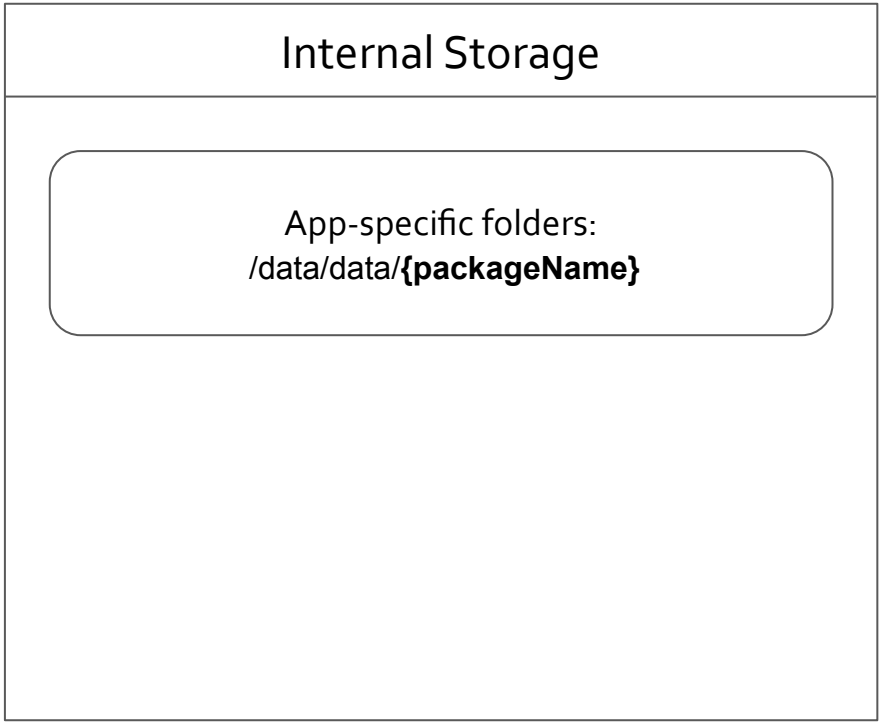
ScopeVerif: Analyzing the Security of Android's Scoped Storage via Differential Analysis

Zeyu Lei[*] , Güliz Seray Tuncay[+], Beatrice Carissa Williem[*],
Z. Berkay Celik[*] , Antonio Bianchi[*]

[*]Purdue University, [+]Google

Storage on Android



Storage on Android

Internal Storage

App-specific folders:
/data/data/**{packageName}**

External Storage

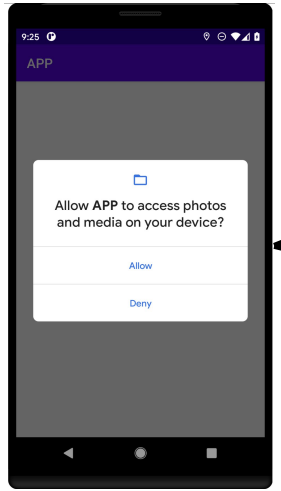
App-specific folders:
/storage/emulated/0/**Android/data/{packageName}**

