Breaking into Android IPC Mechanisms through Advanced AIDL Fuzzing

Rajanish Pathak & Hardik Mehta

HACK.LU
2025
BREAK. BUILD. SHARE.

About Us

Hardik Mehta

@hardw00t

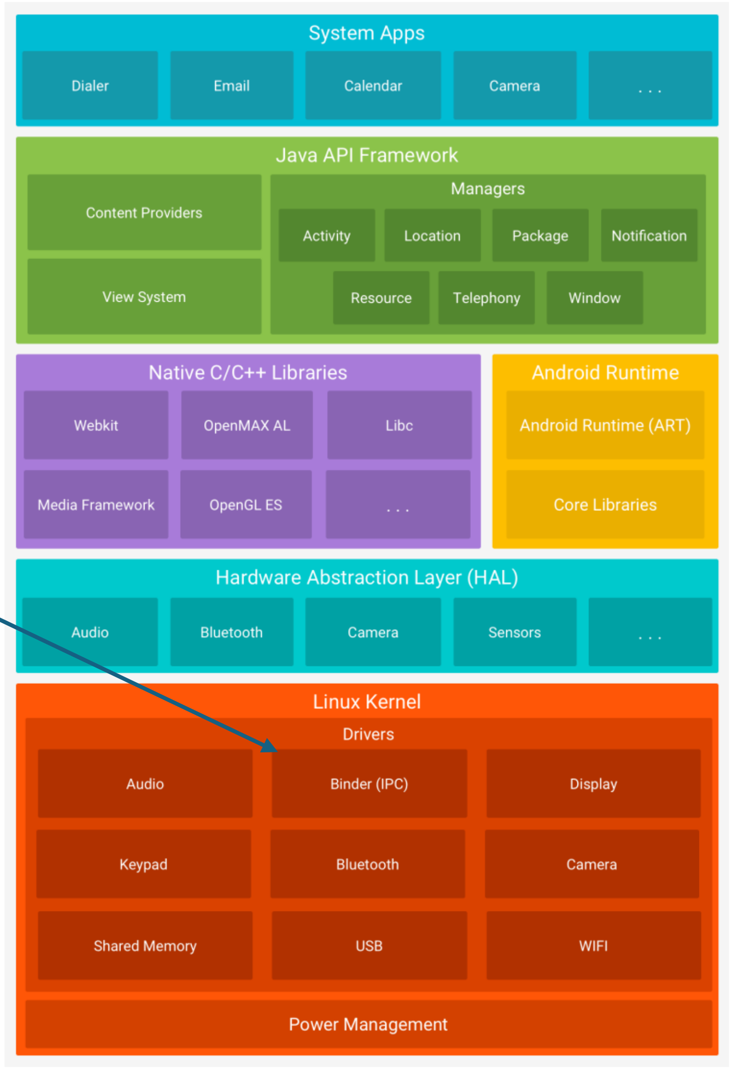
Rajanish Pathak

@h4ckologic

Agenda

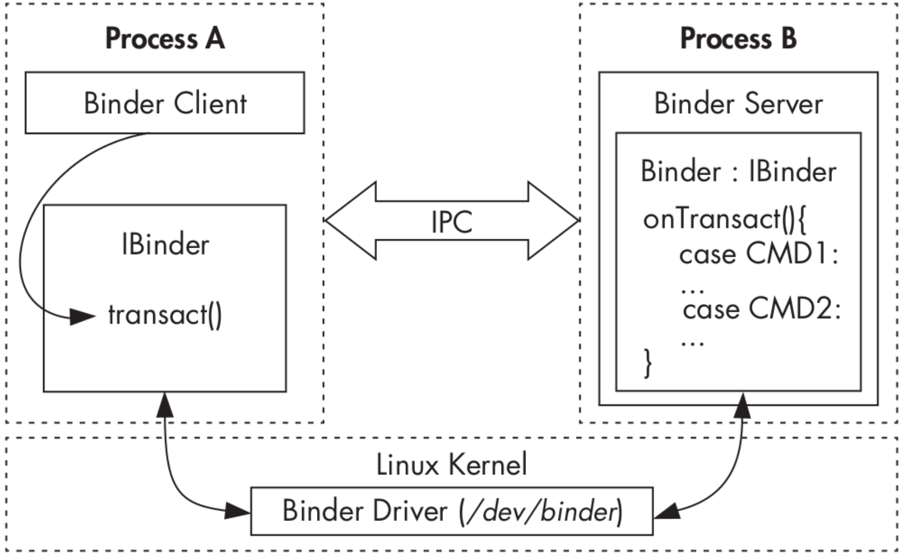
- Overview of Android IPC
- Understanding AIDL in Android
- Security Challenges in Android IPC
- Introduction to AIDL Fuzzing
- Tools and Frameworks for AIDL Fuzzing
- Live AIDL Fuzzing Demo
- Mitigating IPC Vulnerabilities
- Q&A

Overview of Android's IPC Mechanisms



Overview of Android's IPC Mechanisms

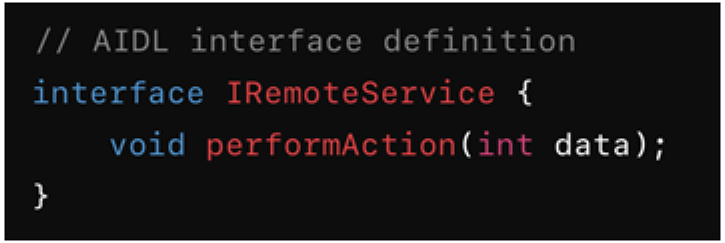
- **What is IPC in Android?**
  - IPC enables communication between processes (e.g., services, activities)
- **Android IPC Mechanisms:**
  - Binders (Kernel-level)
  - Intents
  - AIDL for complex IPC.
- **Why IPC Security Matters:**
  - Attack surfaces between trusted and untrusted processes.



