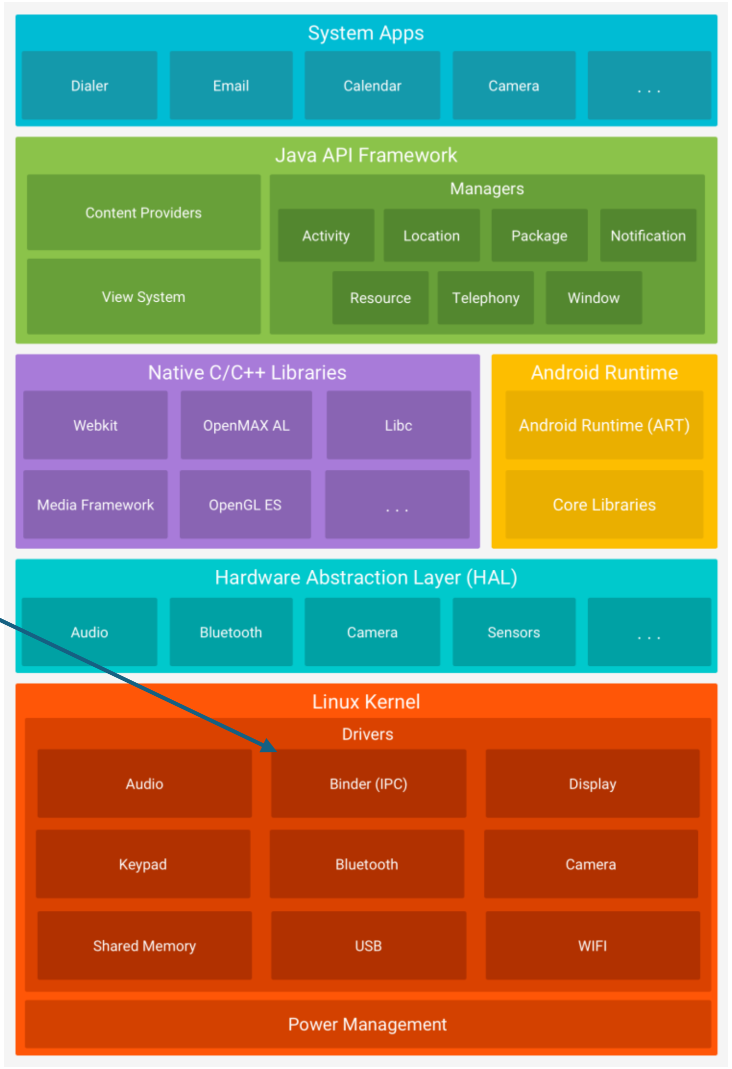
UNDERESTIMATED ATTACK SURFACE OF IPC AND AIDL

REAL-WORLD IMPACT

Agenda

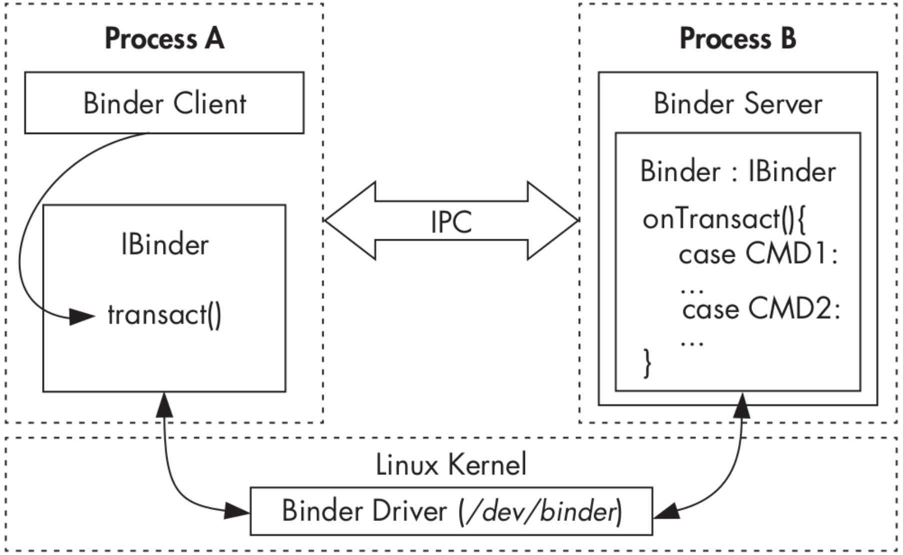
- Overview of Android IPC Mechanisms
- Introduction to AIDL Fuzzing
- Tools and Frameworks for AIDL Fuzzing
- Demo
- Challenges in AIDL Fuzzing
- Q&A

Overview of Android's IPC Mechanisms



Overview of Android's IPC Mechanisms

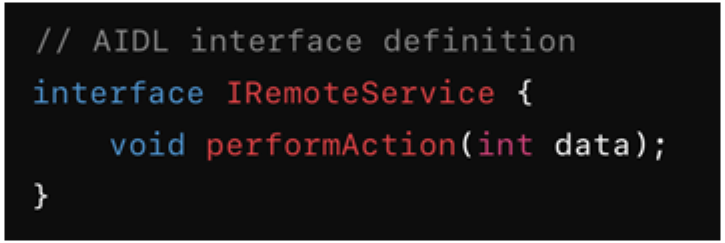
- **What is IPC in Android?**
  - IPC enables communication between processes (e.g., services, activities)
- **Android IPC Mechanisms:**
  - Binders (Kernel-level)
  - Intents
  - AIDL for complex IPC.
- **Why IPC Security Matters:**
  - Attack surfaces between trusted and untrusted processes.



Android Interface Definition Language (AIDL)

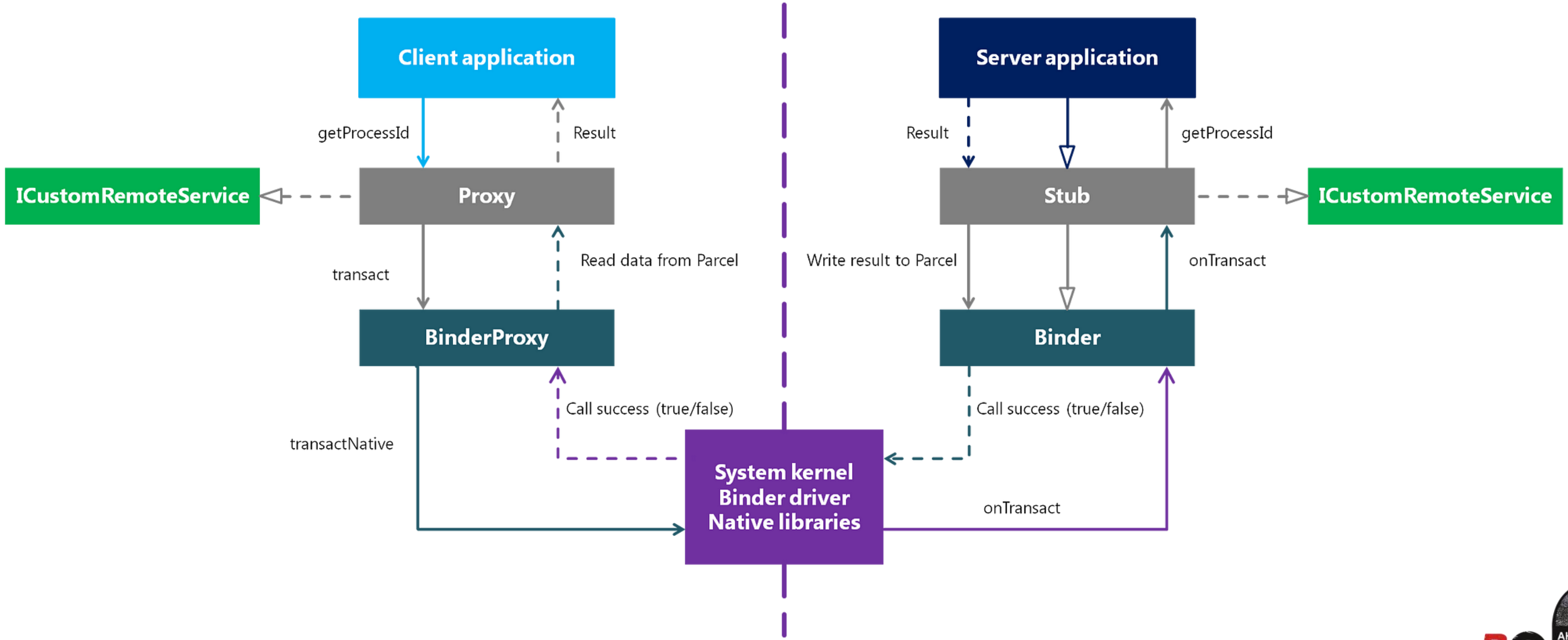
- **What is AIDL?**
  - AIDL allows processes to communicate with each other using defined interfaces.
  - Role in complex applications (e.g., system services, background apps).

