FReD: Identifying File Re-Delegation in Android System Services

Sigmund Albert Gorski III, Seaver Thorn,
William Enck, Haining Chen*
NC State, *Google

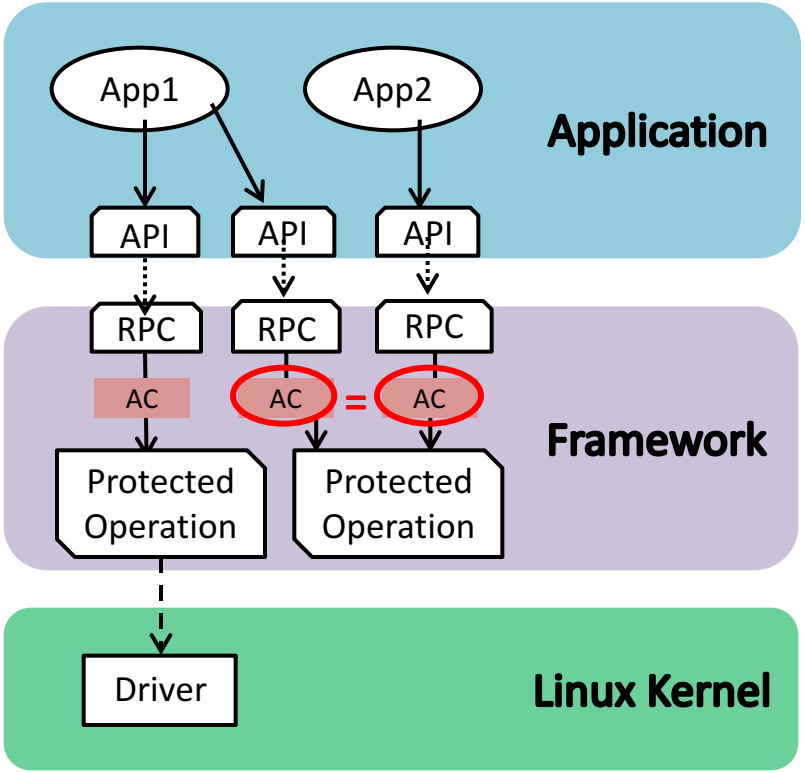
USENIX Security '22

Mobile OSes: Apps are Security Principals

- Benefits
  - Changes ambient authority
  - Better privilege separation
- Challenges
  - OS now responsible for protecting high-level information and resources
  - Complex access control policy needs to be enforced throughout the system