Android Interface Definition Language (AIDL)

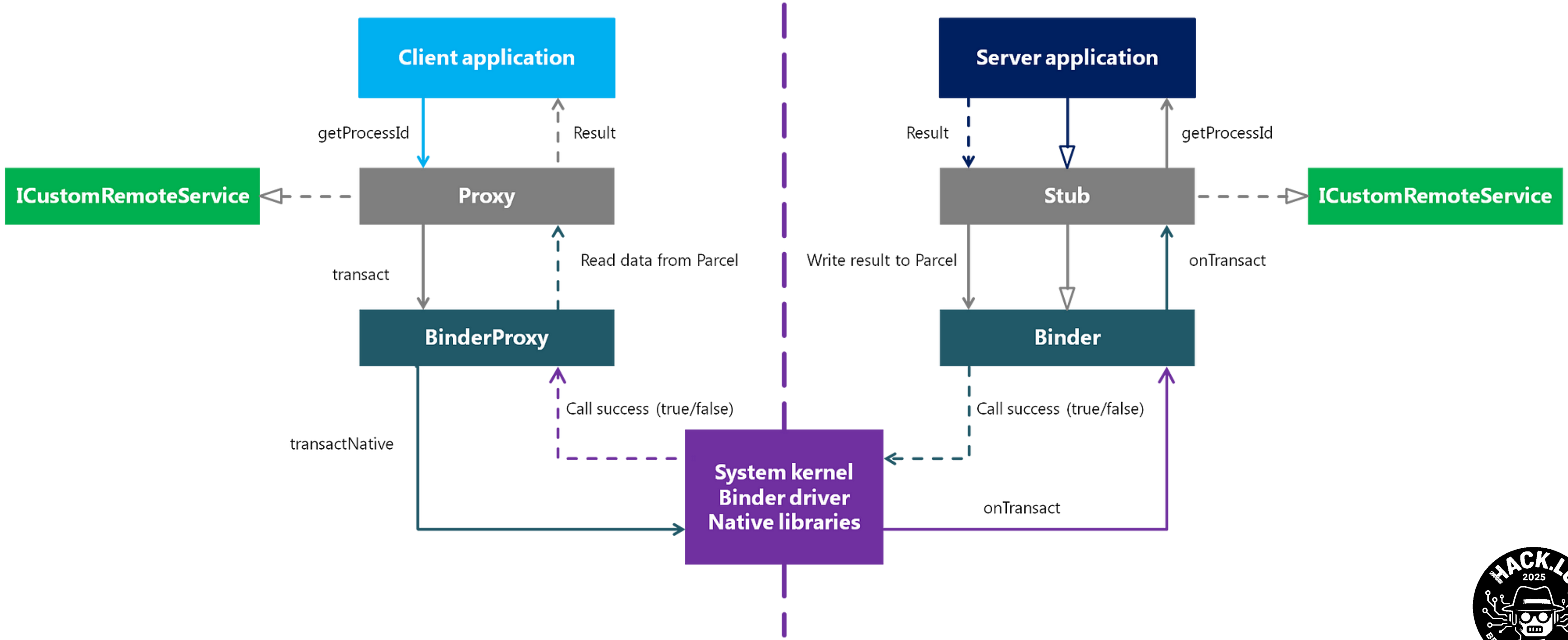
- **What is AIDL?**
  - AIDL allows processes to communicate with each other using defined interfaces.
  - Role in complex applications (e.g., system services, background apps).

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



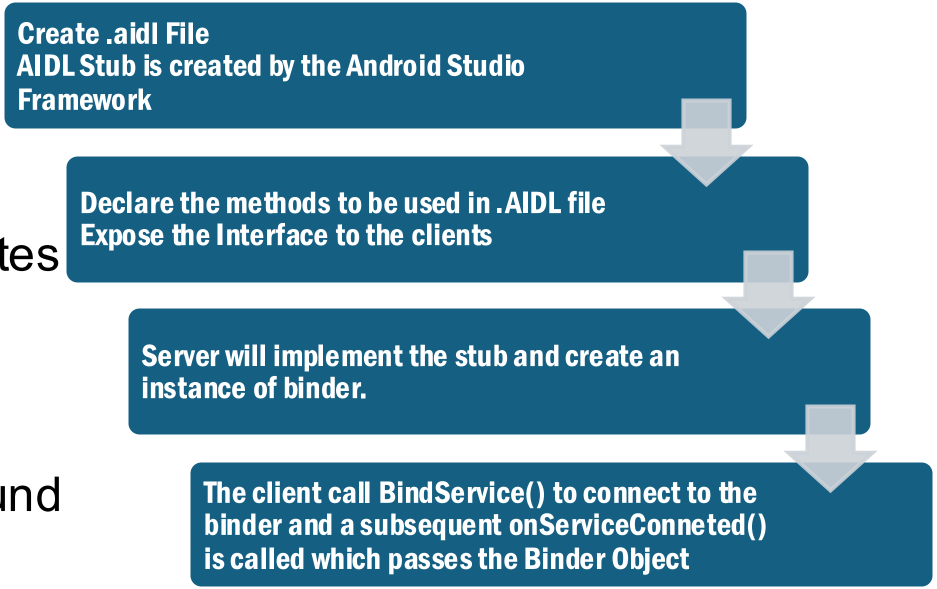
```
// AIDL interface definition
interface IRemoteService {
    void performAction(int data);
}
```

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.



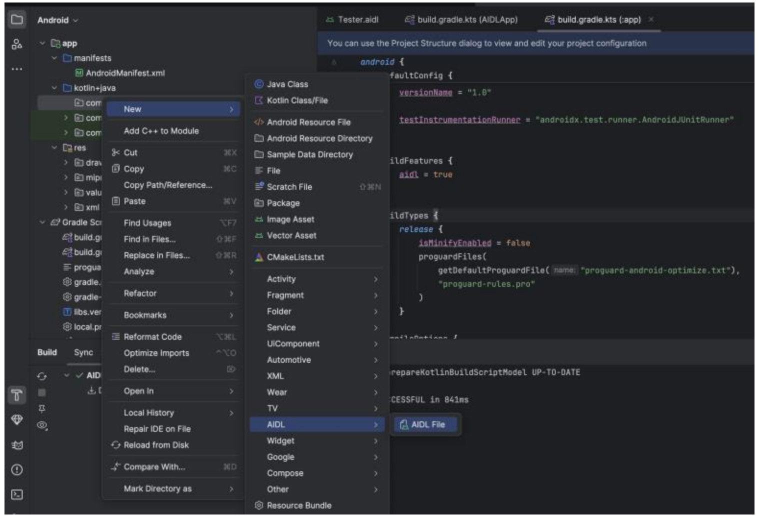
Create .aidl File
AIDL Stub is created by the Android Studio Framework