- **Basic Structure of an AIDL Interface:**
  - Defines methods, data types, and parameters.



```
// AIDL interface definition
interface IRemoteService {
    void performAction(int data);
}
```

Android Interface Definition Language (AIDL)



AIDL in Action

- **How AIDL Works:**
  - The process of using AIDL in Android (service binding).
  - Example flow: App A communicates with service B using AIDL
- **Example of AIDL Use Cases:**
  - Audio playback control, background service management, etc.

Create .aidl File
AIDL Stub is created by the Android Studio Framework

↓

Declare the methods to be used in .AIDL file
Expose the Interface to the clients

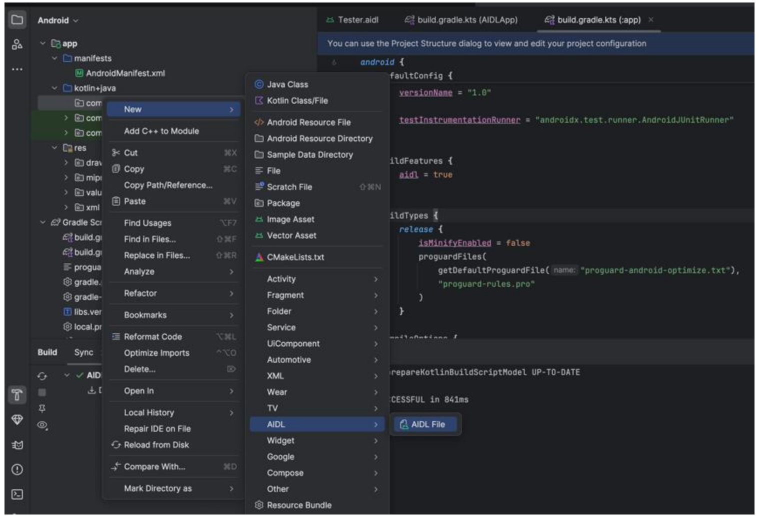
↓

Server will implement the stub and create an instance of binder.

↓

The client call BindService() to connect to the binder and a subsequent onServiceConneted() is called which passes the Binder Object

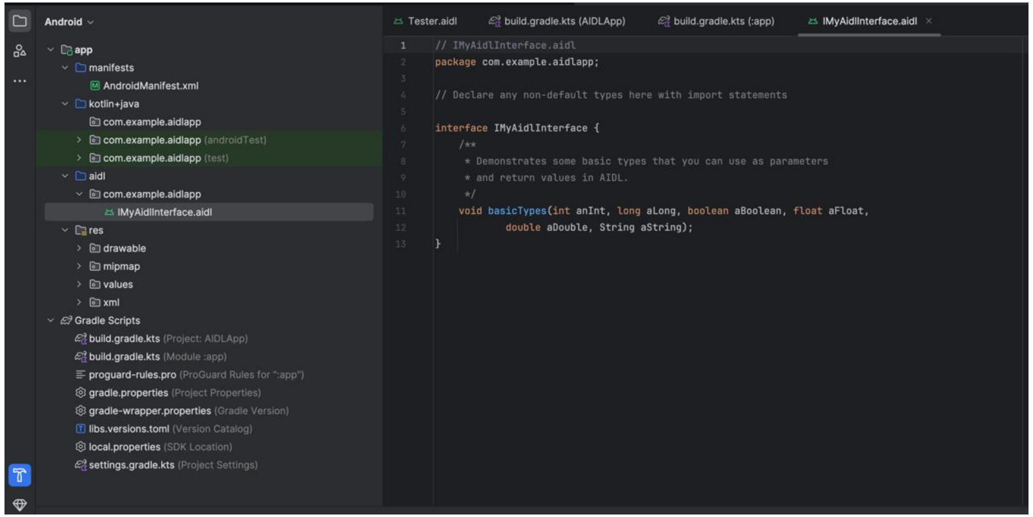
AIDL in Action



You can use the Project Structure dialog to view and edit your project configuration

```
android {
    ...faultConfig {
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    ...ildFeatures {
        aidl = true
    ...ildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile( name: "proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
```

```
...repareKotlinBuildScriptModel UP-TO-DATE

...CESSFUL in 841ms
```

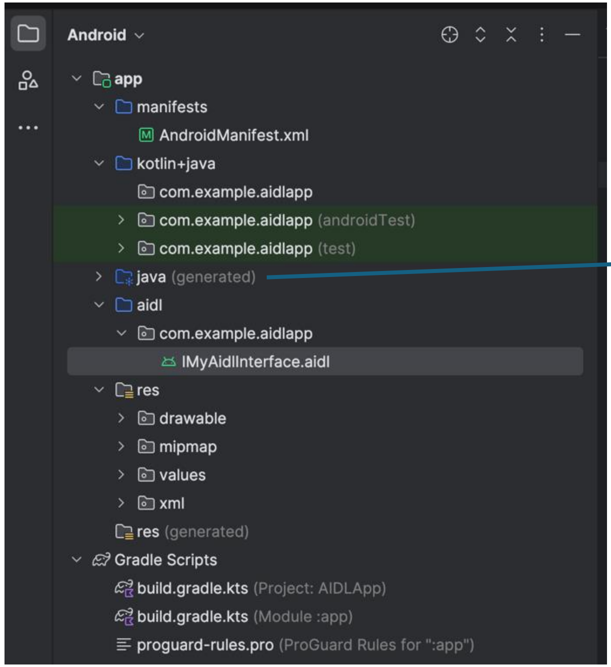

Android
Tester.aidl
build.gradle.kts (AIDLApp)
build.gradle.kts (:app)
IMyAidlInterface.aidl
app
manifests
AndroidManifest.xml
kotlin+java
com.example.aidlapp
com.example.aidlapp (androidTest)
com.example.aidlapp (test)
aidl
com.example.aidlapp
IMyAidlInterface.aidl
res
drawable
mipmap
values
xml
Gradle Scripts
build.gradle.kts (Project: AIDLApp)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)

```
// IMyAidlInterface.aidl
package com.example.aidlapp;

// Declare any non-default types here with import statements

interface IMyAidlInterface {
    /**
     * Demonstrates some basic types that you can use as parameters
     * and return values in AIDL.
     */
    void basicTypes(int anInt, long aLong, boolean aBoolean, float aFloat,
            double aDouble, String aString);
}
```