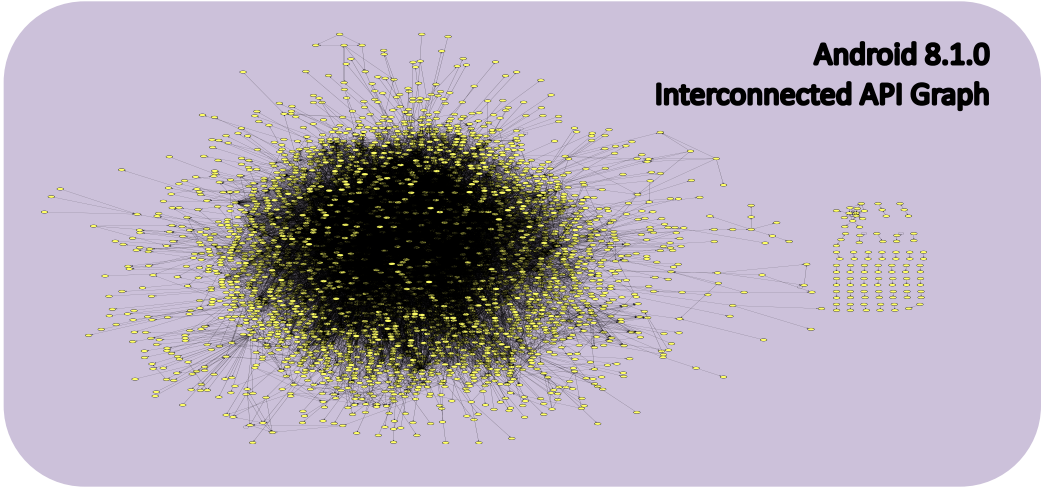
Android Framework

Mapping App APIs to Permissions:
- Stowaway
- PScout
- Arcade

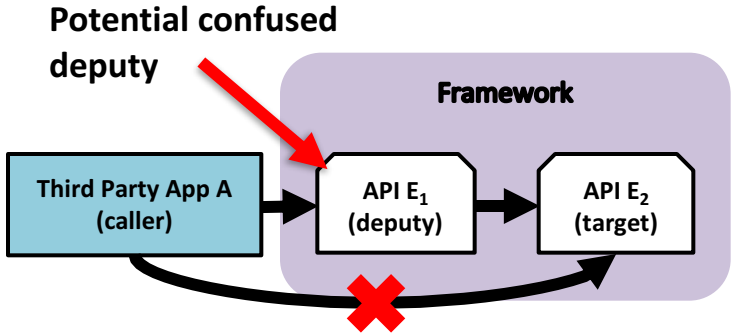


Evaluating correctness (consistency) of access control checks:
- Kratos
- ACEDroid
- **ACMiner**

The Framework is *Very* Interconnected



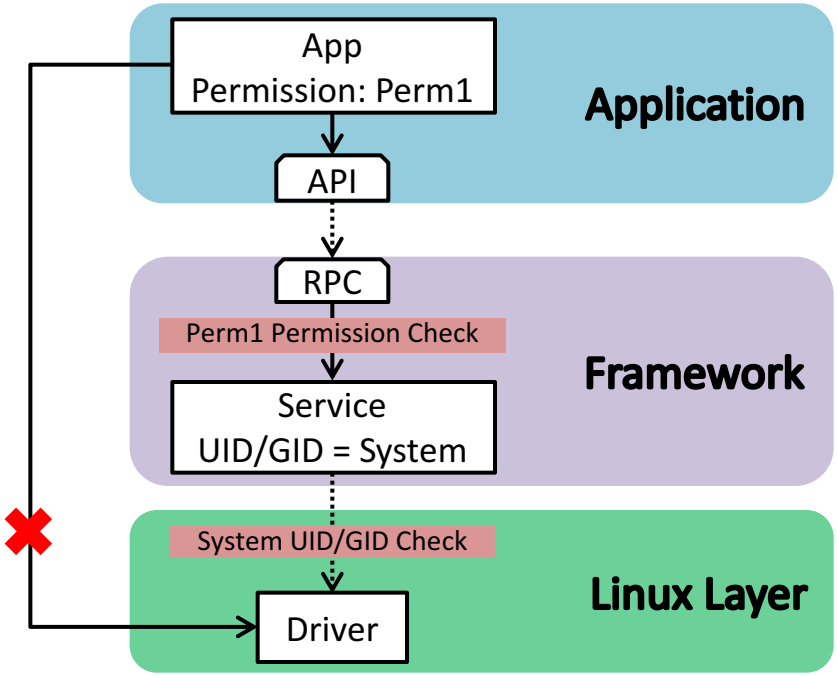
Evaluating re-delegation of permissions between APIs:
- **ARF**



The problem is caused by the **changes in ambient authority** during execution.

Is access to files also being re-delegated?