Shared folders:
/storage/emulated/0/**Download**

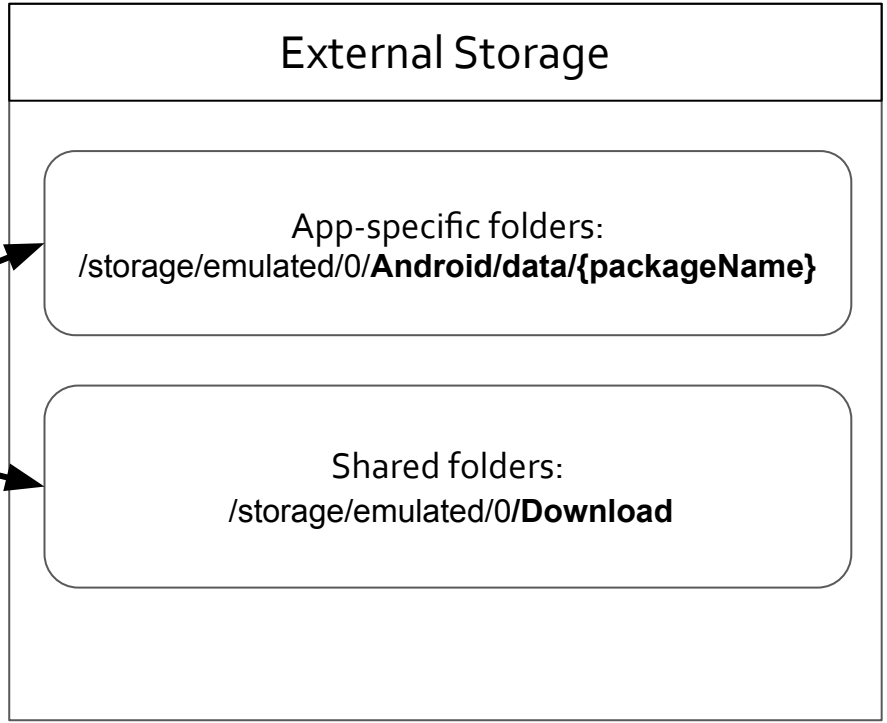
Storage on Android

Prior to Android 10

- Coarse-grained access control



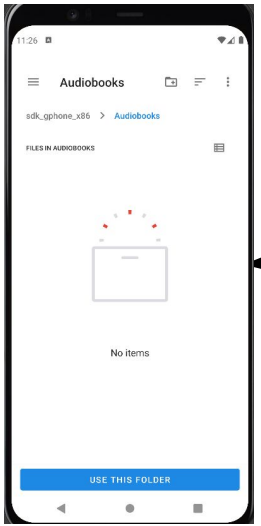
App with permissions



Storage on Android

Prior to Android 10

- Coarse-grained access control



User interaction

External Storage

App-specific folders:
/storage/emulated/0/**Android/data/{packageName}**

Shared folders:
/storage/emulated/0/**Download**

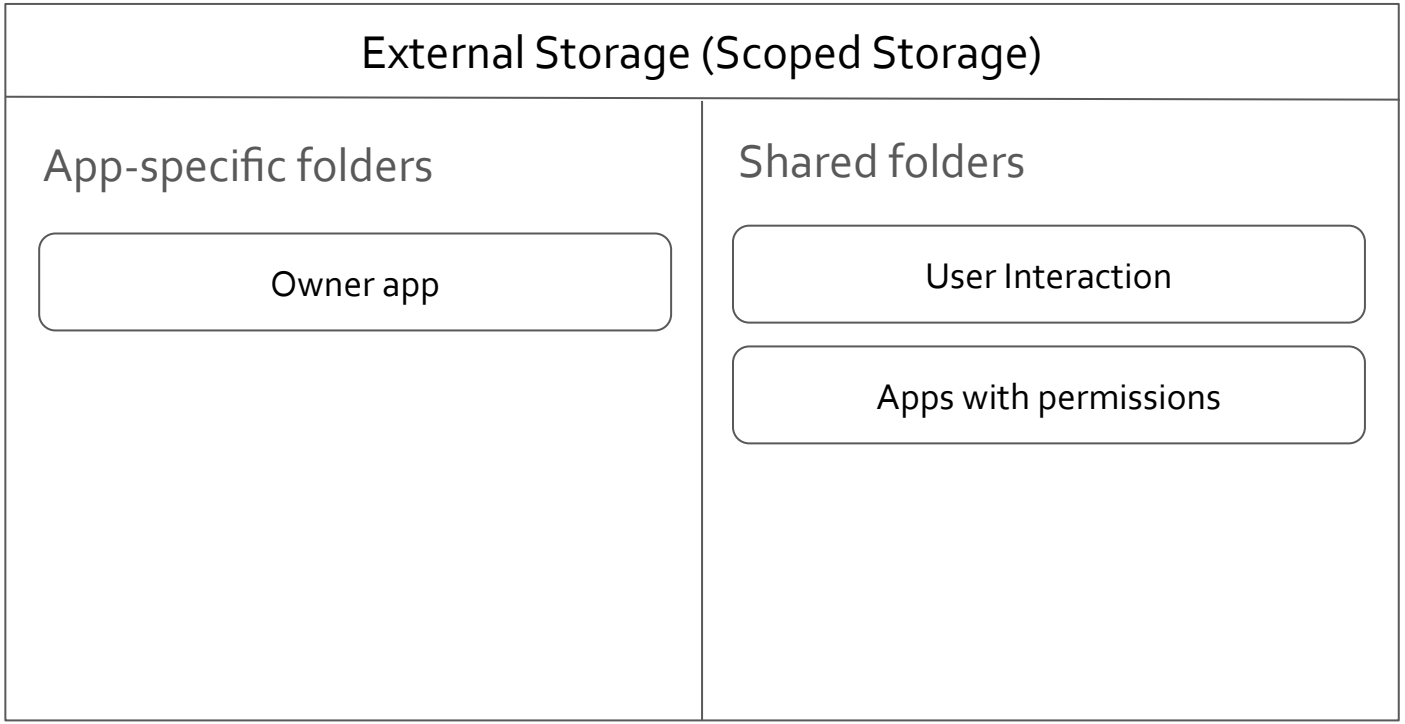
Storage Prior to Android 10

Storage Since Android 10

Since Android 10.

- Scoped storage limits broad acces
- Full access to app's own folder

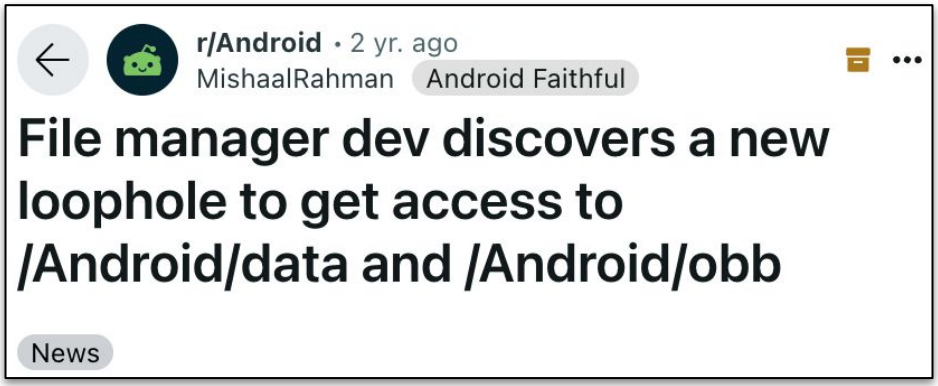
Background


External Storage (Scoped Storage)
App-specific folders
Owner app
Shared folders
User Interaction
Apps with permissions

Motivation

Android Storage

- Complex and varies among different implementations
- Creating unhandled corner cases resulting in security & privacy issues
  - e.g., SAF Loophole:
    - SAF only partially blocks access to app-specific folders.
    - Other apps' private files are still accessible



Motivation

- Unrestricted access to your own app internal and external storage
- Unrestricted access to contribute files to media and download collection
- "MediaStore.Images", "MediaStore.Video", "MediaStore.Audio" collections can be read with the storage permission
- Cannot access files in any other app's dedicated specific directory in external storage for Android 11 and above
- Reading or writing outside of collections (media collections and own app directories) requires user interaction / all file access
- Location permission need to be declared in manifest, and approved by metadata requires declaration in permission from the user.
- Editing and deleting media files made by other apps is not possible without user interaction
- Cannot access files in any other app's dedicated specific directory in external storage for Android 11 and above
- …

Motivation

?

Correctness? Consistency? Effectiveness?