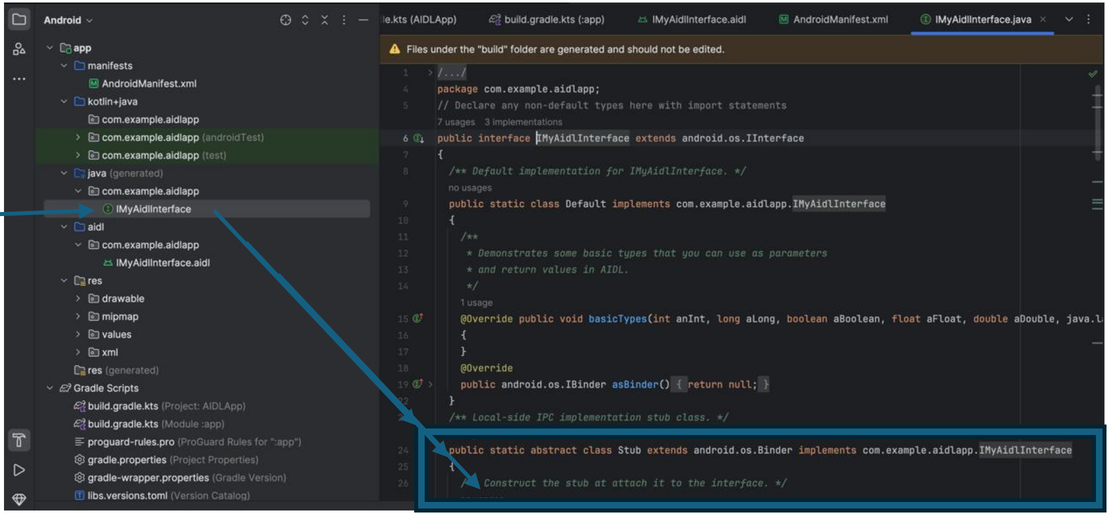
Defining .aidl

Example .aidl

AIDL in Action



```
Android
app
  manifests
    AndroidManifest.xml
  kotlin+java
    com.example.aidlapp
    com.example.aidlapp (androidTest)
    com.example.aidlapp (test)
  java (generated)
  aidl
    com.example.aidlapp
      IMyAidlInterface.aidl
  res
    drawable
    mipmap
    values
    xml
  res (generated)
Gradle Scripts
  build.gradle.kts (Project: AIDLApp)
  build.gradle.kts (Module :app)
  proguard-rules.pro (ProGuard Rules for ":app")
```



```
Android
app
  manifests
    AndroidManifest.xml
  kotlin+java
    com.example.aidlapp
    com.example.aidlapp (androidTest)
    com.example.aidlapp (test)
  java (generated)
    com.example.aidlapp
      IMyAidlInterface
  aidl
    com.example.aidlapp
      IMyAidlInterface.aidl
  res
    drawable
    mipmap
    values
    xml
  res (generated)
Gradle Scripts
  build.gradle.kts (Project: AIDLApp)
  build.gradle.kts (Module :app)
  proguard-rules.pro (ProGuard Rules for ":app")
  gradle.properties (Project Properties)
  gradle-wrapper.properties (Gradle Version)
  libs.versions.toml (Version Catalog)
```

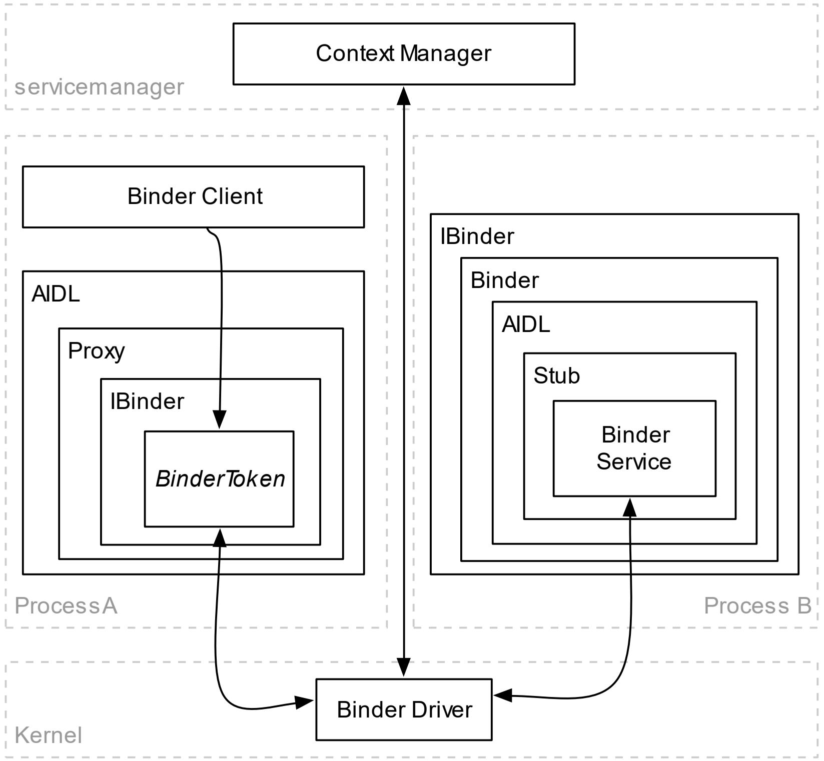
le.kts (AIDLApp) | build.gradle.kts (:app) | IMyAidlInterface.aidl | AndroidManifest.xml | IMyAidlInterface.java

⚠ Files under the "build" folder are generated and should not be edited.

```java
/.../
package com.example.aidlapp;
// Declare any non-default types here with import statements
7 usages  3 implementations
public interface IMyAidlInterface extends android.os.IInterface
{
  /** Default implementation for IMyAidlInterface. */
  no usages
  public static class Default implements com.example.aidlapp.IMyAidlInterface
  {
    /**
     * Demonstrates some basic types that you can use as parameters
     * and return values in AIDL.
     */
    1 usage
    @Override public void basicTypes(int anInt, long aLong, boolean aBoolean, float aFloat, double aDouble, java.l
    {
    }
    @Override
    public android.os.IBinder asBinder() { return null; }
  }
  /** Local-side IPC implementation stub class. */
  public static abstract class Stub extends android.os.Binder implements com.example.aidlapp.IMyAidlInterface
  {
    /** Construct the stub at attach it to the interface. */
```

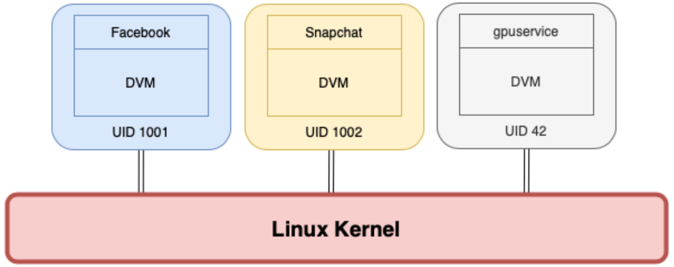
Stub created by the framework

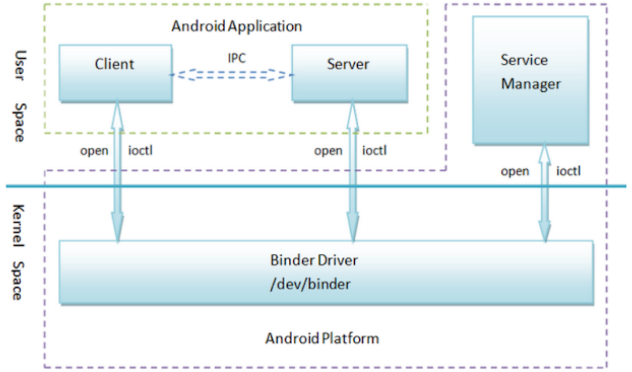
AIDL in Action



Android IPC Security Model

- **Security Principles:**
  - Permissions-based security for IPC.
  - Role of user IDs (UID) and SE Linux policies in restricting IPC access.
- **Security Features:**
  - Android permission model
  - Binder mechanism isolating services





