To Write or not to Write, that is the Question - Abusing of Leaked Data and Vulnerable User Assets on Android

Julien Thomas

protektoid.com - @julien_thomas

March 28th, 2019 - Budapest

About this talk

- About the author
  - (Dr.) Julien Thomas

About this talk

- About the author
  - (Dr.) Julien Thomas
  - Protektoid project, lots of bug hunting
  - Protektoid project, hopefully still lot of work and talks ☺

About this talk

- About the author
  - (Dr.) Julien Thomas
  - Protektoid project, lots of bug hunting
  - Protektoid project, hopefully still lot of work and talks ☺
- Objectives of this talk
  - review data management on Android (and not assets)

About this talk

- About the author
  - (Dr.) Julien Thomas
  - Protektoid project, lots of bug hunting
  - Protektoid project, hopefully still lot of work and talks ☺
- Objectives of this talk
  - review data management on Android (and not assets)
  - review lots of bad practices on data managements on Android
  - introduce (approved) fixes

About this talk

- About the author
  - (Dr.) Julien Thomas
  - Protektoid project, lots of bug hunting
  - Protektoid project, hopefully still lot of work and talks ☺
- Objectives of this talk
  - review data management on Android (and not assets)
  - review lots of bad practices on data managements on Android
  - introduce (approved) fixes
  - talk about bug hunting on Android

About this talk

- About the author
  - (Dr.) Julien Thomas
  - Protektoid project, lots of bug hunting
  - Protektoid project, hopefully still lot of work and talks ☺
- Objectives of this talk
  - review data management on Android (and not assets)
  - review lots of bad practices on data managements on Android
  - introduce (approved) fixes
  - talk about bug hunting on Android
- Data exposure, again?
  - focus on app based (app data) exposure
  - minimal concern regarding insecure shared data loading

Guess it

- Your environment
  - an app with network permission only
    - without any rootkit

Guess it

- Your environment
  - an app with network permission only
    - without any rootkit
  - your device is **not** rooted

Guess it

- Your environment
  - an app with network permission only
    - without any rootkit
  - your device is **not** rooted
  - your device is loaded with many apps

Guess it

- Your environment
  - an app with network permission only
    - without any rootkit
  - your device is **not** rooted
  - your device is loaded with many apps
    - at least a video streaming app*
    - at least a remote control app*
    - at least a remote storage app*

Guess it

- Your environment
  - an app with network permission only
    - without any rootkit
  - your device is **not** rooted
  - your device is loaded with many apps
    - at least a video streaming app*
    - at least a remote control app*
    - at least a remote storage app*
- Question: what can be done in general / worst cases?
  - file listing (other topic)?
  - user identity disclosure?
  - user credential disclosure?
  - (encrypted) remote file disclosure?
  - remote control?

Guess it

- Your environment
  - an app with network permission only
    - without any rootkit
  - your device is **not** rooted
  - your device is loaded with many apps
    - at least a video streaming app*
    - at least a remote control app*
    - at least a remote storage app*
- Question: what can be done in general / worst cases?
  - file listing (other topic)?
  - user identity disclosure?
  - user credential disclosure?
  - (encrypted) remote file disclosure?
  - remote control?

Outline

Outline

Documentation: case 1 - R**M

- *getExternalFilesDirs* (vs *getFilesDir*)[1]: absolute paths to application-specific directories
  - internal to the app, **not typically visible to the user as media**.
  - shared storage may not always be available
  - no security is enforced with these files. Any application holding *WRITE_EXTERNAL_STORAGE* can write to these files.

[1] https://developer.android.com/reference/android/content/Context.html, https://developer.android.com/reference/android/os/Environment.html

Documentation: case 1 - R**M

- *getExternalFilesDirs* (vs *getFilesDir*)[1]: absolute paths to application-specific directories
  - internal to the app, **not typically visible to the user as media**.
  - shared storage may not always be available
  - no security is enforced with these files. Any application holding *WRITE_EXTERNAL_STORAGE* can write to these files.
- getExternalStorageDirectory
  - storage can hold a relatively large amount of data, shared across all apps (does not enforce permissions)
  - each user has their **own isolated shared storage**
    - apps **only have access to the shared storage for the user they're running as**.

[1] `https://developer.android.com/reference/android/content/Context.html`, `https://developer.android.com/reference/android/os/Environment.html`

Documentation: case 2

- Internal storage[2]:
  - **files saved here are accessible by only your app**.
  - when the user uninstalls your app, the system removes all your app's files from internal storage.

[2] https://developer.android.com/training/data-storage/files

Documentation: case 2

- Internal storage[2]:
  - **files saved here are accessible by only your app**.
  - when the user uninstalls your app, the system removes all your app's files from internal storage.
- External storage:
  - it's world-readable, so files saved here may be read outside of your control.
  - when the user uninstalls your app, **the system removes your app's files from here only if you save them in the directory from** *getExternalFilesDir*().
  - external storage is the best place for files that don't require access restrictions and for **files that you want to share with other apps** or allow the user to access with a computer.

[2] `https://developer.android.com/training/data-storage/files`

Internal storage

- Objectives: seems to be a match ♥
  - store app core data
  - store app private data
  - store low volume data

Internal storage

- Objectives: seems to be a match ♥
  - store app core data
  - store app private data
  - store low volume data
- Core storage on */data*
  - cache, databases, files

```
drwxrwx--x 60 system system /data
drwxrwx--x 60 u0_a165 u0_a165 /data/data/my_app
```

Internal storage

- Objectives: seems to be a match ♥
  - store app core data
  - store app private data
  - store low volume data
- Core storage on */data*
  - cache, databases, files

```
drwxrwx--x 60 system system /data
drwxrwx--x 60 u0_a165 u0_a165 /data/data/my_app
```

- Standard calls to Android framework
  - *getFilesDir*(), *getDatabasePath*()
  - *PackageInfo.applicationInfo.dataDir*

Non-Internal storage

- Objectives: not so much of a match💘
  - store data
  - store shared data
  - store non core (aka non Android produced) data
  - store shared data readable by other apps