Technical Challenges

Multiple APIs:

- Different "expected" behaviors
- Lots of exceptions:
  - Special permissions
  - Special files

API that allows background file operations

- File
- MediaStore
- ContentResolver
- …

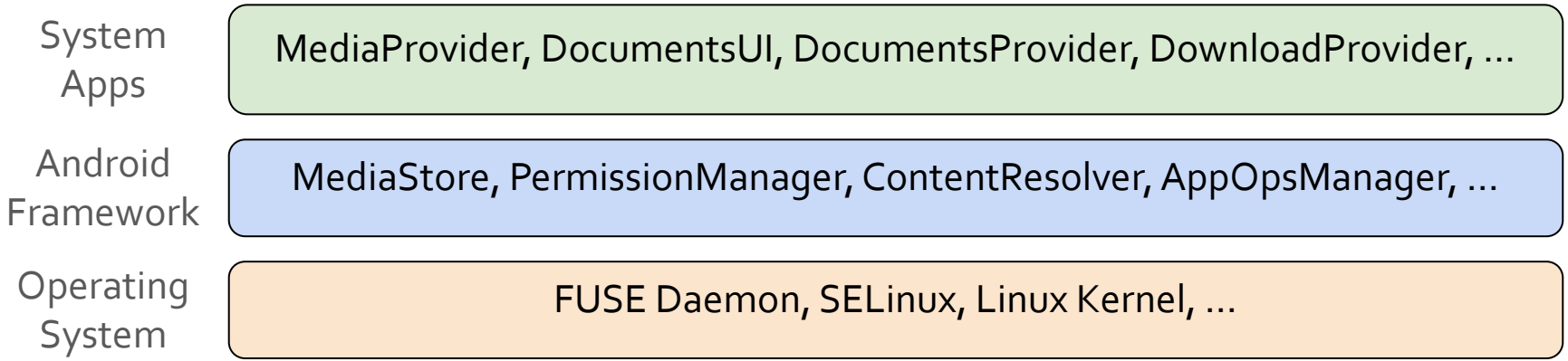
API that require user-Interaction

- SAF Picker
- DocumentFile
- DocumentsContract
- …

Technical Challenges

Large & Fragmented Codebase

- Ideally centralized and unified checks
- In reality, duplicate logics among various concurrent API implementation
  - different components, layers, programming languages

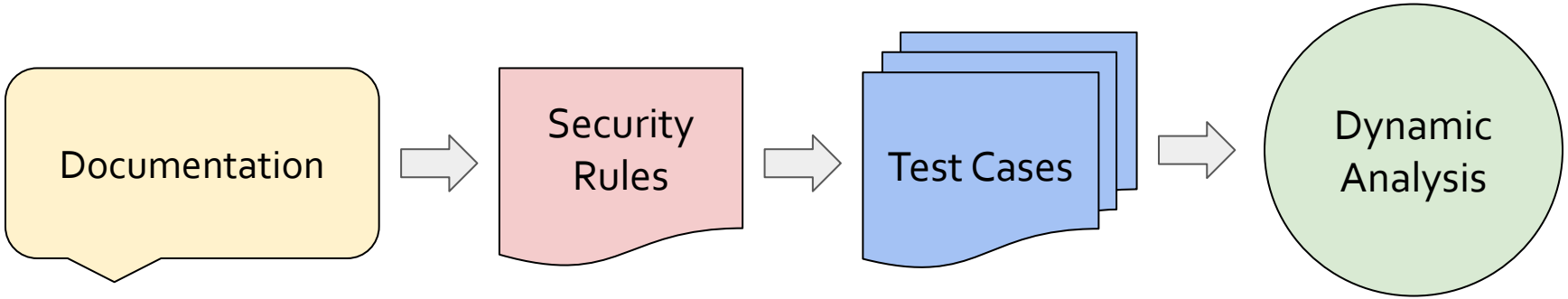


Technical Challenges

Cross-Version & OEM Inconsistencies

- Subtle, undocumented changes/patches across Android 12, 13, and 14
- OEMs like Samsung or Huawei often use customized versions of Android
- Verifying a single device or version is inadequate

Approach


Documentation
Security Rules
Test Cases
Dynamic Analysis

Extracting Security Rules

"On Android 11, apps can no longer access files in any other app's dedicated, app-specific directory within external storage."