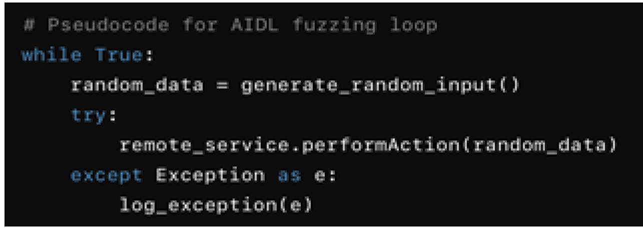
Common Attack Surfaces in Android IPC

Types of Vulnerabilities:

- Unauthorized access to system services
- Privilege escalation through IPC channels
- Data leakage between apps

Overview of Fuzzing & Why AIDL Fuzzing

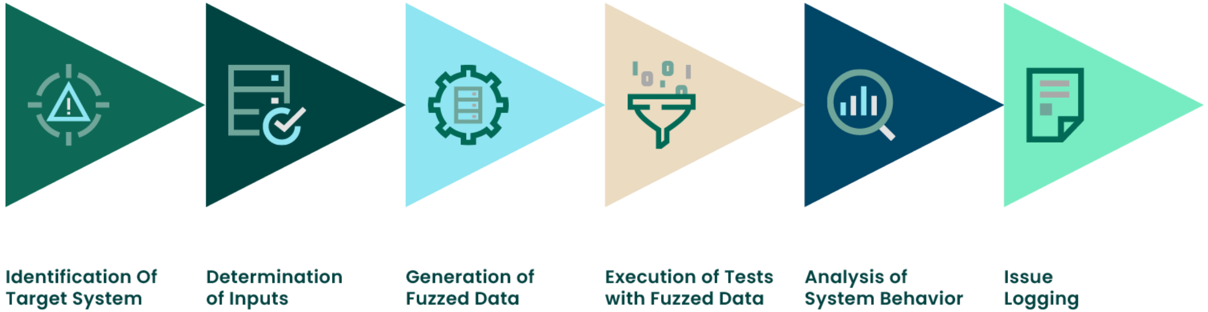
- **What is Fuzzing?**
- **Why focus on AIDL?**
  - The complexity of AIDL interfaces increases the attack surface.
  - Poorly secured AIDL interfaces can expose sensitive functionality.
- **Advantages of AIDL Fuzzing:**
  - Exposes deep-rooted issues in IPC systems.
  - Automates discovery of edge cases leading to crashes or leaks.



```
# Pseudocode for AIDL fuzzing loop
while True:
    random_data = generate_random_input()
    try:
        remote_service.performAction(random_data)
    except Exception as e:
        log_exception(e)
```

How AIDL Fuzzing Works

- Fuzzing Process:
  - Step-by-step breakdown of fuzzing AIDL interfaces.
  - Input generation, mutation, and monitoring results.

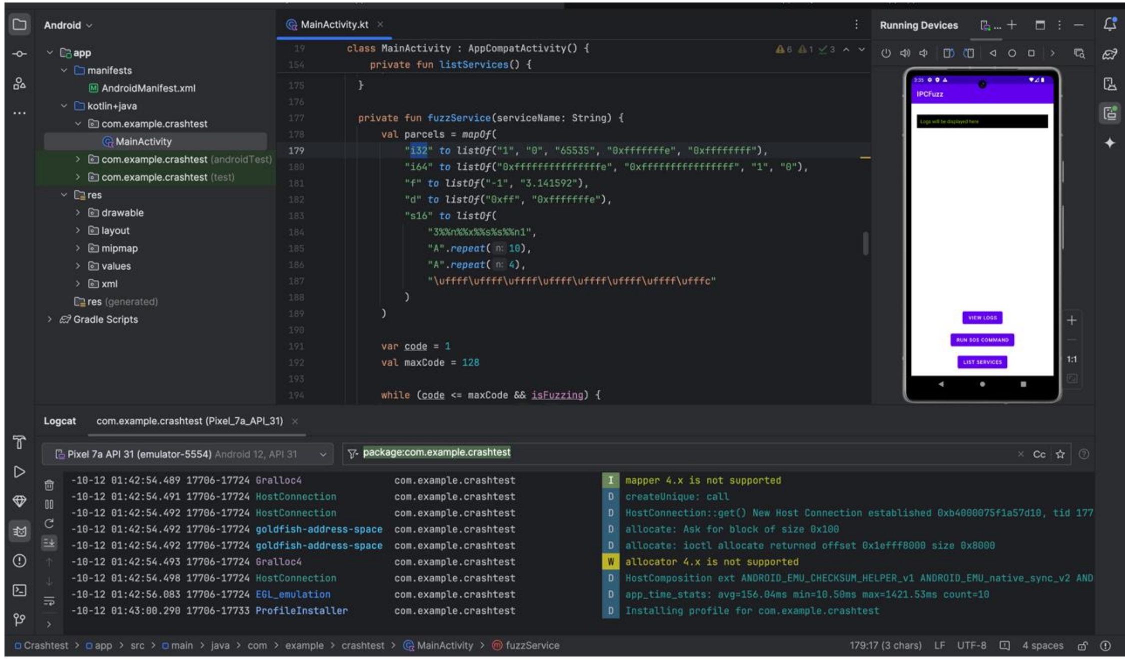

Identification Of Target System
Determination of Inputs
Generation of Fuzzed Data
Execution of Tests with Fuzzed Data
Analysis of System Behavior
Issue Logging

- Targeting AIDL:
  - Example: Choose an AIDL service to fuzz.
  - Creating inputs for defined methods in AIDL.

Setting Up AIDL Fuzzing

- **Requires Tools:**
  - ADB for device interaction.
  - Android Studio
  - Android Device
  - Service to fuzz