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

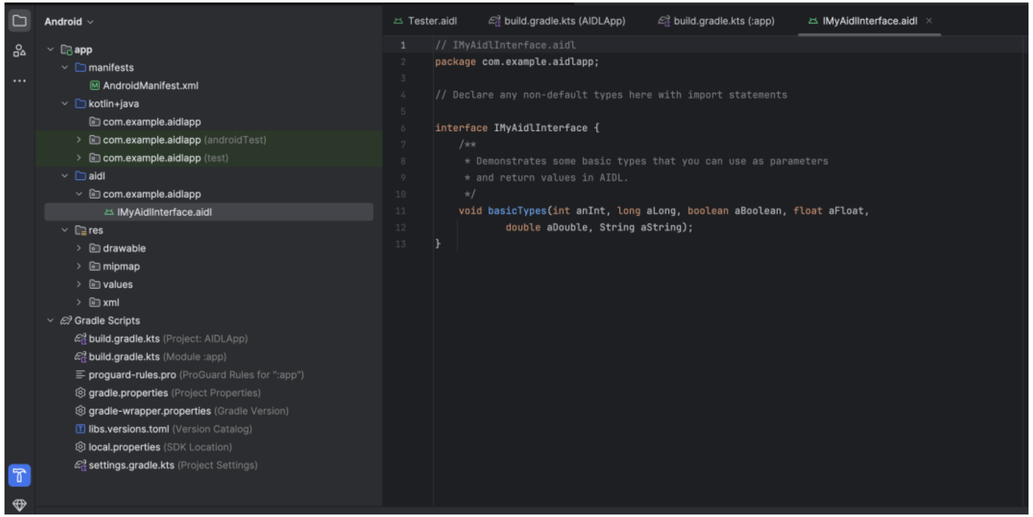
AIDL in Action



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

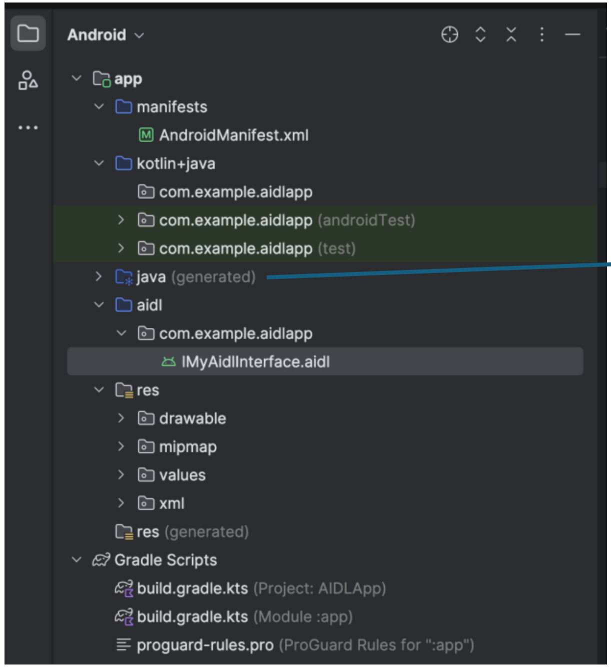
```
android {
        faultConfig {
        versionName = "1.0"

        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
```

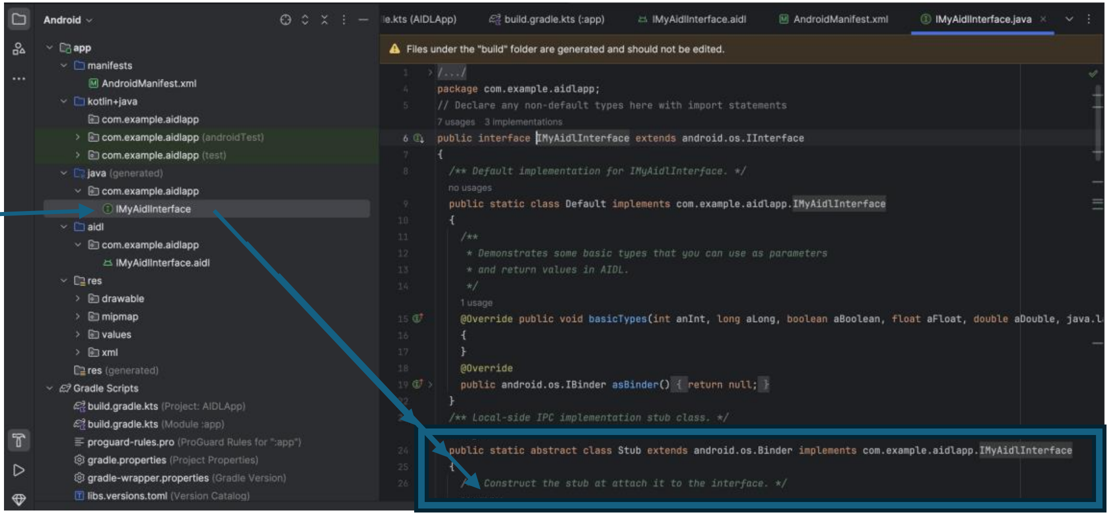

Android
app
manifests
AndroidManifest.xml
kotlin+java
com.example.aidlapp
com.example.aidlapp (androidTest)
com.example.aidlapp (test)
aidl
com.example.aidlapp
IMyAidlInterface.aidl
res
drawable
mipmap
values
xml
Gradle Scripts
build.gradle.kts (Project: AIDLApp)
build.gradle.kts (Module :app)
proguard-rules.pro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

```
// 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);
}
```