Non-Internal storage

- Objectives: not so much of a match💥
  - store data
  - store shared data
  - store non core (aka non Android produced) data
  - store shared data readable by other apps
- Store on */sdcard*
  - /storage/self/primary ↦/mnt/user/0/primary ↦ /storage/emulated/0

```
drwxrwx--x 48 root sdcard_rw /sdcard
drwxrwx--x 48 root sdcard_rw /sdcard/Android/data
drwxrwx--x 48 root sdcard_rw /sdcard/Android/data/my_app
/dev/fuse on /storage/emulated type fuse (rw,nosuid,nodev,noexec,noatime,
      user_id=1023,group_id=1023,default_permissions,allow_other)
```

Non-Internal storage

- Objectives: not so much of a match💥
  - store data
  - store shared data
  - store non core (aka non Android produced) data
  - store shared data readable by other apps
- Store on */sdcard*
  - /storage/self/primary ↦/mnt/user/0/primary ↦ /storage/emulated/0

```
drwxrwx--x 48 root sdcard_rw /sdcard
drwxrwx--x 48 root sdcard_rw /sdcard/Android/data
drwxrwx--x 48 root sdcard_rw /sdcard/Android/data/my_app
/dev/fuse on /storage/emulated type fuse (rw,nosuid,nodev,noexec,noatime,
      user_id=1023,group_id=1023,default_permissions,allow_other)
```

- However
  - sdcard is a misleading name
  - sdcard/android/data is a misleading path
  - documentation triggers wrong perspectives

SD card “support” and features

- Before talking about permissions and security policies

SD card “support” and features

- Before talking about permissions and security policies
- Recurring “external storage features”
  - storage is persistent
  - as per documentation, *getExternalStorageDirectory* makes, per default, file not being removed on uninstall

SD card “support” and features

- Before talking about permissions and security policies
- Recurring “external storage features”
  - storage is persistent
  - as per documentation, *getExternalStorageDirectory* makes, per default, file not being removed on uninstall
- “SDcards are not supported”
  - so per default we are fine unless a SD Card is used

SD card “support” and features

- Before talking about permissions and security policies
- Recurring “external storage features”
  - storage is persistent
  - as per documentation, *getExternalStorageDirectory* makes, per default, file not being removed on uninstall
- “SDcards are not supported”
  - so per default we are fine unless a SD Card is used
- Yet, good luck if you try to go against these arguments in bug bounties

Outline

Core principles

- Private data
  - only the app can directly access /data/data/my_app resources
  - security is enforced with basic Linux permissions (o=x|o=)
  - security is enforced with FileUriExposedException (since N)

Core principles

- Private data
  - only the app can directly access /data/data/my_app resources
  - security is enforced with basic Linux permissions (o=x|o=)
  - security is enforced with FileUriExposedException (since N)
- “External” data as per Android data structure
  - permissions are required (before K)
  - apps can read/write to their own folders
- “Shared” data as per Android data structure
  - permissions are required (since K and before Q)
  - apps can read/write to other app folders

Core principles

- Private data
  - only the app can directly access /data/data/my_app resources
  - security is enforced with basic Linux permissions (o=x|o=)
  - security is enforced with FileUriExposedException (since N)
- “External” data as per Android data structure
  - permissions are required (before K)
  - apps can read/write to their own folders
- “Shared” data as per Android data structure
  - permissions are required (since K and before Q)
  - apps can read/write to other app folders
- “Shared” data as per storage structure
  - permissions are required (since K and before Q)
  - Apps can read/write to other folders

Shared storage: implementation

- Generate folders for each apps: “mount” on demand
  - MountService: buildExternalStorageAppDataDirs

[3] https://source.android.com/devices/storage

Shared storage: implementation

- Generate folders for each apps: “mount” on demand
  - MountService: buildExternalStorageAppDataDirs
- Folder/file accesses are controlled at filesystem (wrapper) level
  - FUSE (Filesystem in Userspace)[3]

[3] https://source.android.com/devices/storage

Shared storage: implementation

- Generate folders for each apps: “mount” on demand
  - MountService: buildExternalStorageAppDataDirs
- Folder/file accesses are controlled at filesystem (wrapper) level
  - FUSE (Filesystem in Userspace)[3]
  - SDCardFS (Android O) sdcardfs/sdcardfs.h,sdcardfs/inode.c

[3] https://source.android.com/devices/storage

Shared storage: implementation

- Generate folders for each apps: “mount” on demand
  - MountService: buildExternalStorageAppDataDirs
- Folder/file accesses are controlled at filesystem (wrapper) level
  - FUSE (Filesystem in Userspace)[3]
  - SDCardFS (Android O) sdcardfs/sdcardfs.h,sdcardfs/inode.c
- Until Q, everything is owned by *sdcard_rw*
  - Android Q: more-fine grained?

[3] `https://source.android.com/devices/storage`

Content provider

- Private data setup shall never be altered

Content provider

- Private data setup shall never be altered
- Private data can be made readable/editable
  - with ContentProvider
  - will be the case for any owned data starting from Q

Content provider

- Private data setup shall never be altered
- Private data can be made readable/editable
  - with ContentProvider
  - will be the case for any owned data starting from Q
- Question: who knows about Content provider ?
  - Concept
  - Vulnerabilties
  - Analysis tools

Content provider (2)

- A feature to expose data out of the box (exposes files/shared)

```
<provider android:name="android.support.v4.content.FileProvider"
  android:authorities="com.mydomain.fileprovider"
  android:etported="false" android:grantUriPermissions="true">
  <meta-data android:name="android.support.FILE_PROVIDER_PATHS"
    android:resource="@tml/fileprovider" />
</provider>
<?xml version="1.0" encoding="utf-8"?>
<paths tmlns:android="http://schemas.android.com/apk/res/android">
<cache-path name="cache" path="shared"/>
</paths>
```

Content provider (2)

- A feature to expose data out of the box (exposes files/shared)

```
<provider android:name="android.support.v4.content.FileProvider"
  android:authorities="com.mydomain.fileprovider"
  android:etported="false" android:grantUriPermissions="true">
  <meta-data android:name="android.support.FILE_PROVIDER_PATHS"
    android:resource="@tml/fileprovider" />
</provider>
<?xml version="1.0" encoding="utf-8"?>
<paths tmlns:android="http://schemas.android.com/apk/res/android">
<cache-path name="cache" path="shared"/>
</paths>
```