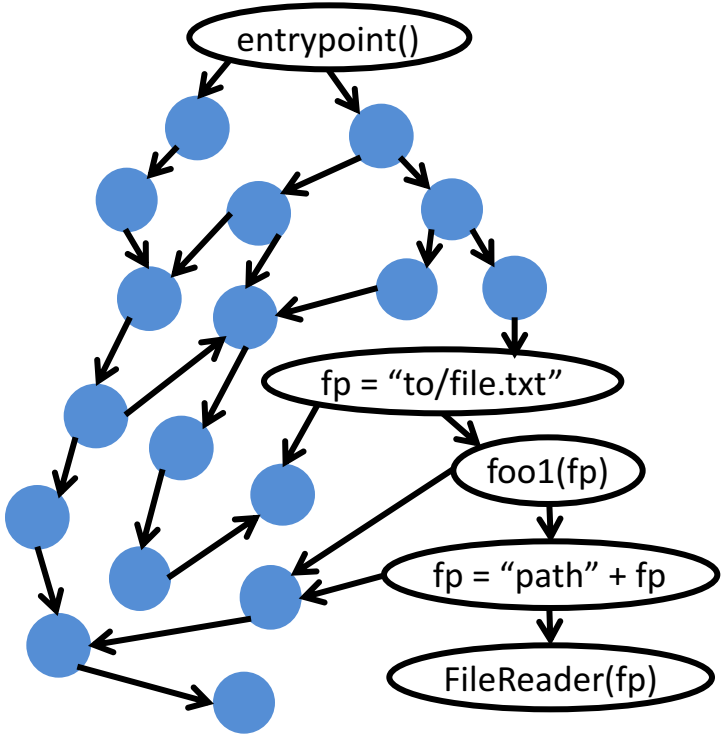
File Access Re-Delegation



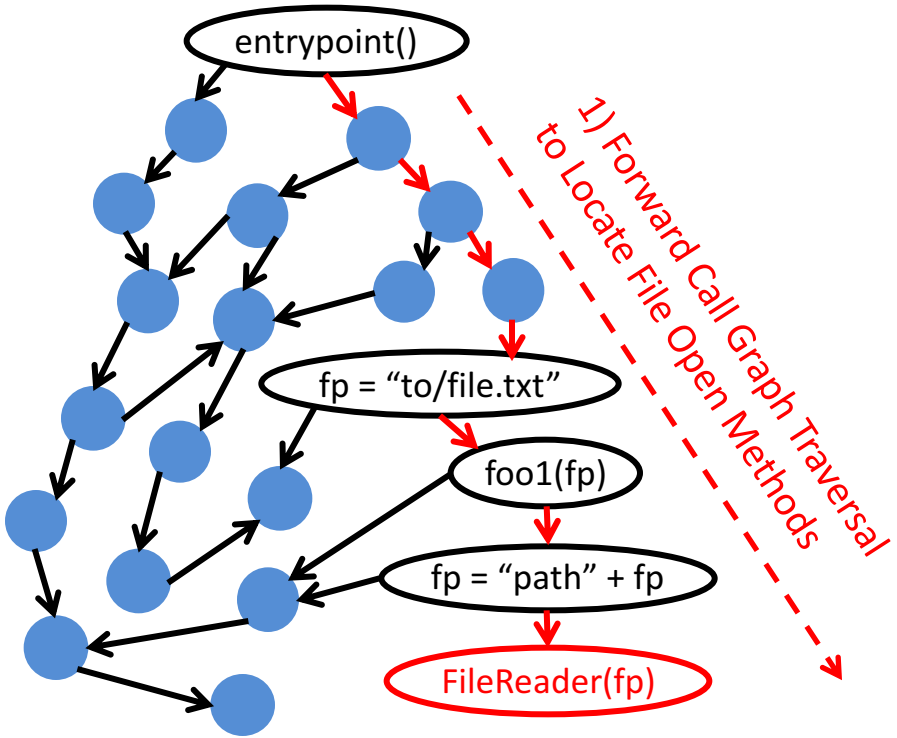
- All traditional OSes require file re-delegation to provide differentiated access.
- We want to find **improper** re-delegation
- **Hypothesis:** If you can find all file re-delegation, most proper re-delegation to sensitive files can be programmatically filtered