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")
```



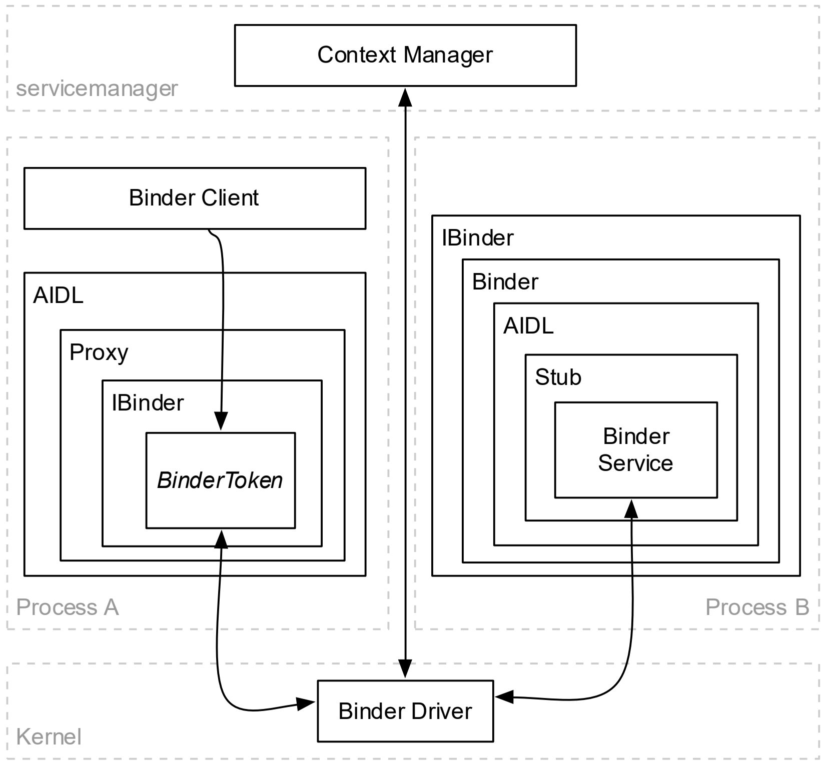
```
Android
app
  manifests
    AndroidManifest.xml
  kotlin+java
    com.example.aidlapp
    com.example.aidlapp (androidTest)
    com.example.aidlapp (test)
  java (generated)
    com.example.aidlapp
      IMyAidlInterface
  aidl
    com.example.aidlapp
      IMyAidlInterface.aidl
  res
    drawable
    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.

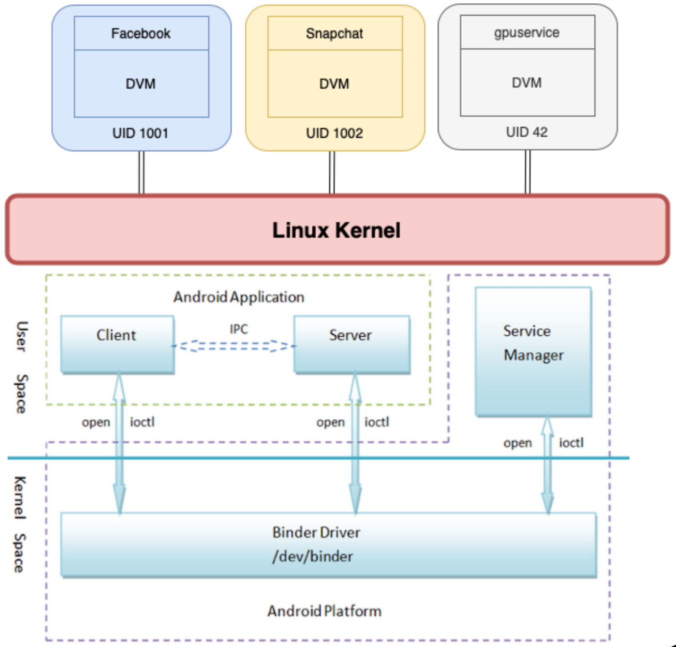
```java
/.../
package com.example.aidlapp;
// Declare any non-default types here with import statements
7 usages   3 implementations
public interface IMyAidlInterface extends android.os.IInterface
{
  /** Default implementation for IMyAidlInterface. */
  no usages
  public static class Default implements 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. */
```

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



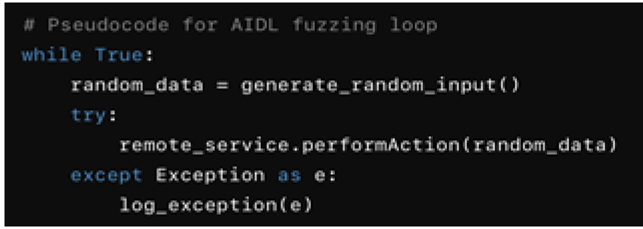
Common Attack Surfaces in Android IPC

Types of Vulnerabilities:

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

Overview of Fuzzing & Why AIDL Fuzzing

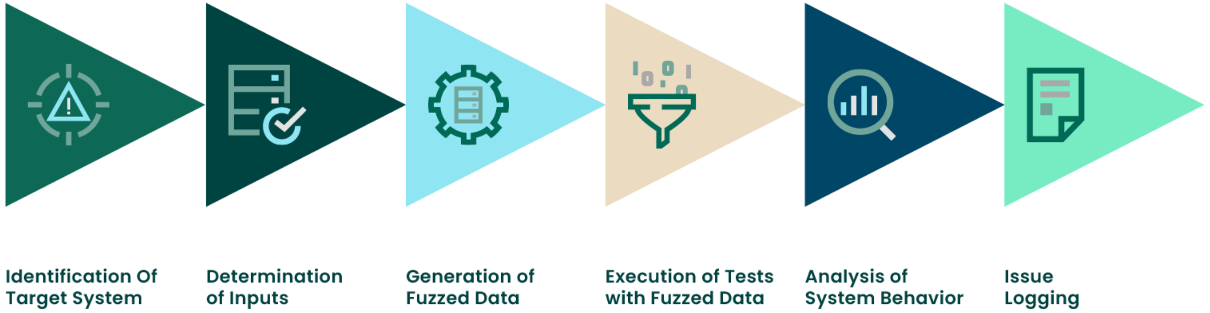
- **What is Fuzzing?**
- **Why focus on AIDL?**
  - The complexity of AIDL interfaces increases the attack surface.
  - Poorly secured AIDL interfaces can expose sensitive functionality.
- **Advantages of AIDL Fuzzing:**
  - Exposes deep-rooted issues in IPC systems.
  - Automates discovery of edge cases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