- **Setting up the Environment:**
  - Setup instructions

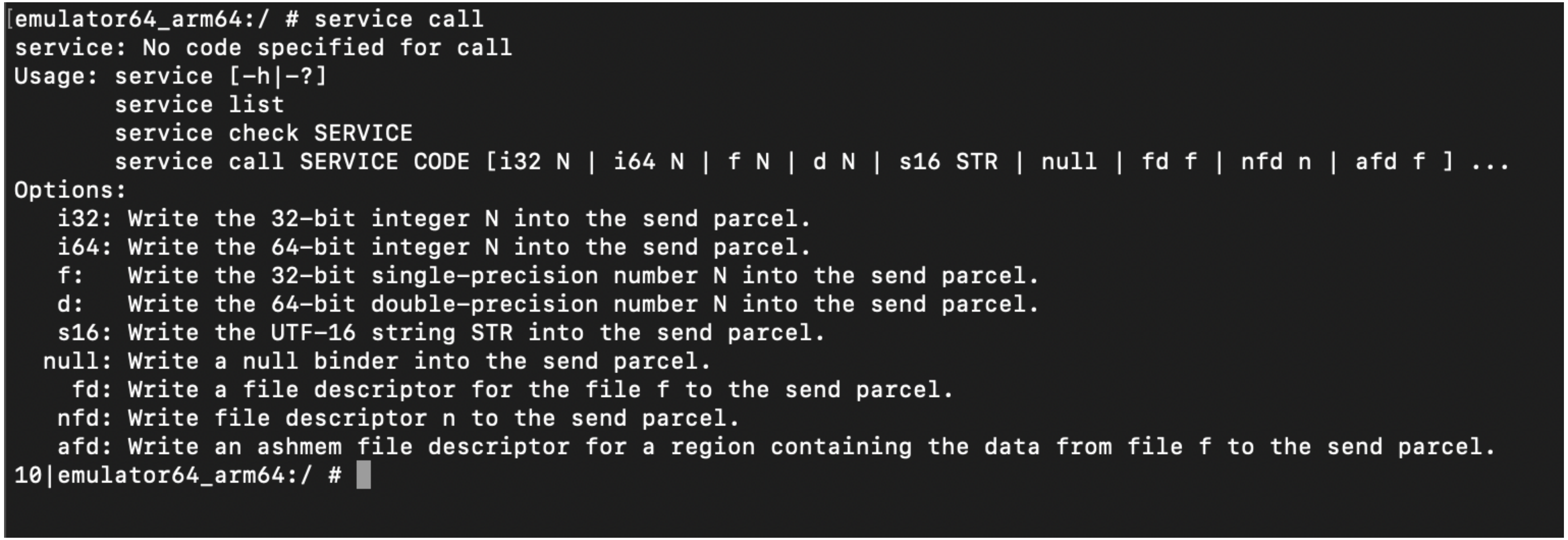


Android
app
manifests
AndroidManifest.xml
kotlin+java
com.example.crashtest
MainActivity
com.example.crashtest (androidTest)
com.example.crashtest (test)
res
drawable
layout
mipmap
values
xml
res (generated)
Gradle Scripts
MainActivity.kt
Running Devices
IPCFuzz
VIEW LOGS
RUN SOS COMMAND
LIST SERVICES
Logcat
com.example.crashtest (Pixel_7a_API_31)
Pixel 7a API 31 (emulator-5554) Android 12, API 31
package:com.example.crashtest
Crashtest > app > src > main > java > com > example > crashtest > MainActivity > fuzzService
179:17 (3 chars) LF UTF-8 4 spaces

```kotlin
class MainActivity : AppCompatActivity() {
    private fun listServices() {
    }

    private fun fuzzService(serviceName: String) {
        val parcels = mapOf(
            "i32" to listOf("1", "0", "65535", "0xfffffffe", "0xffffffff"),
            "i64" to listOf("0xfffffffffffffffe", "0xffffffffffffffff", "1", "0"),
            "f" to listOf("-1", "3.141592"),
            "d" to listOf("0xff", "0xfffffffe"),
            "s16" to listOf(
                "3%%n%%x%%s%s%%n1",
                "A".repeat( n: 10),
                "A".repeat( n: 4),
                "\uffff\uffff\uffff\uffff\uffff\uffff\uffff\ufffc"
            )
        )

        var code = 1
        val maxCode = 128

        while (code <= maxCode && isFuzzing) {
```

```
-10-12 01:42:54.489 17706-17724 Gralloc4              com.example.crashtest    I  mapper 4.x is not supported
-10-12 01:42:54.491 17706-17724 HostConnection        com.example.crashtest    D  createUnique: call
-10-12 01:42:54.492 17706-17724 HostConnection        com.example.crashtest    D  HostConnection::get() New Host Connection established 0xb4000075f1a57d10, tid 177
-10-12 01:42:54.492 17706-17724 goldfish-address-space com.example.crashtest   D  allocate: Ask for block of size 0x100
-10-12 01:42:54.492 17706-17724 goldfish-address-space com.example.crashtest   D  allocate: ioctl allocate returned offset 0x1efff8000 size 0x8000
-10-12 01:42:54.493 17706-17724 Gralloc4              com.example.crashtest    W  allocator 4.x is not supported
-10-12 01:42:54.498 17706-17724 HostConnection        com.example.crashtest    D  HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 AND
-10-12 01:42:56.083 17706-17724 EGL_emulation         com.example.crashtest    D  app_time_stats: avg=156.04ms min=10.50ms max=1421.53ms count=10
-10-12 01:43:00.290 17706-17733 ProfileInstaller      com.example.crashtest    D  Installing profile for com.example.crashtest
```

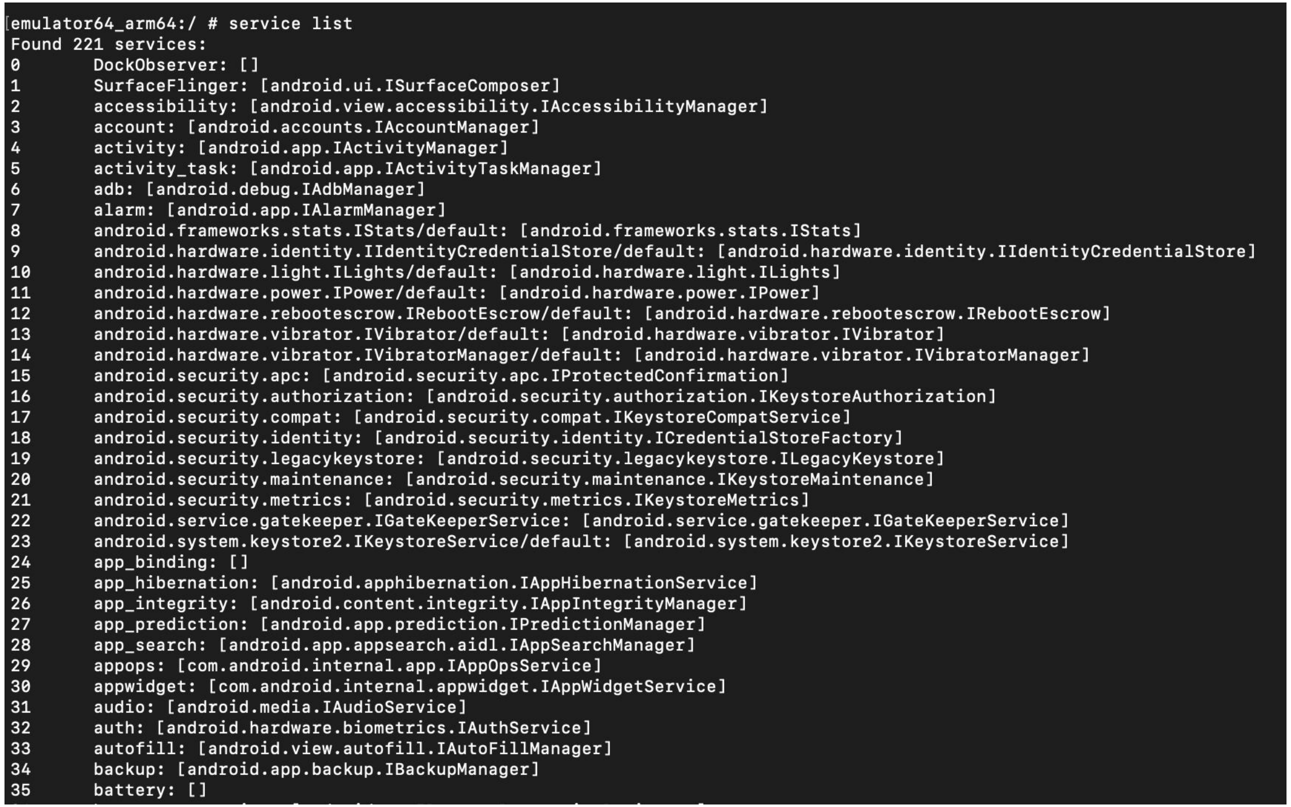
What to Fuzz.?



```
[emulator64_arm64:/ # service call
service: No code specified for call
Usage: service [-h|-?]
       service list
       service check SERVICE
       service call SERVICE CODE [i32 N | i64 N | f N | d N | s16 STR | null | fd f | nfd n | afd f ] ...
Options:
   i32: Write the 32-bit integer N into the send parcel.
   i64: Write the 64-bit integer N into the send parcel.
   f:   Write the 32-bit single-precision number N into the send parcel.
   d:   Write the 64-bit double-precision number N into the send parcel.
   s16: Write the UTF-16 string STR into the send parcel.
  null: Write a null binder into the send parcel.
    fd: Write a file descriptor for the file f to the send parcel.
   nfd: Write file descriptor n to the send parcel.
   afd: Write an ashmem file descriptor for a region containing the data from file f to the send parcel.
10|emulator64_arm64:/ # 
```