- A framework to expose data as desired

```
public ParcelFileDescriptor openFile (Uri p3, String p4);
public Cursor query(Uri uri, @Nullable String[] projection, @Nullable
  String selection, @Nullable String[] selectionArgs, @Nullable String sortOrder);
```

Content provider (2)

- A feature to expose data out of the box (exposes files/shared)

```
<provider android:name="android.support.v4.content.FileProvider"
  android:authorities="com.mydomain.fileprovider"
  android:etported="false" android:grantUriPermissions="true">
  <meta-data android:name="android.support.FILE_PROVIDER_PATHS"
    android:resource="@tml/fileprovider" />
</provider>
<?xml version="1.0" encoding="utf-8"?>
<paths tmlns:android="http://schemas.android.com/apk/res/android">
<cache-path name="cache" path="shared"/>
</paths>
```

- A framework to expose data as desired

```
public ParcelFileDescriptor openFile (Uri p3, String p4);
public Cursor query(Uri uri, @Nullable String[] projection, @Nullable
  String selection, @Nullable String[] selectionArgs, @Nullable String sortOrder);
```

- A powerful framework
  - If not fully understood, better stay away ... from custom implementation

Developer perspectives

- Documentation is complex (learning curve?) and not the most sexiest thing on the planet
- Private folder is a constraint
  - hard to debug
  - hard to share data (really, what about content provider framework then?)

Developer perspectives

- Documentation is complex (learning curve?) and not the most sexiest thing on the planet
- Private folder is a constraint
  - hard to debug
  - hard to share data (really, what about content provider framework then?)
- External folder is great
  - easy to debug (logs are accessible through USB)
  - lot of storage capability
  - persistence is possible (“hacking”?)
  - data sharing is easy (once again, is it really easier?)

Consequences

- Private folder is mostly (x%) used only when strongly suggested
  - getDatabasePath

Consequences

- Private folder is mostly (x%) used only when strongly suggested
  - getDatabasePath
- External “private” folder is extensively used

Consequences

- Private folder is mostly (x%) used only when strongly suggested
  - getDatabasePath
- External “private” folder is extensively used
- For developers that “care about security”, tricks are used

Consequences

- Private folder is mostly (x%) used only when strongly suggested
  - getDatabasePath
- External “private” folder is extensively used
- For developers that “care about security”, tricks are used
  - trick 1: put files in */sdcard* instead of */sdcard/Android/data/my_app*

Consequences

- Private folder is mostly (x%) used only when strongly suggested
  - getDatabasePath
- External “private” folder is extensively used
- For developers that “care about security”, tricks are used
  - trick 1: put files in */sdcard* instead of */sdcard/Android/data/my_app*
  - trick 2: make them “invisible” with the “.” prefix

Consequences

- Private folder is mostly (x%) used only when strongly suggested
  - getDatabasePath
- External “private” folder is extensively used
- For developers that “care about security”, tricks are used
  - trick 1: put files in */sdcard* instead of */sdcard/Android/data/my_app*
  - trick 2: make them “invisible” with the “.” prefix
  - (trick?) 3: not declaring them to media manager

Consequences

- Private folder is mostly (x%) used only when strongly suggested
  - getDatabasePath
- External “private” folder is extensively used
- For developers that “care about security”, tricks are used
  - trick 1: put files in */sdcard* instead of */sdcard/Android/data/my_app*
  - trick 2: make them “invisible” with the “.” prefix
  - (trick?) 3: not declaring them to media manager

Outline

Objectives and “false positives”

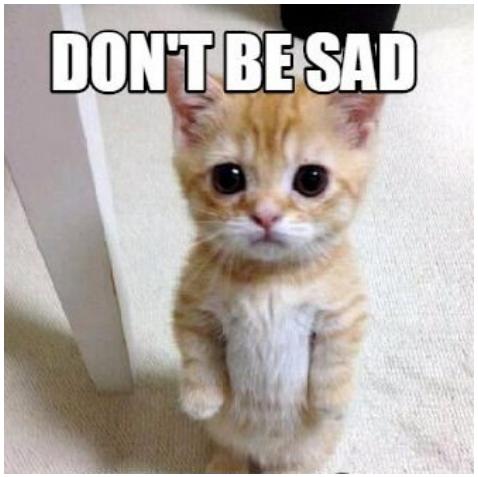
- Initial objectives: report any non explicitly shared data

Objectives and “false positives”

- Initial objectives: report any non explicitly shared data
  - did not end well
  - “non critical” (did you ask your users?), “no better option” (really?), “user can change it” (really, user fault?)

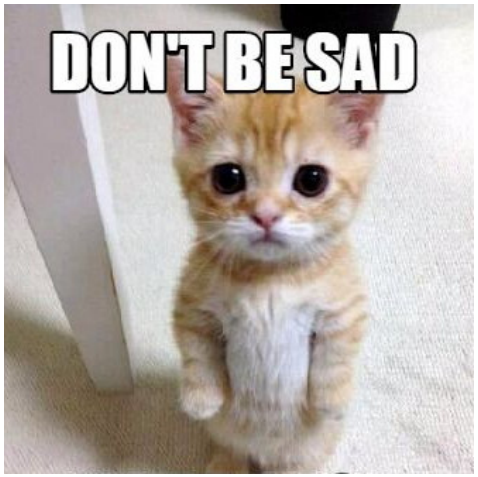
Objectives and “false positives”

- Initial objectives: report any non explicitly shared data
  - did not end well
  - “non critical” (did you ask your users?), “no better option” (really?), “user can change it” (really, user fault?)