Extracting Security Rules

"On Android 11, apps can no longer access files in any other app's dedicated, app-specific directory within external storage."

Type: Confidentiality
Actions: Read, Write, Move, Rename, …
Targets: Other apps' private files
Attributes: Content, Path, Size, …
APIs: File API, MediaStore API, …
Permissions:

- MANAGE_EXTERNAL_STORAGE
- ACCESS_MEDIA_LOCATION
- …

Generating Test Cases

Security Rule:

Type: Confidentiality
Actions: Read, Write, Move, Rename, …
Targets: Other apps' private files
Attributes: Content, Path, Size, …
APIs: File API, MediaStore API, …
Permissions:

- MANAGE_EXTERNAL_STORAGE
- ACCESS_MEDIA_LOCATION
- …

Multiple Test Cases:

type: Confidentiality
action: Read
api: File
path: "sdcard/Android/data/*.pdf"
permissions:

- ACCESS_MEDIA_LOCATION
- READ_EXTERNAL_STORAGE
- etc.

Generating Test Cases

Security Rule:

Type: Confidentiality
Actions: Read, ~~Write, Move, Rename, …~~
Targets: Other apps' private files
Attributes: Content, Path, Size, …
APIs: File API, MediaStore API, …
Permissions:

- MANAGE_EXTERNAL_STORAGE
- ACCESS_MEDIA_LOCATION
- …

Multiple Test Cases:

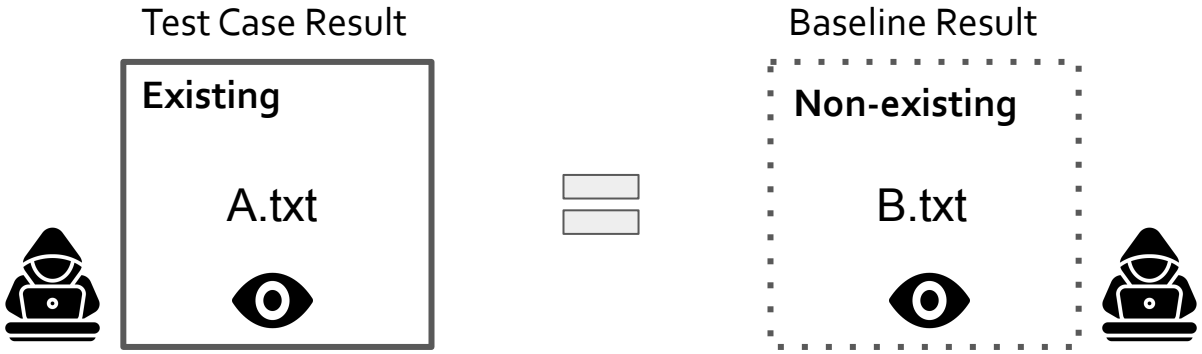
type: Confidentiality
action: Read
api: File
path: "sdcard/Android/data/*.pdf"
permissions:

- ACCESS_MEDIA_LOCATION
- READ_EXTERNAL_STORAGE
- etc.

Dynamic Analysis

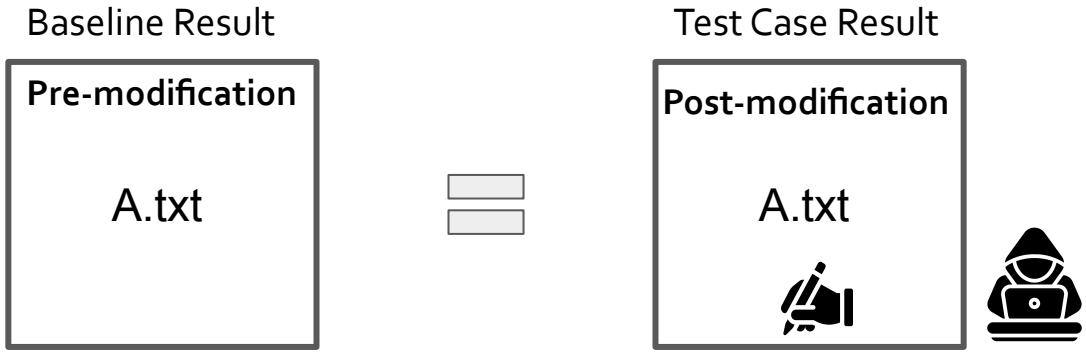
- For each test case, run test case twice: Baseline and Test
  - Baselines are constructed based on the type of security rule
- We collect and compare