Calling a service : service call statusbar 1

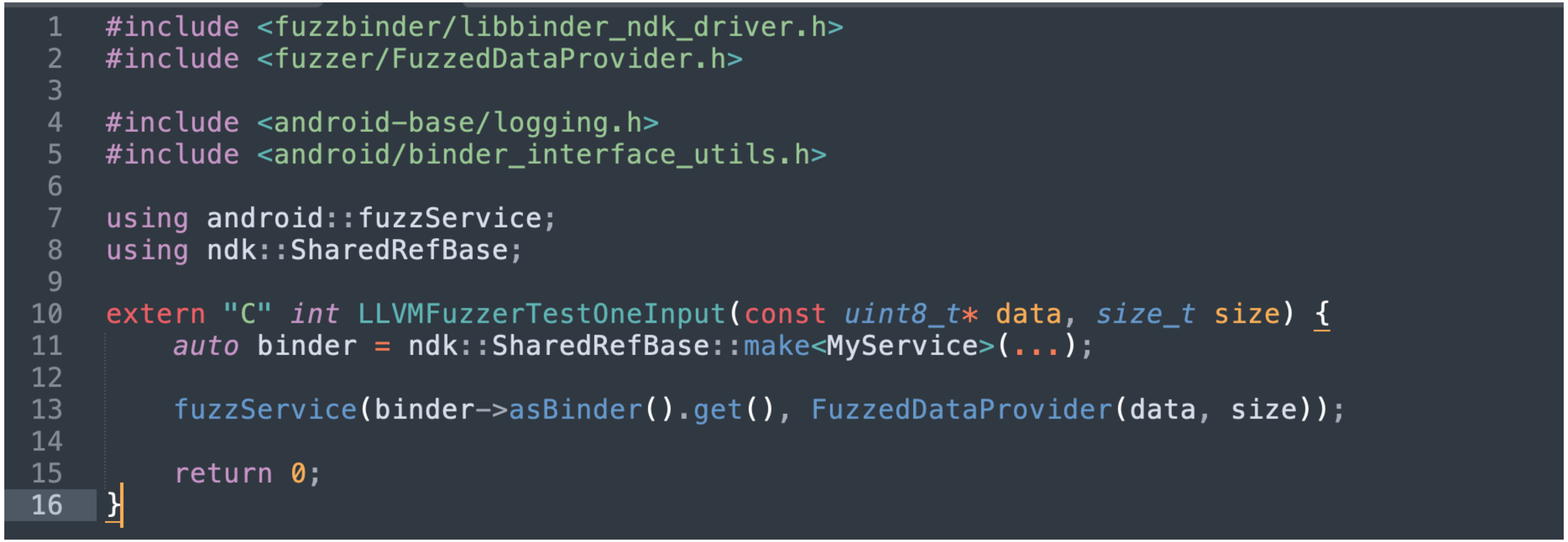
What to Fuzz.?



```
emulator64_arm64:/ # service list
Found 221 services:
0       DockObserver: []
1       SurfaceFlinger: [android.ui.ISurfaceComposer]
2       accessibility: [android.view.accessibility.IAccessibilityManager]
3       account: [android.accounts.IAccountManager]
4       activity: [android.app.IActivityManager]
5       activity_task: [android.app.IActivityTaskManager]
6       adb: [android.debug.IAdbManager]
7       alarm: [android.app.IAlarmManager]
8       android.frameworks.stats.IStats/default: [android.frameworks.stats.IStats]
9       android.hardware.identity.IIdentityCredentialStore/default: [android.hardware.identity.IIdentityCredentialStore]
10      android.hardware.light.ILights/default: [android.hardware.light.ILights]
11      android.hardware.power.IPower/default: [android.hardware.power.IPower]
12      android.hardware.rebootescrow.IRebootEscrow/default: [android.hardware.rebootescrow.IRebootEscrow]
13      android.hardware.vibrator.IVibrator/default: [android.hardware.vibrator.IVibrator]
14      android.hardware.vibrator.IVibratorManager/default: [android.hardware.vibrator.IVibratorManager]
15      android.security.apc: [android.security.apc.IProtectedConfirmation]
16      android.security.authorization: [android.security.authorization.IKeystoreAuthorization]
17      android.security.compat: [android.security.compat.IKeystoreCompatService]
18      android.security.identity: [android.security.identity.ICredentialStoreFactory]
19      android.security.legacykeystore: [android.security.legacykeystore.ILegacyKeystore]
20      android.security.maintenance: [android.security.maintenance.IKeystoreMaintenance]
21      android.security.metrics: [android.security.metrics.IKeystoreMetrics]
22      android.service.gatekeeper.IGateKeeperService: [android.service.gatekeeper.IGateKeeperService]
23      android.system.keystore2.IKeystoreService/default: [android.system.keystore2.IKeystoreService]
24      app_binding: []
25      app_hibernation: [android.apphibernation.IAppHibernationService]
26      app_integrity: [android.content.integrity.IAppIntegrityManager]
27      app_prediction: [android.app.prediction.IPredictionManager]
28      app_search: [android.app.appsearch.aidl.IAppSearchManager]
29      appops: [com.android.internal.app.IAppOpsService]
30      appwidget: [com.android.internal.appwidget.IAppWidgetService]
31      audio: [android.media.IAudioService]
32      auth: [android.hardware.biometrics.IAuthService]
33      autofill: [android.view.autofill.IAutoFillManager]
34      backup: [android.app.backup.IBackupManager]
35      battery: []
```

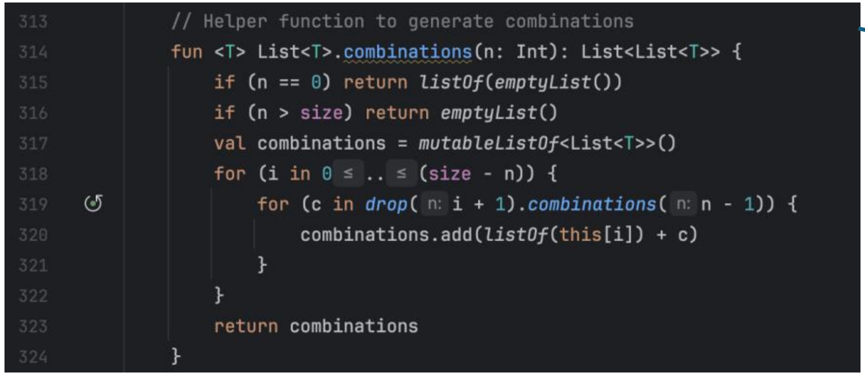
Example Fuzzing Code

- **Fuzzing Code Sample:**

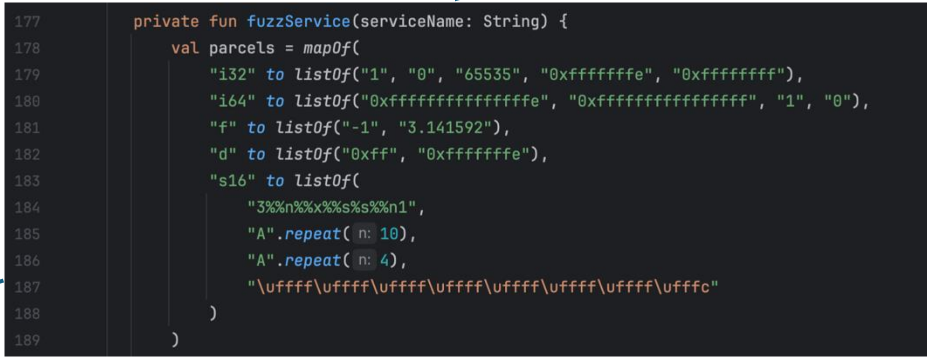


```cpp
#include <fuzzbinder/libbinder_ndk_driver.h>
#include <fuzzer/FuzzedDataProvider.h>

#include <android-base/logging.h>
#include <android/binder_interface_utils.h>

using android::fuzzService;
using ndk::SharedRefBase;

extern "C" int LLVMFuzzerTestOneInput(const uint8_t* data, size_t size) {
    auto binder = ndk::SharedRefBase::make<MyService>(...);

    fuzzService(binder->asBinder().get(), FuzzedDataProvider(data, size));

    return 0;
}
```