Objectives and “false positives”

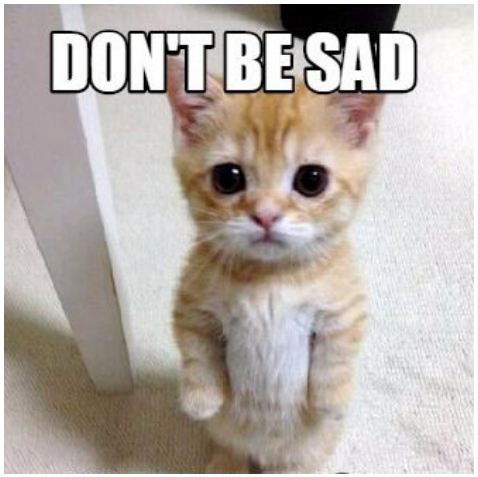
- Initial objectives: report any non explicitly shared data
  - did not end well
  - “non critical” (did you ask your users?), “no better option” (really?), “user can change it” (really, user fault?)



- Upgraded objectives: categorize leaks and focus on top ones
  - exposes user PII, app secrets, user passwords, user web activity, corporate (sensitive) data, user (sensitive) data

Objectives and “false positives”

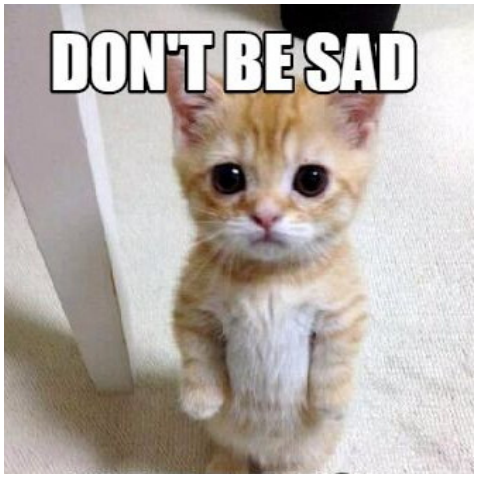
- Initial objectives: report any non explicitly shared data
  - did not end well
  - “non critical” (did you ask your users?), “no better option” (really?), “user can change it” (really, user fault?)



- Upgraded objectives: categorize leaks and focus on top ones
  - exposes user PII, app secrets, user passwords, user web activity, corporate (sensitive) data, user (sensitive) data
  - non-deniable leak (i.e. permission protected data)

Objectives and “false positives”

- Initial objectives: report any non explicitly shared data
  - did not end well
  - “non critical” (did you ask your users?), “no better option” (really?), “user can change it” (really, user fault?)



- Upgraded objectives: categorize leaks and focus on top ones
  - exposes user PII, app secrets, user passwords, user web activity, corporate (sensitive) data, user (sensitive) data
  - non-deniable leak (i.e. permission protected data)
- ended a little better, still in progress

Candidates

- (Recommended) apps with a reasonable user base
- Available on playstore

Candidates

- (Recommended) apps with a reasonable user base
- Available on playstore

Type	Example	Reason	Detect	
user PII	email	PII, phone owner		
	user profile	PII, phone owner		
user secrets	password			
	bank info			
app secrets	auth token	account takeover		
	credentials	account takeover		
	protocol IDs	device takeover		
user activity	browsing	user profiling		
	geo location	protected info		
corporate data	sensitive data	corporate secrets		
	credentials			

Candidates

- (Recommended) apps with a reasonable user base
- Available on playstore

Type	Example	Reason	Detect	Users
user PII	email	PII, phone owner	✓	
	user profile	PII, phone owner	✓	
user secrets	password		✓	
	bank info		✓	
app secrets	auth token	account takeover	✓	
	credentials	account takeover	✓	
	protocol IDs	device takeover	✓	
user activity	browsing	user profiling	✓	
	geo location	protected info	✓	
corporate data	sensitive data	corporate secrets	✓	
	credentials		✓	

Candidates

- (Recommended) apps with a reasonable user base
- Available on playstore

Type	Example	Reason	Detect	Users
user PII	email	PII, phone owner	✓	200k
	user profile	PII, phone owner	✓	5.5b
user secrets	password		✓	100k
	bank info		✓	6m
app secrets	auth token	account takeover	✓	1m
	credentials	account takeover	✓	100m
	protocol IDs	device takeover	✓	10m
user activity	browsing	user profiling	✓	100m
	geo location	protected info	✓	110m
corporate data	sensitive data	corporate secrets	✓	511m
	credentials		✓	1m

Hunting for bugs: the approach

1. Categorize issues and list relevant candidates

Hunting for bugs: the approach

1. Categorize issues and list relevant candidates
2. Detect the issue, a manual step
   - 2.1 categorize app to generalize the issue
   - 2.2 find reproduction patterns
   - 2.3 abstract reproduction patterns for automation

Hunting for bugs: the approach

1. Categorize issues and list relevant candidates
2. Detect the issue, a manual step
   - 2.1 categorize app to generalize the issue
   - 2.2 find reproduction patterns
   - 2.3 abstract reproduction patterns for automation
3. Run the automation on any installed app

Hunting for bugs: the approach

1. Categorize issues and list relevant candidates
2. Detect the issue, a manual step
   - 2.1 categorize app to generalize the issue
   - 2.2 find reproduction patterns
   - 2.3 abstract reproduction patterns for automation
3. Run ~~the automation~~ on any installed app
4. Check each app with automated brain ON

Hunting for bugs: the approach

1. Categorize issues and list relevant candidates
2. Detect the issue, a manual step
   - 2.1 categorize app to generalize the issue
   - 2.2 find reproduction patterns
   - 2.3 abstract reproduction patterns for automation
3. Run ~~the automation~~ on any installed app
4. Check each app with automated brain ON
5. Restart at 1 or 2.1

Hunting for bugs: the approach

1. Categorize issues and list relevant candidates
2. Detect the issue, a manual step
   - 2.1 categorize app to generalize the issue
   - 2.2 find reproduction patterns
   - 2.3 abstract reproduction patterns for automation
3. Run ~~the automation~~ on any installed app
4. Check each app with automated brain ON
5. Restart at 1 or 2.1

- Next sections: interactive session
  - detection patterns
  - fixes

Extracting PII from logs

- In order to ease debug, logs are often verbose

Extracting PII from logs

- In order to ease debug, logs are often verbose
- Insecure approaches
  - to not impact storage size, app stores logs on external storage
  - to ease debugging, logs contains user PII
  - to ease debugging, logs contains links to user identifiers