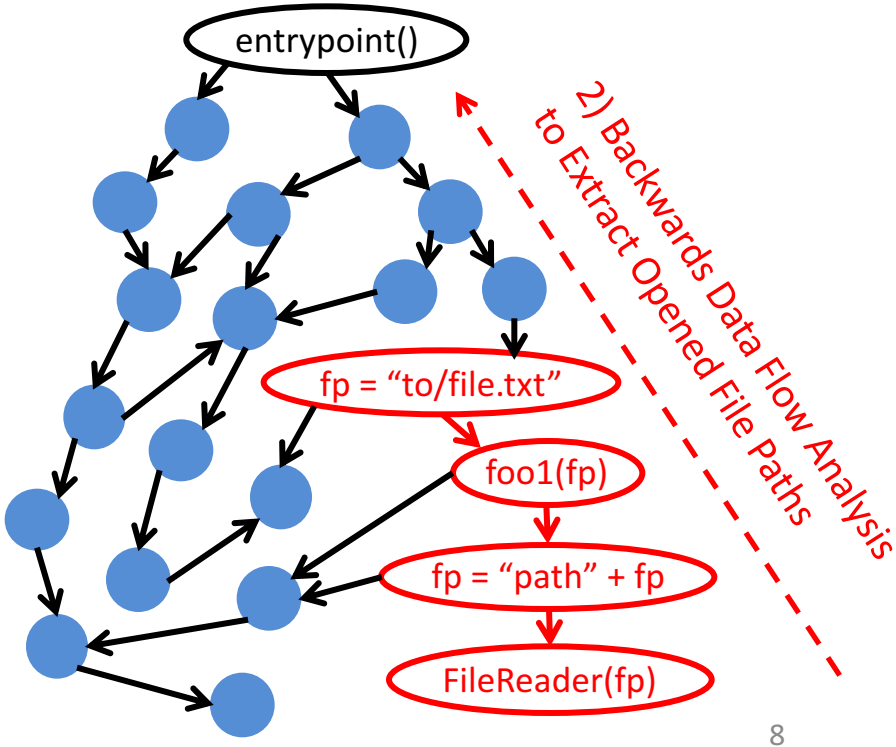
File Re-Delegation Finder (FReD)

- **Question**: How can we find improper re-delegation of files?
  - Which service APIs open files?
  - What file paths are opened?
  - Which opened file paths are sensitive?
  - Which opened sensitive file paths are re-delegated to callers?
- **FReD:** static program analysis of
  - Android's *Java* system services **and**
  - associated *native code* to identify these files.
  - Builds on ACMiner and ARF



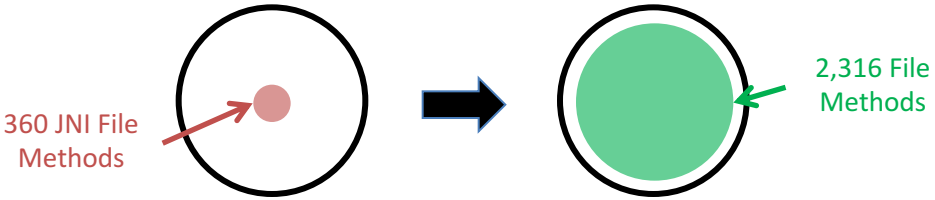
Conceptual Approach



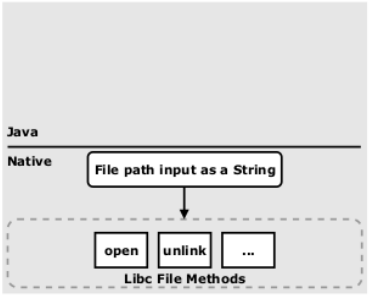


Identifying File Methods

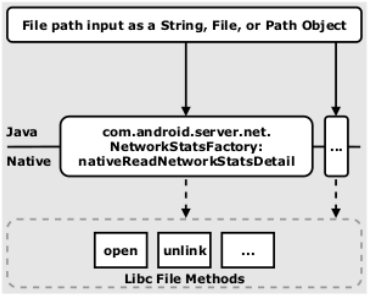
- **Problem:** backwards dataflow analysis from generic file methods lack sufficient context
- **Solution:** control flow analysis to the highest level wrapper file method
  - Start with JNI file methods set F
  - Identify "public" Android APIs with control flow to method in F
  - Manually confirm and add to F
  - Repeat until fixed point



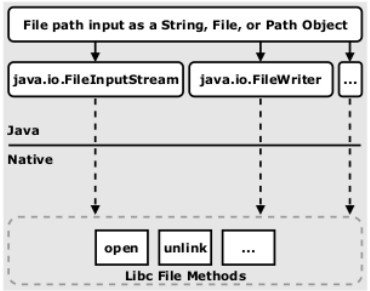
File method identification for AOSP 10.0.0