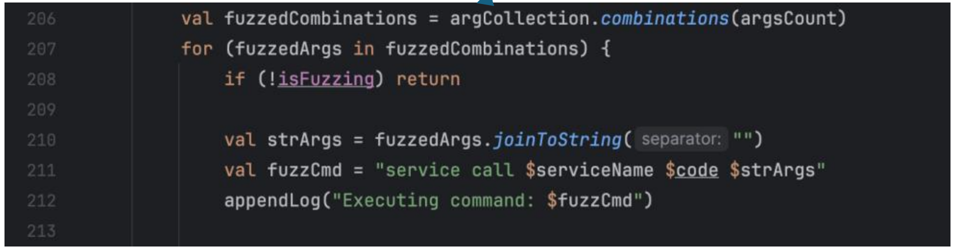
Our Fuzzing Code



```kotlin
// Helper function to generate combinations
fun <T> List<T>.combinations(n: Int): List<List<T>> {
    if (n == 0) return listOf(emptyList())
    if (n > size) return emptyList()
    val combinations = mutableListOf<List<T>>()
    for (i in 0 ≤ .. ≤ (size - n)) {
        for (c in drop( n: i + 1).combinations( n: n - 1)) {
            combinations.add(listOf(this[i]) + c)
        }
    }
    return combinations
}
```



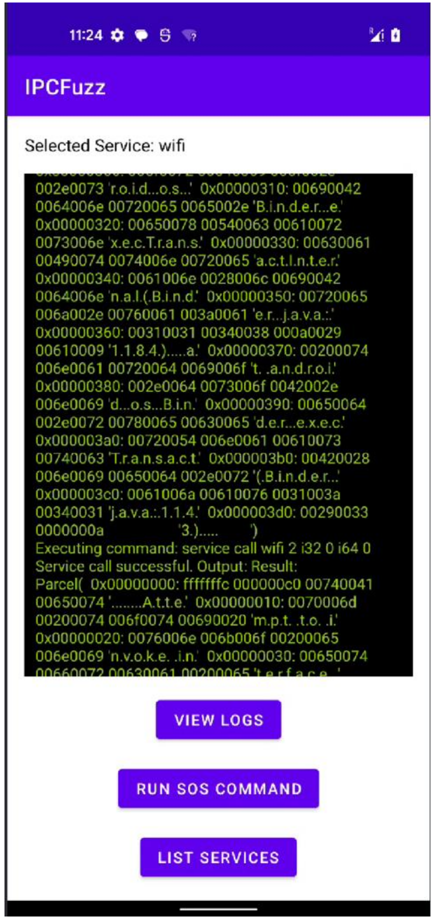
```kotlin
private fun fuzzService(serviceName: String) {
    val parcels = mapOf(
        "i32" to listOf("1", "0", "65535", "0xfffffffe", "0xffffffff"),
        "i64" to listOf("0xfffffffffffffffe", "0xffffffffffffffff", "1", "0"),
        "f" to listOf("-1", "3.141592"),
        "d" to listOf("0xff", "0xfffffffe"),
        "s16" to listOf(
            "3%%n%%x%%s%s%%n1",
            "A".repeat( n: 10),
            "A".repeat( n: 4),
            "\uffff\uffff\uffff\uffff\uffff\uffff\uffff\ufffc"
        )
    )
```



```kotlin
    val fuzzedCombinations = argCollection.combinations(argsCount)
    for (fuzzedArgs in fuzzedCombinations) {
        if (!isFuzzing) return

        val strArgs = fuzzedArgs.joinToString( separator: "")
        val fuzzCmd = "service call $serviceName $code $strArgs"
        appendLog("Executing command: $fuzzCmd")

```

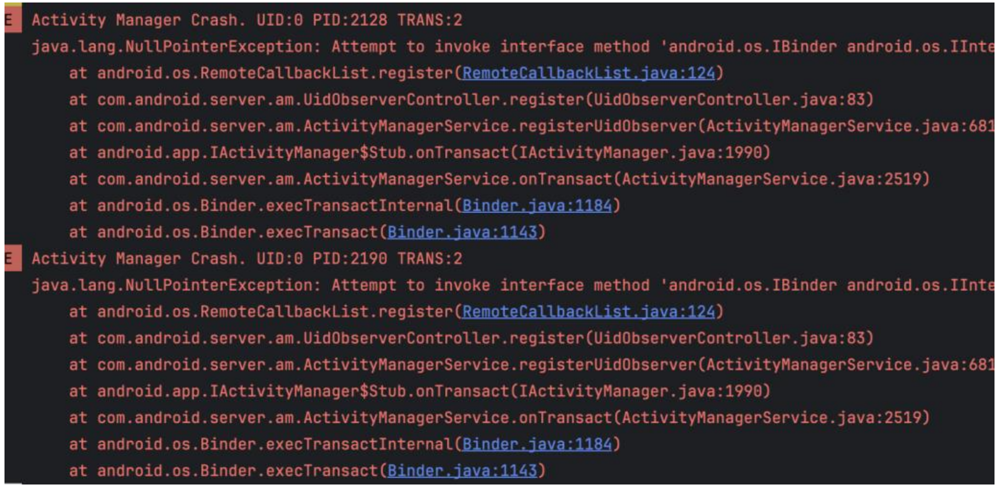
Demo

- **Fuzzing Demo Overview:**
  - Quick look at the tools and setup.
  - Choose a service to fuzz.
  - Executing fuzzing and capturing results.