Extracting PII from logs

- In order to ease debug, logs are often verbose
- Insecure approaches
  - to not impact storage size, app stores logs on external storage
  - to ease debugging, logs contains user PII
  - to ease debugging, logs contains links to user identifiers
- Examples

```
m_strLogin \00\00\00#\00\00\00#\EMAIL\00#m_strLoginHashNew
  \00\00\00#\00\00\00!\00HASH_PWD00#m_strUserId
  \00\00\00#\00\00\00%\00e5f9c327-a251-45fe-b42f-29904cecbb62
https://AAAAAAA/v2/account?api_key=BBBBBBB&account_token=CCCCC
```

Extracting PII from logs

- In order to ease debug, logs are often verbose
- Insecure approaches
  - to not impact storage size, app stores logs on external storage
  - to ease debugging, logs contains user PII
  - to ease debugging, logs contains links to user identifiers
- Examples

```
m_strLogin \00\00\00#\00\00\00#\EMAIL\00#m_strLoginHashNew
  \00\00\00#\00\00\00!\00HASH_PWD00#m_strUserId
  \00\00\00#\00\00\00%\00e5f9c327-a251-45fe-b42f-29904cecbb62
https://AAAAAAA/v2/account?api_key=BBBBBBB&account_token=CCCCC
```

- Detection method

Extracting PII from logs

- In order to ease debug, logs are often verbose
- Insecure approaches
  - to not impact storage size, app stores logs on external storage
  - to ease debugging, logs contains user PII
  - to ease debugging, logs contains links to user identifiers
- Examples

```
m_strLogin \00\00\00#\00\00\00#\EMAIL\00#m_strLoginHashNew
  \00\00\00#\00\00\00!\00HASH_PWD00#m_strUserId
  \00\00\00#\00\00\00%\00e5f9c327-a251-45fe-b42f-29904cecbb62
https://AAAAAAA/v2/account?api_key=BBBBBBB&account_token=CCCCC
```

- Detection method
  - grep for email address
  - grep for keywords like “auth” or “token”

Extracting PII from logs

- In order to ease debug, logs are often verbose
- Insecure approaches
  - to not impact storage size, app stores logs on external storage
  - to ease debugging, logs contains user PII
  - to ease debugging, logs contains links to user identifiers
- Examples

```
m_strLogin \00\00\00#\00\00\00#\EMAIL\00#m_strLoginHashNew
  \00\00\00#\00\00\00!\00HASH_PWD00#m_strUserId
  \00\00\00#\00\00\00%\00e5f9c327-a251-45fe-b42f-29904cecbb62
https://AAAAAAA/v2/account?api_key=BBBBBBB&account_token=CCCCC
```

- Detection method
  - grep for email address
  - grep for keywords like “auth” or “token”

- clean your logs from user identifiers
- keep your logs private and rotate them

Extracting PII thanks to . . . naming conventions

- In case of multi-accounts, app needs to separate datasets

Extracting PII thanks to . . . naming conventions

- In case of multi-accounts, app needs to separate datasets
- Insecure approaches
  - user email as folder name
  - user ID as folder name

Extracting PII thanks to . . . naming conventions

- In case of multi-accounts, app needs to separate datasets
- Insecure approaches
  - user email as folder name
  - user ID as folder name
- Examples

```
/sdcard/Android/data/APP/file/julien.thomas@protektoid.com/FOLDER/
/sdcard/Android/data/APP/cache/offline/SOCIAL_ID/
```

Extracting PII thanks to . . . naming conventions

- In case of multi-accounts, app needs to separate datasets
- Insecure approaches
  - user email as folder name
  - user ID as folder name
- Examples

```
/sdcard/Android/data/APP/file/julien.thomas@protektoid.com/FOLDER/
/sdcard/Android/data/APP/cache/offline/SOCIAL_ID/
```

- Detection method

Extracting PII thanks to . . . naming conventions

- In case of multi-accounts, app needs to separate datasets
- Insecure approaches
  - user email as folder name
  - user ID as folder name
- Examples

```
/sdcard/Android/data/APP/file/julien.thomas@protektoid.com/FOLDER/
/sdcard/Android/data/APP/cache/offline/SOCIAL_ID/
```

- Detection method
  - find for email address
  - find for social network IDs

Extracting PII thanks to . . . naming conventions

- In case of multi-accounts, app needs to separate datasets
- Insecure approaches
  - user email as folder name
  - user ID as folder name
- Examples

```
/sdcard/Android/data/APP/file/julien.thomas@protektoid.com/FOLDER/
/sdcard/Android/data/APP/cache/offline/SOCIAL_ID/
```

- Detection method
  - find for email address
  - find for social network IDs

- hash of (user email, user ID) as folder name
- local_id $\mapsto$ user mapping in private database

User Profiling

- Raw IP address is not a sufficient leak (too easy)
  - need to identify when, where, what

User Profiling

- Raw IP address is not a sufficient leak (too easy)
  - need to identify when, where, what
- Insecure approaches
  - social activity (leak leads to PII disclosure)
    - cache contains picture header of each visited pages, timestamp being the date of the first visit
    - cache contains picture of user profiles, oldest one being the app user

User Profiling

- Raw IP address is not a sufficient leak (too easy)
  - need to identify when, where, what
- Insecure approaches
  - social activity (leak leads to PII disclosure)
    - cache contains picture header of each visited pages, timestamp being the date of the first visit
    - cache contains picture of user profiles, oldest one being the app user
  - web browsing
    - cache contains ... webview cache, that is basically the whole page

User Profiling

- Raw IP address is not a sufficient leak (too easy)
  - need to identify when, where, what
- Insecure approaches
  - social activity (leak leads to PII disclosure)
    - cache contains picture header of each visited pages, timestamp being the date of the first visit
    - cache contains picture of user profiles, oldest one being the app user
  - web browsing
    - cache contains ... webview cache, that is basically the whole page

assert the behavior of underlying components

Playing with geo locations

- When dealing with maps and GPS, app often needs lot of data

Playing with geo locations

- When dealing with maps and GPS, app often needs lot of data
- Insecure approaches
  - assume all GEO-related data are similar