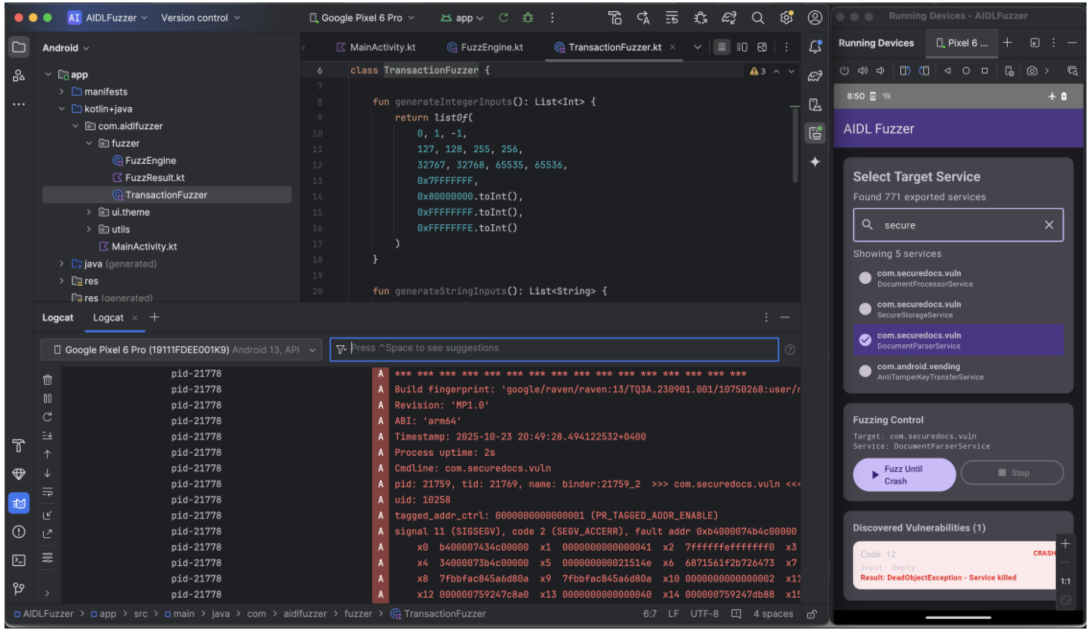
Identification Of Target System
Determination of Inputs
Generation of Fuzzed Data
Execution of Tests with Fuzzed Data
Analysis of System Behavior
Issue Logging

- 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



```
class TransactionFuzzer {

    fun generateIntegerInputs(): List<Int> {
        return listOf(
            0, 1, -1,
            127, 128, 255, 256,
            32767, 32768, 65535, 65536,
            0x7FFFFFFF,
            0x80000000.toInt(),
            0xFFFFFFFF.toInt(),
            0xFFFFFFFE.toInt()
        )
    }

    fun generateStringInputs(): List<String> {
```

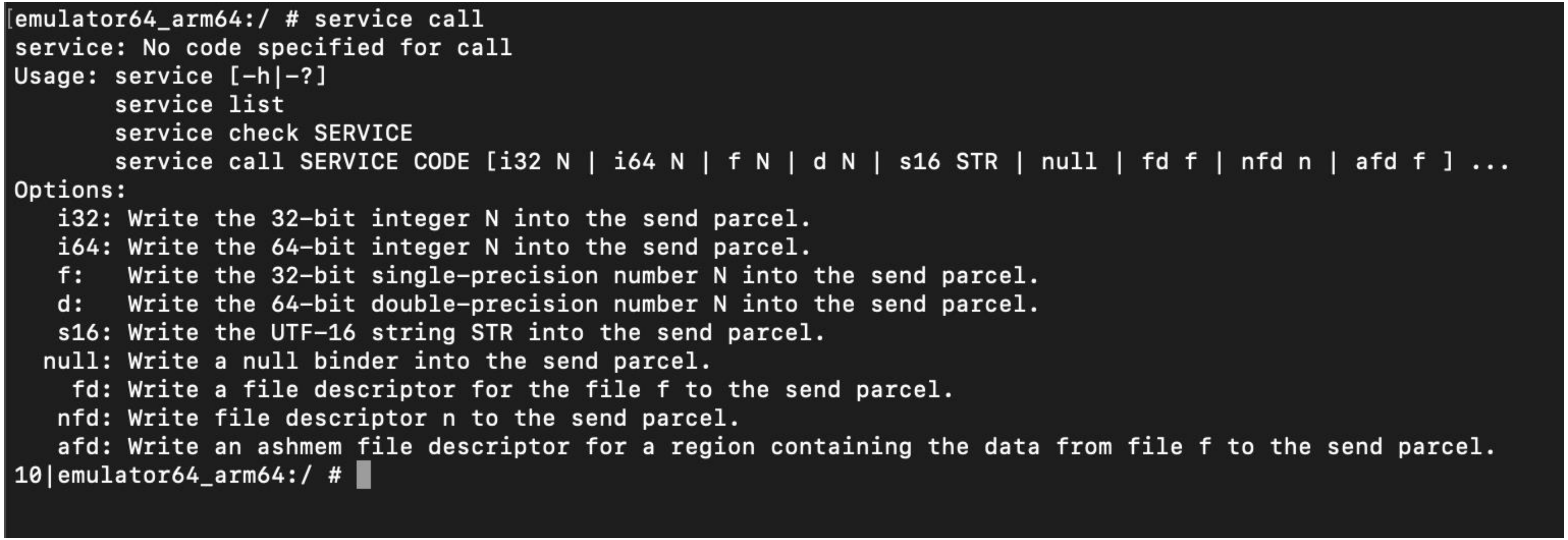
```
pid-21778    A  *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
pid-21778    A  Build fingerprint: 'google/raven/raven:13/TQ3A.230901.001/10750268:user/r
pid-21778    A  Revision: 'MP1.0'
pid-21778    A  ABI: 'arm64'
pid-21778    A  Timestamp: 2025-10-23 20:49:28.494122532+0400
pid-21778    A  Process uptime: 2s
pid-21778    A  Cmdline: com.securedocs.vuln
pid-21778    A  pid: 21759, tid: 21769, name: binder:21759_2  >>> com.securedocs.vuln <<
pid-21778    A  uid: 10258
pid-21778    A  tagged_addr_ctrl: 0000000000000001 (PR_TAGGED_ADDR_ENABLE)
pid-21778    A  signal 11 (SIGSEGV), code 2 (SEGV_ACCERR), fault addr 0xb4000074b4c00000
pid-21778    A      x0  b400007434c00000  x1  0000000000000041  x2  7ffffffeffffff0  x3
pid-21778    A      x4  34000073b4c00000  x5  000000000021514e  x6  6871561f2b726473  x7
pid-21778    A      x8  7fbbfac845a6d80a  x9  7fbbfac845a6d80a  x10 0000000000000002  x1
pid-21778    A      x12 000000759247c8a0  x13 0000000000000040  x14 000000759247db88  x1
```

Select Target Service
Found 771 exported services
secure
Showing 5 services
- com.securedocs.vuln DocumentProcessorService
- com.securedocs.vuln SecureStorageService
- [✓ com.securedocs.vuln DocumentParserService]
- com.android.vending AntiTamperKeyTransferService