Dynamic Analysis



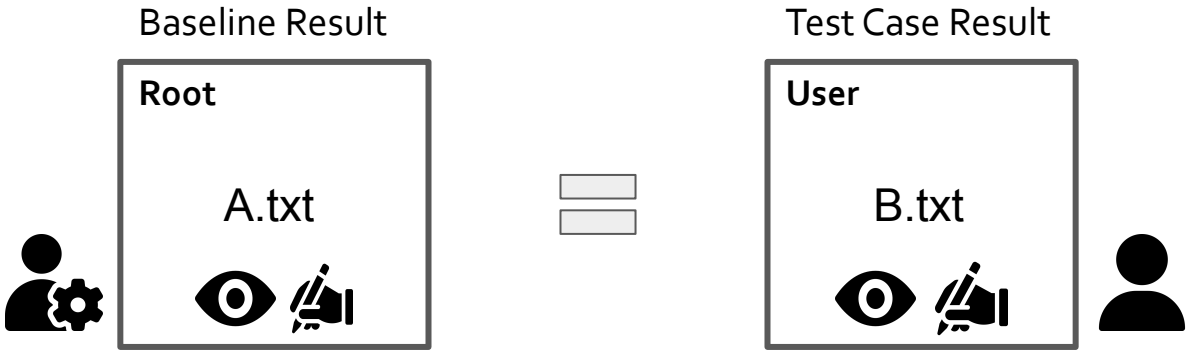
Confidentiality: compare feedback between accessing existing and non-existing path.

Dynamic Analysis



Integrity: compare file before and after modify attempts.

Dynamic Analysis



Availability: compare the operation feedback between root and the user.

Results Summary

ScopeVerif found **10** issues, **9** of which were **previously unknown**

ScopeVerif revealed **inconsistent implementations** across Android versions and OEMs.

ScopeVerif can automatically identify previously unknown security issues **within a day**.

Case Study: Metadata Leak

- File API:
  - "No such file or directory" (file doesn't exist) vs "Permission denied." (if file exists)
  - **No permission needed.**
- Privacy violations
  - Cross-app user identification
  - Covert channel between apps
    - File Path Existence = 1 Bit, multiple Files = multiple Bits
  - Collaborating apps can track users across different applications

Case Study: SAF Loophole (Huawei)

- On Huawei's Android 14 builds, SAF picker does not block:
  - Create or overwrite files in another app's private directory (e.g., /Android/data/another.app/)
  - Squatting attack
    - Mislead the victim app into using the attacker-created file instead of their original files.
- Google's Patch in Android 13 & 14
  - **Huawei did not fully adopt these fixes in their Android customization.**

Summary

- Systematic, automated analysis of Android's Permission Model regarding file access ⟶ ScopeVerif
- ScopeVerif utilizes differential analysis
  - Verify the correctness of implementation
  - Identify inconsistencies between Android versions and devices
- ScopeVerif found previously unknown issues
  - Including security and privacy issues
  - We reported our findings to Google and Huawei
  - Both companies offered us bug bounties

Zeyu Lei[*] , Güliz Seray Tuncay[+], Beatrice Carissa Williem[*], Z. Berkay Celik[*] , Antonio Bianchi[*] ([*]Purdue University, [+]Google)

Thank you! Questions?

lei76@purdue.edu

https://github.com/purseclab/ScopeVerif

PURDUE UNIVERSITY

CERIAS

PurSec Lab

Google