Playing with geo locations

- When dealing with maps and GPS, app often needs lot of data
- Insecure approaches
  - assume all GEO-related data are similar
- Examples
  - save user current location in shared config for next app launch
  - save user databases “next to” map files (logically linked)

```
.ext.profile@addresses/UID/db.sqlite //serialized
sqlite3 .. "SELECT quote(value) from items where id=23"|cut -d\' -f2|xxd -r -p|xxd -g1
```

Playing with geo locations

- When dealing with maps and GPS, app often needs lot of data
- Insecure approaches
  - assume all GEO-related data are similar
- Examples
  - save user current location in shared config for next app launch
  - save user databases “next to” map files (logically linked)

```
.ext.profile@addresses/UID/db.sqlite //serialized
sqlite3 .. "SELECT quote(value) from items where id=23"|cut -d\' -f2|xxd -r -p|xxd -g1
```

- Detection method
  - look for exposed databases (the 100-x%)
  - look for strange format similar to databases or hidden sqlite

Playing with geo locations

- When dealing with maps and GPS, app often needs lot of data
- Insecure approaches
  - assume all GEO-related data are similar
- Examples
  - save user current location in shared config for next app launch
  - save user databases “next to” map files (logically linked)

```
.ext.profile@addresses/UID/db.sqlite //serialized
sqlite3 .. "SELECT quote(value) from items where id=23"|cut -d\' -f2|xxd -r -p|xxd -g1
```

- Detection method
  - look for exposed databases (the 100-x%)
  - look for strange format similar to databases or hidden sqlite

Custom format can be read on another device.
Assert device data policy: *READ_EXTERNAL_STORAGE* does not equal *ACCESS_FINE_LOCATION*

Playing with geo locations (2)

- When dealing with maps and GPS, app often needs lot of data

Playing with geo locations (2)

- When dealing with maps and GPS, app often needs lot of data
- Insecure approaches
  - consider all GEO-related data are similar regarding integrity

Playing with geo locations (2)

- When dealing with maps and GPS, app often needs lot of data
- Insecure approaches
  - consider all GEO-related data are similar regarding integrity
- Examples

```
Res/db/items.dat: sqlite: uid|name|lon|lat|..
```

Playing with geo locations (2)

- When dealing with maps and GPS, app often needs lot of data
- Insecure approaches
  - consider all GEO-related data are similar regarding integrity
- Examples

```
Res/db/items.dat: sqlite: uid|name|lon|lat|..
```

- Detection method
  - look for exposed databases (the 100-x%)
  - look for strange format similar to databases (XML) (the 100-x%)

Playing with geo locations (2)

- When dealing with maps and GPS, app often needs lot of data
- Insecure approaches
  - consider all GEO-related data are similar regarding integrity
- Examples

```
Res/db/items.dat: sqlite: uid|name|lon|lat|..
```

- Detection method
  - look for exposed databases (the 100-x%)
  - look for strange format similar to databases (XML) (the 100-x%)

- if you think exposing is still fine, timestamp control cache
- never allow user input alteration

Extracting implicit confidential assets

- Documents are often saved on device only for offline support

Extracting implicit confidential assets

- Documents are often saved on device only for offline support
- Insecure approaches
  - consider documents are NOT sensitive
  - consider users do not need to know

Extracting implicit confidential assets

- Documents are often saved on device only for offline support
- Insecure approaches
  - consider documents are NOT sensitive
  - consider users do not need to know
- Examples
  - no option at all, hardcoded root path

Extracting implicit confidential assets

- Documents are often saved on device only for offline support
- Insecure approaches
  - consider documents are NOT sensitive
  - consider users do not need to know
- Examples
  - no option at all, hardcoded root path
  - hidden “Change the storage root directory”

Extracting implicit confidential assets

- Documents are often saved on device only for offline support
- Insecure approaches
  - consider documents are NOT sensitive
  - consider users do not need to know
- Examples
  - no option at all, hardcoded root path
  - hidden “Change the storage root directory”
- Detection Methods

Extracting implicit confidential assets

- Documents are often saved on device only for offline support
- Insecure approaches
  - consider documents are NOT sensitive
  - consider users do not need to know
- Examples
  - no option at all, hardcoded root path
  - hidden “Change the storage root directory”
- Detection Methods
  - monitor file changes

Extracting implicit confidential assets

- Documents are often saved on device only for offline support
- Insecure approaches
  - consider documents are NOT sensitive
  - consider users do not need to know
- Examples
  - no option at all, hardcoded root path
  - hidden “Change the storage root directory”
- Detection Methods
  - monitor file changes

- make it obvious or at least secure by default
- let user devices the storage root directory

Extracting explicit confidential data

- “Encrypted” documents are often saved on device only for offline support

Extracting explicit confidential data

- “Encrypted” documents are often saved on device only for offline support
- Insecure approaches
  - consider mobile storage as “Not a question”
  - consider encryption key as not locally available

Extracting explicit confidential data

- “Encrypted” documents are often saved on device only for offline support
- Insecure approaches
  - consider mobile storage as “Not a question”
  - consider encryption key as not locally available
- Examples

Extracting explicit confidential data

- “Encrypted” documents are often saved on device only for offline support
- Insecure approaches
  - consider mobile storage as “Not a question”
  - consider encryption key as not locally available
- Examples
  - file stored in app external folders

Extracting explicit confidential data

- “Encrypted” documents are often saved on device only for offline support
- Insecure approaches
  - consider mobile storage as “Not a question”
  - consider encryption key as not locally available
- Examples
  - file stored in app external folders
  - sensitive attachments automatically downloaded to */sdcard*

Extracting explicit confidential data

- “Encrypted” documents are often saved on device only for offline support
- Insecure approaches
  - consider mobile storage as “Not a question”
  - consider encryption key as not locally available
- Examples
  - file stored in app external folders
  - sensitive attachments automatically downloaded to */sdcard*
  - file preview automatically downloaded to */sdcard* (power of reusable and third party components)