Fuzzing Control
Target: com.securedocs.vuln
Service: DocumentParserService
Fuzz Until Crash | Stop

Discovered Vulnerabilities (1)
Code 12 CRASH
Input: Empty
Result: DeadObjectException - Service killed

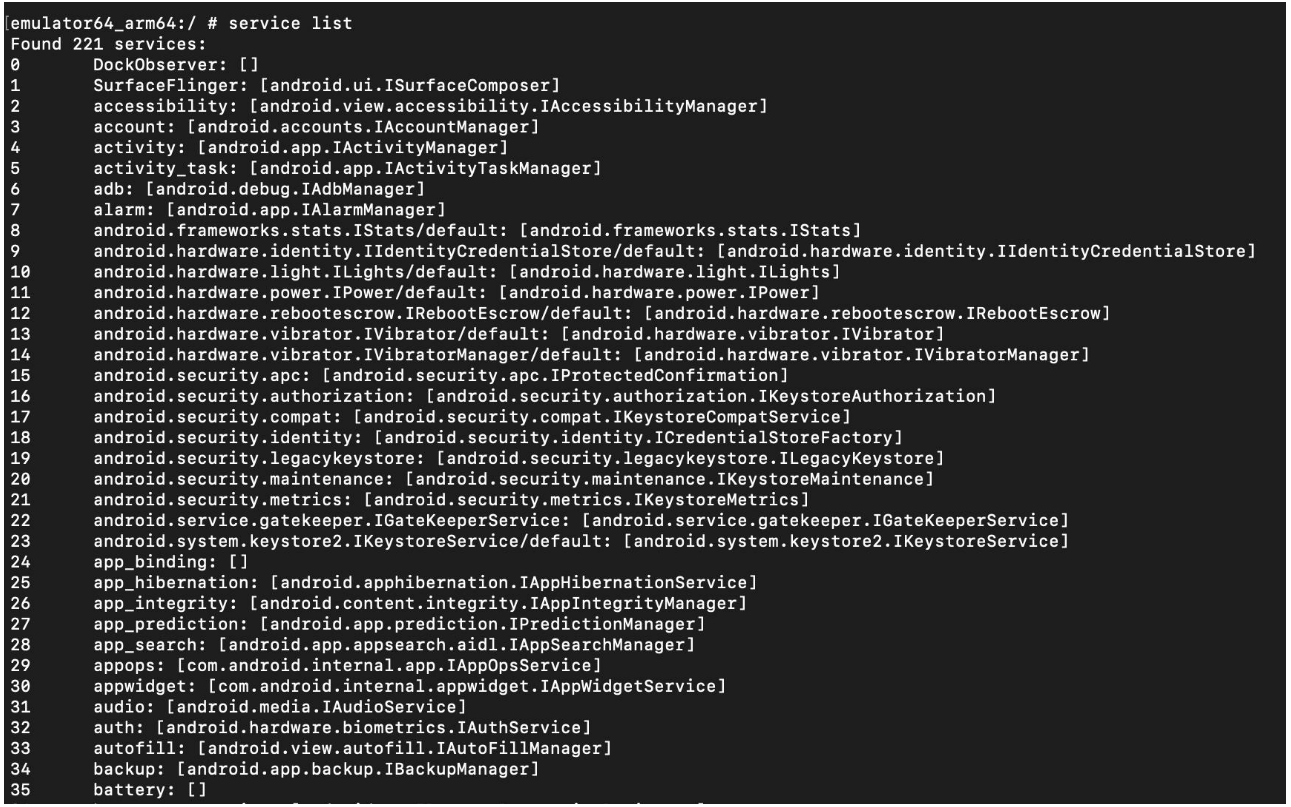
What to Fuzz.?



```
[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

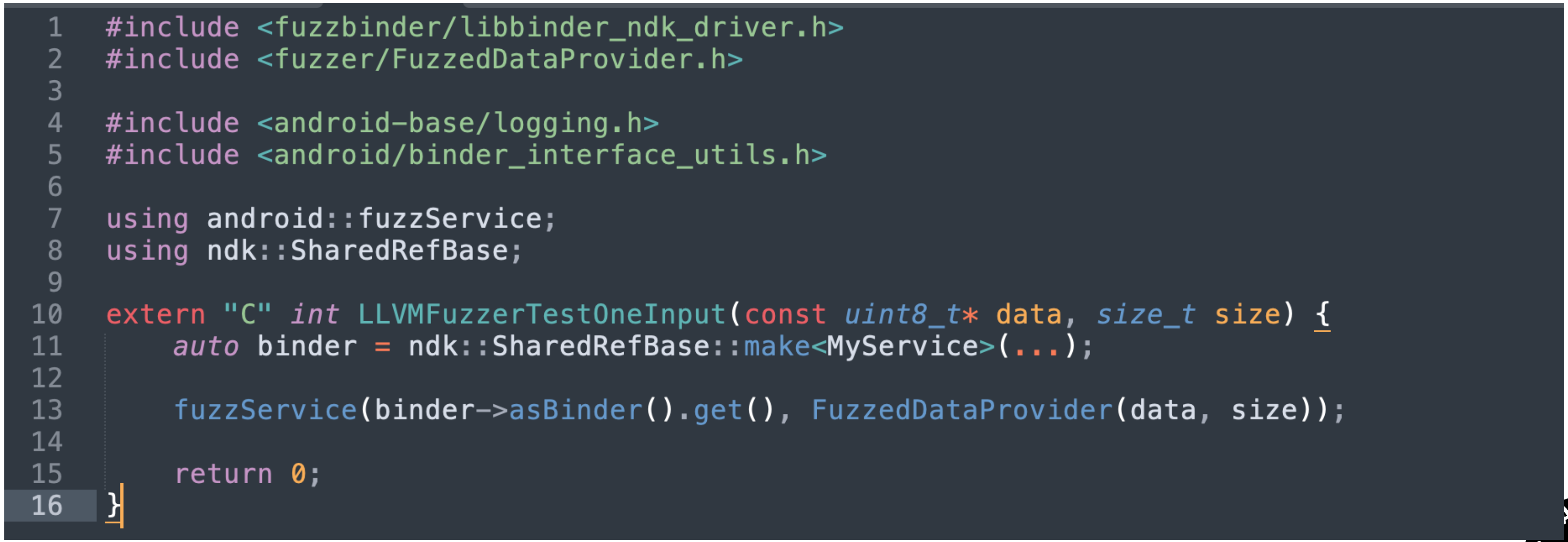
What to Fuzz.?