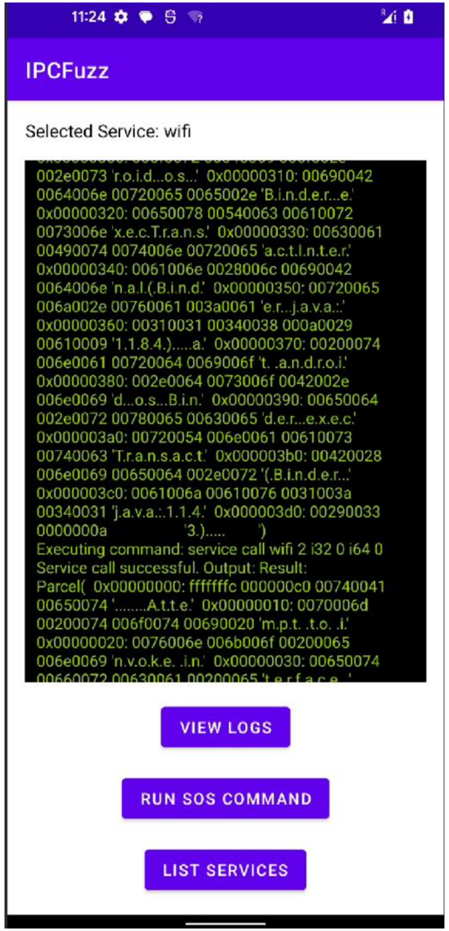


Selected Service: wifi

```
002e0073 'r.o.i.d...o.s...'  0x00000310: 00690042
0064006e 00720065 0065002e 'B.i.n.d.e.r...e.'
0x00000320: 00650078 00540063 00610072
0073006e 'x.e.c.T.r.a.n.s.'  0x00000330: 00630061
00490074 0074006e 00720065 'a.c.t.I.n.t.e.r.'
0x00000340: 0061006e 0028006c 00690042
0064006e 'n.a.l.(.B.i.n.d.'  0x00000350: 00720065
006a002e 00760061 003a0061 'e.r...j.a.v.a.:.'
0x00000360: 00310031 00340038 000a0029
00610009 '1.1.8.4.)......a.'  0x00000370: 00200074
006e0061 00720064 0069006f 't. .a.n.d.r.o.i.'
0x00000380: 002e0064 0073006f 0042002e
006e0069 'd...o.s...B.i.n.'  0x00000390: 00650064
002e0072 00780065 00630065 'd.e.r...e.x.e.c.'
0x000003a0: 00720054 006e0061 00610073
00740063 'T.r.a.n.s.a.c.t.'  0x000003b0: 00420028
006e0069 00650064 002e0072 '(.B.i.n.d.e.r...'
0x000003c0: 0061006a 00610076 0031003a
00340031 'j.a.v.a.:.1.1.4.'  0x000003d0: 00290033
0000000a                    '3.).....        ')
Executing command: service call wifi 2 i32 0 i64 0
Service call successful. Output: Result:
Parcel(  0x00000000: fffffffc 000000c0 00740041
00650074 '........A.t.t.e.'  0x00000010: 0070006d
00200074 006f0074 00690020 'm.p.t. .t.o. .i.'
0x00000020: 0076006e 006b006f 00200065
006e0069 'n.v.o.k.e. .i.n.'  0x00000030: 00650074
```



```
E  Activity Manager Crash. UID:0 PID:2128 TRANS:2
   java.lang.NullPointerException: Attempt to invoke interface method 'android.os.IBinder android.os.IInte
       at android.os.RemoteCallbackList.register(RemoteCallbackList.java:124)
       at com.android.server.am.UidObserverController.register(UidObserverController.java:83)
       at com.android.server.am.ActivityManagerService.registerUidObserver(ActivityManagerService.java:681
       at android.app.IActivityManager$Stub.onTransact(IActivityManager.java:1990)
       at com.android.server.am.ActivityManagerService.onTransact(ActivityManagerService.java:2519)
       at android.os.Binder.execTransactInternal(Binder.java:1184)
       at android.os.Binder.execTransact(Binder.java:1143)
E  Activity Manager Crash. UID:0 PID:2190 TRANS:2
   java.lang.NullPointerException: Attempt to invoke interface method 'android.os.IBinder android.os.IInte
       at android.os.RemoteCallbackList.register(RemoteCallbackList.java:124)
       at com.android.server.am.UidObserverController.register(UidObserverController.java:83)
       at com.android.server.am.ActivityManagerService.registerUidObserver(ActivityManagerService.java:681
       at android.app.IActivityManager$Stub.onTransact(IActivityManager.java:1990)
       at com.android.server.am.ActivityManagerService.onTransact(ActivityManagerService.java:2519)
       at android.os.Binder.execTransactInternal(Binder.java:1184)
       at android.os.Binder.execTransact(Binder.java:1143)
```



Selected Service: wifi

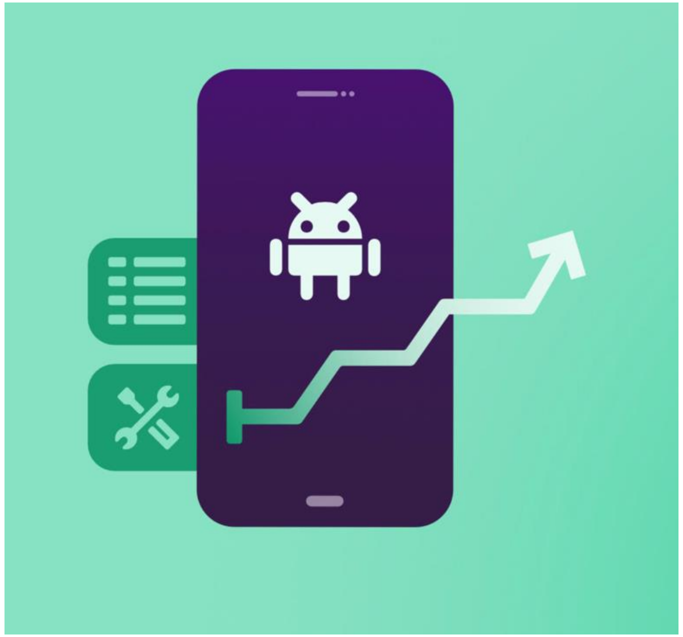
```
002e0073 'r.o.i.d...o.s...'  0x00000310: 00690042
0064006e 00720065 0065002e 'B.i.n.d.e.r...e.'
0x00000320: 00650078 00540063 00610072
0073006e 'x.e.c.T.r.a.n.s.'  0x00000330: 00630061
00490074 0074006e 00720065 'a.c.t.I.n.t.e.r.'
0x00000340: 0061006e 0028006c 00690042
0064006e 'n.a.l.(.B.i.n.d.'  0x00000350: 00720065
006a002e 00760061 003a0061 'e.r...j.a.v.a.:.'
0x00000360: 00310031 00340038 000a0029
00610009 '1.1.8.4.).....a.'  0x00000370: 00200074
006e0061 00720064 0069006f 't. .a.n.d.r.o.i.'
0x00000380: 002e0064 0073006f 0042002e
006e0069 'd...o.s...B.i.n.'  0x00000390: 00650064
002e0072 00780065 00630065 'd.e.r...e.x.e.c.'
0x000003a0: 00720054 006e0061 00610073
00740063 'T.r.a.n.s.a.c.t.'  0x000003b0: 00420028
006e0069 00650064 002e0072 '(.B.i.n.d.e.r...'
0x000003c0: 0061006a 00610076 0031003a
00340031 'j.a.v.a.:.1.1.4.'  0x000003d0: 00290033
0000000a                    '3.).....        ')
Executing command: service call wifi 2 i32 0 i64 0
Service call successful. Output: Result:
Parcel(  0x00000000: fffffffc 000000c0 00740041
00650074 '........A.t.t.e.'  0x00000010: 0070006d
00200074 006f0074 00690020 'm.p.t. .t.o. .i.'
0x00000020: 0076006e 006b006f 00200065
006e0069 'n.v.o.k.e. .i.n.'  0x00000030: 00650074
```

Demo: Running AIDL Fuzzing

- **Running the fuzzer:**
  - Show fuzzing in action using ADB and logcat.
  - Real-time output: Crashes, exceptions and anomalies
  - How to interpret logs and identify vulnerabilities.

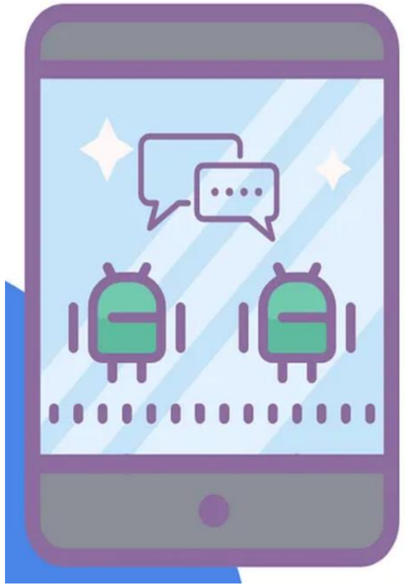
Challenges in AIDL Fuzzing

- **Hurdles**
  - Handling complex data structures in AIDL interfaces.
  - Dealing with permissions restrictions and sandboxing.
- **Solutions**
  - Crafting specialized inputs.
  - Bypassing IPC restrictions for testing



Securing Android IPC: Best Practices

- **Mitigation Strategies:**
  - Secure AIDL interface design: least privilege principle.
  - Input validation for AIDL methods.
- **Strengthening IPC Mechanisms:**
  - Using strong permissions and SELinux policies.
  - Regular fuzzing and vulnerability assessments.



Q&A

"Fuzzing is like a box of chocolates: you never know what you're going to get.”
- Charlie Miller

Thank you!