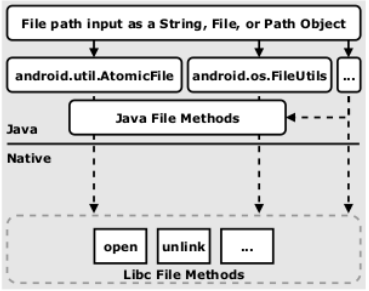
(a) JNI methods using file paths defined in native code



(b) JNI methods using file paths defined in Java code



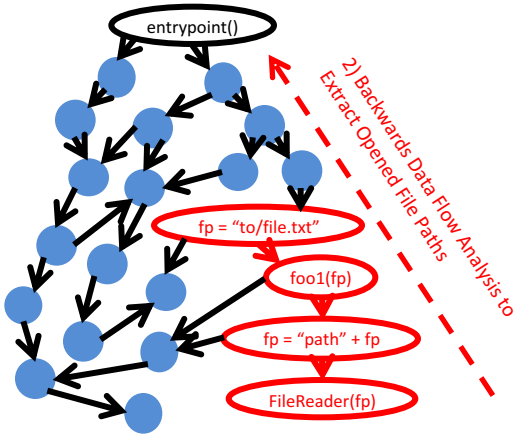
(c) Java API File Methods



(d) Android API File Methods

Extracting File Paths (Java)

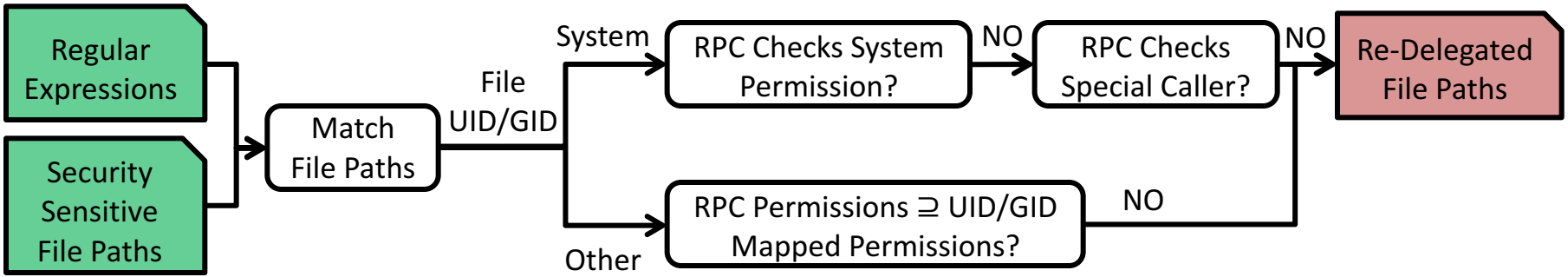
- Backwards data flow analysis keeps track of possible values for variables
- Creates an **intermediate representation** based on the types of operations that occur
- Ultimately generates regular expressions for each file path used in open, etc.
- Used angr to find file paths in native code called via JNI (see paper)



Node Name	Type	Description
Constant	Leaf	A Java primitive or string constant value.
Any	Leaf	Represents a value that could not be determined.
Unknown	Leaf	Represents a unexpected outcome in the analysis.
PlaceHolder	Leaf	A placeholder for a value that is currently being computed.
Append	Branch	Concatenates the values of its children.
Or	Branch	Represents the possibility that any one of its children is a valid value.
Loop	Branch	Indicates the existence of a loop.
Parent	Branch	The value is the parent path of its child's value.
Name	Branch	The value is the file name of its child's value.
EnvVar	Branch	Represents a value retrieved from the environment where its child is the key.
SysVar	Branch	Represents a value retrieved from the system properties where its child is the key.