```
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: []
```

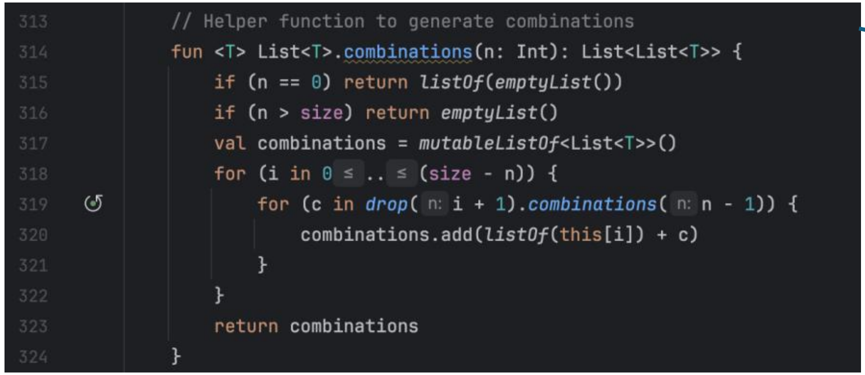
Example Fuzzing Code

- **Fuzzing Code Sample:**

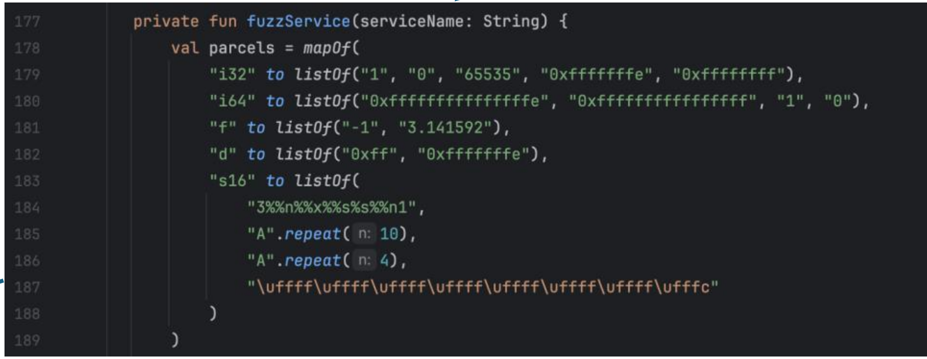


```cpp
#include <fuzzbinder/libbinder_ndk_driver.h>
#include <fuzzer/FuzzedDataProvider.h>

#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;
}
```

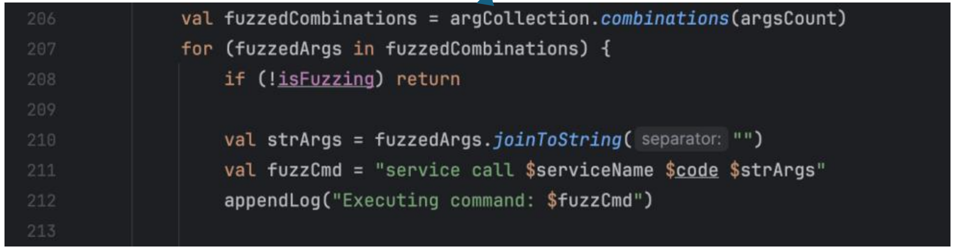
Our Fuzzing Code



```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
}
```



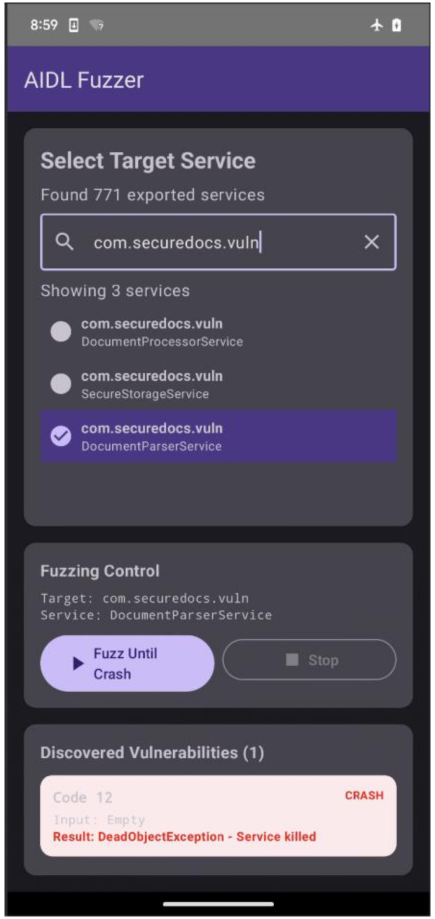
```kotlin
private fun fuzzService(serviceName: String) {
    val parcels = mapOf(
        "i32" to listOf("1", "0", "65535", "0xfffffffe", "0xffffffff"),
        "i64" to listOf("0xfffffffffffffffe", "0xffffffffffffffff", "1", "0"),
        "f" to listOf("-1", "3.141592"),
        "d" to listOf("0xff", "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.



8:59

AIDL Fuzzer

Select Target Service

Found 771 exported services

com.securedocs.vuln

Showing 3 services

com.securedocs.vuln
DocumentProcessorService

com.securedocs.vuln
SecureStorageService

com.securedocs.vuln
DocumentParserService

Fuzzing Control

```
Target: com.securedocs.vuln
Service: DocumentParserService
```

Fuzz Until Crash

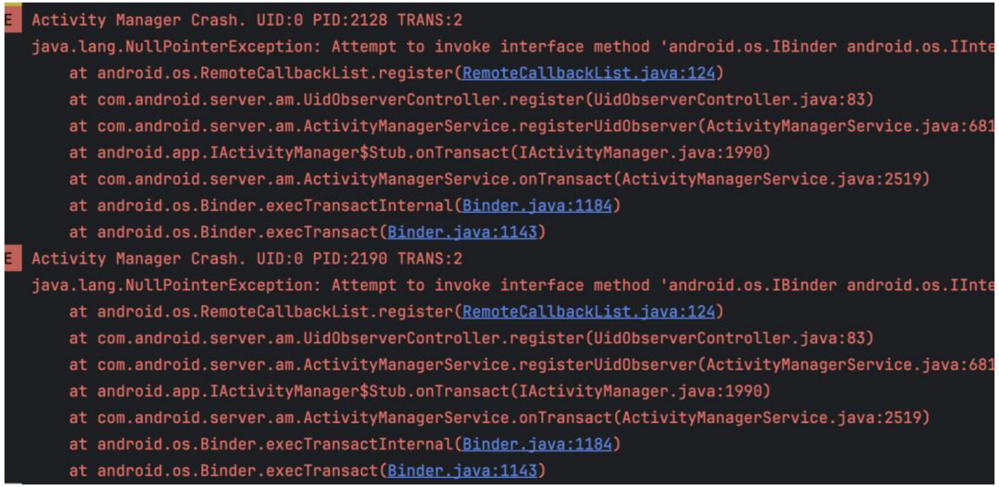
Stop

Discovered Vulnerabilities (1)