Extracting explicit confidential data

- “Encrypted” documents are often saved on device only for offline support
- Insecure approaches
  - consider mobile storage as “Not a question”
  - consider encryption key as not locally available
- Examples
  - file stored in app external folders
  - sensitive attachments automatically downloaded to */sdcard*
  - file preview automatically downloaded to */sdcard* (power of reusable and third party components)
- Detection Methods

Extracting explicit confidential data

- “Encrypted” documents are often saved on device only for offline support
- Insecure approaches
  - consider mobile storage as “Not a question”
  - consider encryption key as not locally available
- Examples
  - file stored in app external folders
  - sensitive attachments automatically downloaded to */sdcard*
  - file preview automatically downloaded to */sdcard* (power of reusable and third party components)
- Detection Methods
  - monitor file changes

Extracting explicit confidential data

- “Encrypted” documents are often saved on device only for offline support
- Insecure approaches
  - consider mobile storage as “Not a question”
  - consider encryption key as not locally available
- Examples
  - file stored in app external folders
  - sensitive attachments automatically downloaded to */sdcard*
  - file preview automatically downloaded to */sdcard* (power of reusable and third party components)
- Detection Methods
  - monitor file changes

- always review data storage and data “format”
- we are talking about confidential data and paid feature ...

Extracting explicit confidential data (2)

- Conversation involves media exchanges (MMS, ..)

Extracting explicit confidential data (2)

- Conversation involves media exchanges (MMS, ..)
- Insecure approaches
  - any conversation media is shared media
  - heard of “[FR] Celebgate?”

Extracting explicit confidential data (2)

- Conversation involves media exchanges (MMS, ..)
- Insecure approaches
  - any conversation media is shared media
  - heard of “[FR] Celebgate?”
- Examples

```
/sdcard/../Images/Private
/sdcard/../Images/Sent
/sdcard/../APP/
/sdcard/Android/APP/ #not a conversion app per say
```

Extracting explicit confidential data (2)

- Conversation involves media exchanges (MMS, ..)
- Insecure approaches
  - any conversation media is shared media
  - heard of “[FR] Celebgate?”
- Examples

```
/sdcard/../Images/Private
/sdcard/../Images/Sent
/sdcard/../APP/
/sdcard/Android/APP/ #not a conversion app per say
```

- Detection Methods

Extracting explicit confidential data (2)

- Conversation involves media exchanges (MMS, ..)
- Insecure approaches
  - any conversation media is shared media
  - heard of “[FR] Celebgate?”
- Examples

```
/sdcard/../Images/Private
/sdcard/../Images/Sent
/sdcard/../APP/
/sdcard/Android/APP/ #not a conversion app per say
```

- Detection Methods
  - monitor file changes

Extracting explicit confidential data (2)

- Conversation involves media exchanges (MMS, ..)
- Insecure approaches
  - any conversation media is shared media
  - heard of “[FR] Celebgate?”
- Examples

```
/sdcard/../Images/Private
/sdcard/../Images/Sent
/sdcard/../APP/
/sdcard/Android/APP/ #not a conversion app per say
```

- Detection Methods
  - monitor file changes

- Assert the way media is generated: “Camera generated” differs from “Taken from Media folders”
- or discloses (and delete) them all

Injecting “private” data

- Integrity and Confidentiality are two different words

Injecting “private” data

- Integrity and Confidentiality are two different words
- Insecure approaches
  - assume that alteration and access are similar
  - assume that alteration and access are similar from a user point of view
  - triggers (implicit) broadcast event in the middle of the process

Injecting “private” data

- Integrity and Confidentiality are two different words
- Insecure approaches
  - assume that alteration and access are similar
  - assume that alteration and access are similar from a user point of view
  - triggers (implicit) broadcast event in the middle of the process
- Examples with user actions
  - two step internal actions with shared temporary assets
  - sharing payloadable assets with shared temporary assets

Injecting “private” data

- Integrity and Confidentiality are two different words
- Insecure approaches
  - assume that alteration and access are similar
  - assume that alteration and access are similar from a user point of view
  - triggers (implicit) broadcast event in the middle of the process
- Examples with user actions
  - two step internal actions with shared temporary assets
  - sharing payloadable assets with shared temporary assets
- Detection Methods

Injecting “private” data

- Integrity and Confidentiality are two different words
- Insecure approaches
  - assume that alteration and access are similar
  - assume that alteration and access are similar from a user point of view
  - triggers (implicit) broadcast event in the middle of the process
- Examples with user actions
  - two step internal actions with shared temporary assets
  - sharing payloadable assets with shared temporary assets
- Detection Methods
  - monitor file changes
  - some intuition

Injecting “private” data

- Integrity and Confidentiality are two different words
- Insecure approaches
  - assume that alteration and access are similar
  - assume that alteration and access are similar from a user point of view
  - triggers (implicit) broadcast event in the middle of the process
- Examples with user actions
  - two step internal actions with shared temporary assets
  - sharing payloadable assets with shared temporary assets
- Detection Methods
  - monitor file changes
  - some ~~intuition~~ luck

- heard of race conditions?
- heard of trusted inputs (outputs)?

Injecting “private” data (2)

- Integrity and Confidentiality are two different words

Injecting “private” data (2)

- Integrity and Confidentiality are two different words
- Insecure approaches
  - assume that alteration and access are similar
  - assume that alteration and access are similar from a user point of view

Injecting “private” data (2)

- Integrity and Confidentiality are two different words
- Insecure approaches
  - assume that alteration and access are similar
  - assume that alteration and access are similar from a user point of view
- Examples without user actions
  - files loaded from external storage and inserted in the app as is
  - files auto-sync without any considerations
  - non-offline files auto-sync without any considerations

Injecting “private” data (2)

- Integrity and Confidentiality are two different words
- Insecure approaches
  - assume that alteration and access are similar
  - assume that alteration and access are similar from a user point of view
- Examples without user actions
  - files loaded from external storage and inserted in the app as is
  - files auto-sync without any considerations
  - non-offline files auto-sync without any considerations
- Detection Methods

Injecting “private” data (2)

- Integrity and Confidentiality are two different words
- Insecure approaches
  - assume that alteration and access are similar
  - assume that alteration and access are similar from a user point of view
- Examples without user actions
  - files loaded from external storage and inserted in the app as is
  - files auto-sync without any considerations
  - non-offline files auto-sync without any considerations
- Detection Methods
  - monitor file changes
  - some intuition (reproducibility)

Injecting “private” data (2)

- Integrity and Confidentiality are two different words
- Insecure approaches
  - assume that alteration and access are similar
  - assume that alteration and access are similar from a user point of view
- Examples without user actions
  - files loaded from external storage and inserted in the app as is
  - files auto-sync without any considerations
  - non-offline files auto-sync without any considerations
- Detection Methods
  - monitor file changes
  - some intuition (reproducibility)

- heard of race conditions?
- heard of user decisions?