File Access Re-Delegation Detection

- What is a security sensitive file path?
  - Any file assigned the system UID/GID
  - Any file assigned a UID/GID from the XML configuration files in /system/etc/permissions/
- Requires a real device to extract (many files in /data)



Firmware Analysis

- **AOSP Pixel 3a**
  - Android 10 r1
- **Google Pixel 3a**
  - Android 10 build QQ3A.200805.001
- **Samsung S20**
  - Android 10 build QP1A.19071.020

	AOSP Pixel 3 (9.0.0)	AOSP Pixel 3a (10.0.0)	Google Pixel 3a (10.0.0)	Samsung S20 (10.0.0)
# API Methods	65,508	73,073	73,461	83,346
# JNI Methods	5,176	5,416	5,419	7,023
# JNI File Methods	368	360	360	360
# Java API File Methods	907	888	888	888
# Android API File Methods	962	1,068	1,087	1,155
# Total File Methods	2,237	2,316	2,335	2,403
# Total RPC Entry Points	5,337	6,287	6,304	12,169
# RPCs with File Methods	1,966	2,927	2,985	4,163
# File Methods in RPCs	661	602	717	766
# JNI Methods in RPCs	365	387	390	860
# Sec. Sensitive File Paths	139,463	157,074	56,434	56,559

Automated exclusion criteria reduced most RPC Entry points

Candidate RPC Entry Points Requiring Manual Inspection

# Intermediate Exprs.	23	42	113
# Regexes	7	10	36
# RPC Entry Points	23	31	60
# Files	51	44	143

Findings Overview

Deputy (RPC Service)	Impact	Firmware	
removeSharedAccountAsUser (AccountManagerService)	Moderate (CVE-2020-0208)	AOSP	Data Manipulation
renameSharedAccountAsUser (AccountManagerService)	Moderate (CVE-2020-0209)	AOSP	
invalidateAuthToken (AccountManagerService)	Minor	AOSP	
updateCredentials (AccountManagerService)	Minor	AOSP	
add (DropBoxManagerService)	Minor	AOSP	
getSharedAccountsAsUser (AccountManagerService)	Moderate (CVE-2020-0210)	AOSP	Data Leaks
getPreviousName (AccountManagerService)	Minor	AOSP	
isCredentialsUpdateSuggested (AccountManagerService)	Minor	AOSP	
getTotalBytes (StorageStatsService)	Minor	AOSP	
isStorageLow (PackageManagerService)	Minor	AOSP	
sspInit (BlockchainTZService)	Moderate (CVE-2021-25459)	Samsung	Denial of Service
sspExit (BlockchainTZService)	Moderate (CVE-2021-25460)	Samsung	

Example File Access Re-Delegation

- **Impact:** Allows any app to removed shared accounts they do not manage
- **Problem:** Missing check isAccountManagedByCaller
- RPC renameSharedAccountAsUser has a similar problem and impact
- CVE-2020-0208 and CVE-2020-0209

```
1  boolean removeSharedAccountAsUser(Account ac, int userId) {
2    int uid = getCallingUid();
3    userId = handleIncomingUser(uid);
4    UserAccounts acs = getUserAccounts(userId);
5    boolean deleted = acs.accountsDb.deleteSharedAccount(ac);
6    if (deleted)
7      removeAccountInternal(acs, ac, uid);
8    return deleted;
9  }
```

Should be calling isAccountManagedByCaller

Files: /data/system_ce/0/accounts_ce.db
and /data/system_de/0/accounts_de.db
Access: Only system can RWX

Limitations

- Requires files from a real device (files in /data cannot be extracted from a firmware image, e.g., via BigMAC)
  - Re-delegated files must be in this FS dump
- Only considers Java services and associated JNI
- Inherits ACMiner limitations

Thank you!

- Treating apps as security principals significantly improves security,
  - … but it also puts significant burden on the OS.
- FReD uses static analysis to identify when functionality related to files can be abused by third-party applications.
- **Source Code:**
  - https://github.com/wspr-ncsu/fred

William Enck
Director, WSPR Lab
co-Director, Secure Computing Institute
Professor, Computer Science
North Carolina State University
W: https://enck.org E: whenck@ncsu.edu

In collaboration with
Sigmund Albert Gorski III (NCSU),
Seaver Thorn (NCSU), and
Haining Chen (Google)

Support from: Google ASPIRE award and Army Research Office grant W911NF-16-1-0299