Code 12 CRASH

Input: Empty

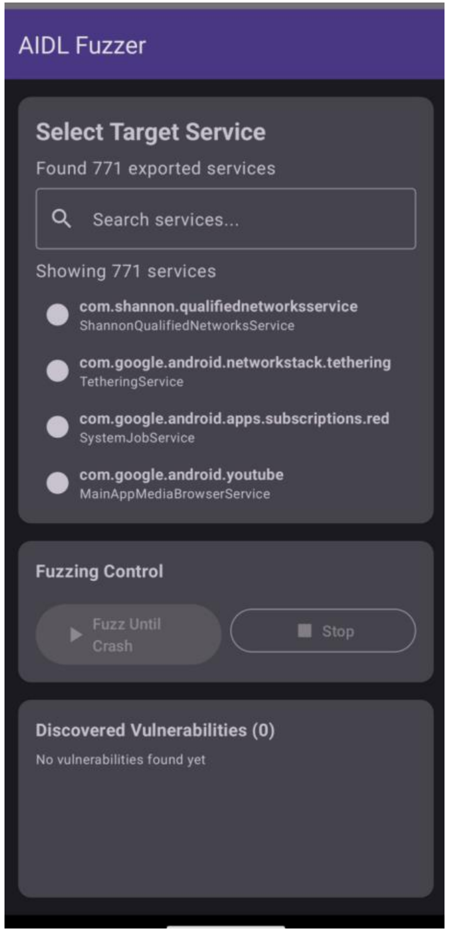
Result: DeadObjectException - Service killed



```
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)
```

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.



AIDL Fuzzer

Select Target Service

Found 771 exported services

Search services...

Showing 771 services

- com.shannon.qualifiednetworksservice
  ShannonQualifiedNetworksService
- com.google.android.networkstack.tethering
  TetheringService
- com.google.android.apps.subscriptions.red
  SystemJobService
- com.google.android.youtube
  MainAppMediaBrowserService

Fuzzing Control

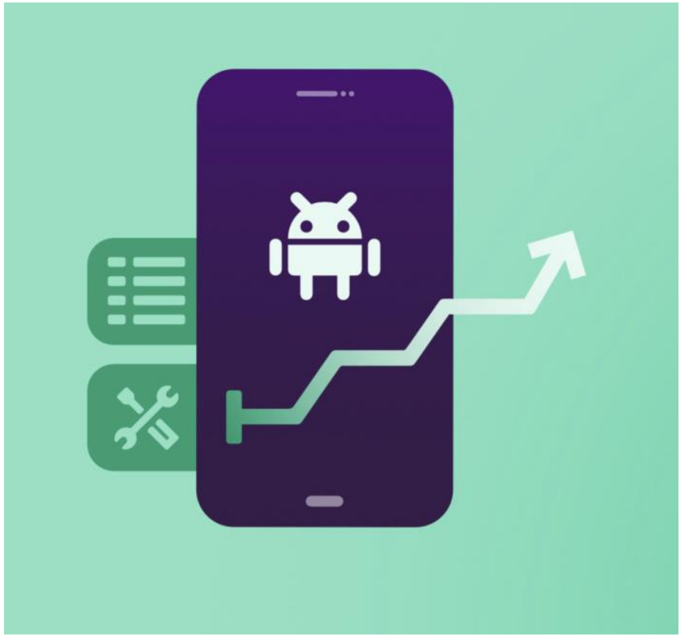
Discovered Vulnerabilities (0)

No vulnerabilities found yet

Challenges in AIDL Fuzzing

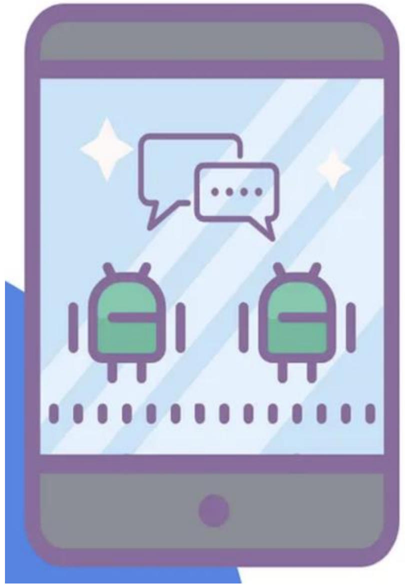
- **Hurdles**
  - Handling complex data structures in AIDL interfaces.
  - Dealing with permissions restrictions and sandboxing.

- **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

WHERE?
WHEN? WHY?
WHEN?
WHY?
WHO?
WHO? WHERE? WHAT? HOW?
WHAT? HOW? WHY?
WHEN?
WHO?
WHAT?
WHERE?
HOW? HOW?
WHEN?
WHEN?
WHERE?
WHERE? WHICH? WHOSE? WHEN? WHY?
WHOSE?
WHEN? WHERE?
HOW?
HOW?
WHICH?
WHOSE?
WHAT? HOW? WHY?
WHO? WHERE? WHAT? HOW?
WHY? HOW?
WHAT? HOW? WHY?
WHO? WHERE? WHAT? HOW?
WHERE? WHICH? WHOSE? WHEN? WHY?
WHAT?
WHY? HOW? WHERE?
WHO? WHOSE?
WHERE? WHAT? HOW?
WHAT?
WHY?
WHO?
HOW?
WHERE?
WHERE? WHAT? HOW?
WHERE? WHICH? WHOSE? WHEN? WHY?
HOW? WHERE? WHY?
WHERE?
HOW?
HOW? WHO?
WHO? WHERE? WHAT? HOW?
WHAT? HOW? WHY?
WHERE?
WHAT?
HOW?
HOW?
WHAT?
WHERE?

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

Thank you!