Extracting “shared private” data

- Data declassification always trigger extra declassification

Extracting “shared private” data

- Data declassification always trigger extra declassification
- Insecure approaches
  - consider that action traces are not sensitive
  - assume that interaction traces are not sensitive

Extracting “shared private” data

- Data declassification always trigger extra declassification
- Insecure approaches
  - consider that action traces are not sensitive
  - assume that interaction traces are not sensitive
- Examples
  - shared data can be analyzed to reverse protocols
  - shared data can be aggregated to reproduce protocols

```
2019/01/07 08:25:27.050 23437-23507 AddParticipant: [ID,..] ... name=NAME
```

Extracting “shared private” data

- Data declassification always trigger extra declassification
- Insecure approaches
  - consider that action traces are not sensitive
  - assume that interaction traces are not sensitive
- Examples
  - shared data can be analyzed to reverse protocols
  - shared data can be aggregated to reproduce protocols

```
2019/01/07 08:25:27.050 23437-23507 AddParticipant: [ID,..] ... name=NAME
```

- anonymize internal inputs unless you need long term traceability or identification

Extracting user secrets

- Some data in-app are de facto sensitive

Extracting user secrets

- Some data in-app are de facto sensitive
- Insecure approaches
  - assume that any in-app view allows data disclosure
  - assume that any in-app export allows data disclosure
  - assume that any rejected API action is user fault only

Extracting user secrets

- Some data in-app are de facto sensitive
- Insecure approaches
  - assume that any in-app view allows data disclosure
  - assume that any in-app export allows data disclosure
  - assume that any rejected API action is user fault only
- Examples
  - offline API and authentication call log
  - viewing/exporting bank statements

```
/sdcard/Download/WEIRD_NAME/BA.pdf
/sdcard/APP/account/month.csv
/sdcard/Android/...log
  "input":"{\"action\":\"provision\",\"username\":\"EMAIL\",\"firstname\":\"\",
  \"lastname\":\"\",\"password\":\"(INCORECT\_)PASSWORD\",\"devicename\":...}"
```

Extracting user secrets

- Some data in-app are de facto sensitive
- Insecure approaches
  - assume that any in-app view allows data disclosure
  - assume that any in-app export allows data disclosure
  - assume that any rejected API action is user fault only
- Examples
  - offline API and authentication call log
  - viewing/exporting bank statements

```
/sdcard/Download/WEIRD_NAME/BA.pdf
/sdcard/APP/account/month.csv
/sdcard/Android/...log
  "input":"{\"action\":\"provision\",\"username\":\"EMAIL\",\"firstname\":\"\",
  \"lastname\":\"\",\"password\":\"(INCORECT\_)PASSWORD\",\"devicename\":...}"
```

- assert data security before exporting them
- do not (try to) hide externalized data, it's worst

Outline

Analysis process

- Time consuming
  - manually install each app
  - manually sign up to each app

Analysis process

- Time consuming
  - manually install each app
  - manually sign up to each app
- Permanent retro-actions

Analysis process

- Time consuming
  - manually install each app
  - manually sign up to each app
- Permanent retro-actions
- Hard to standardize the approach
  - /sdcard direct storage
  - “.” prefixed naming
  - specific leaks

Analysis process

- Time consuming
  - manually install each app
  - manually sign up to each app
- Permanent retro-actions
- Hard to standardize the approach
  - /sdcard direct storage
  - “.” prefixed naming
  - specific leaks
- Often relies on the combo (install app) + (“find -mtime”) + (find/grep PII)
  - not the most pleasant job
  - better target the right app category

Classification of the issues

- Various vulnerable data
  - app private data
  - various user private data
  - various device protected assets

Classification of the issues

- Various vulnerable data
  - app private data
  - various user private data
  - various device protected assets
- Limited set of explanation
  - failure to R..M
  - failure to learn literature (race conditions, external storage exposure)

Classification of the issues

- Various vulnerable data
  - app private data
  - various user private data
  - various device protected assets
- Limited set of explanation
  - failure to R..M
  - failure to learn literature (race conditions, external storage exposure)
  - failure to define security policies
  - failure to assert components security
  - failure to assert attacker model

Classification of the issues

- Various vulnerable data
  - app private data
  - various user private data
  - various device protected assets
- Limited set of explanation
  - failure to R..M
  - failure to learn literature (race conditions, external storage exposure)
  - failure to define security policies
  - failure to assert components security
  - failure to assert attacker model
  - failure to perform security tests

Classification of the vendors

- Various vendor profiles

Classification of the vendors

- Various vendor profiles
- Responsive vendors
  - top ones, often in managed program but not only
  - issues acknowledged
  - issues acknowledged with “1 year license to the solution”

Classification of the vendors

- Various vendor profiles
- Responsive vendors
  - top ones, often in managed program but not only
  - issues acknowledged
  - issues acknowledged with “1 year license to the solution”
- Non trustworthy vendors
  - issues get rejected quickly as “out of scope”
  - issues get rejected quickly as “Won’t fix” (too much work to fix)
  - issues acknowledged but “Won’t fixes” as end of life (OEM)

Classification of the vendors

- Various vendor profiles
- Responsive vendors
  - top ones, often in managed program but not only
  - issues acknowledged
  - issues acknowledged with “1 year license to the solution”
- Non trustworthy vendors
  - issues get rejected quickly as “out of scope”
  - issues get rejected quickly as “Won’t fix” (too much work to fix)
  - issues acknowledged but “Won’t fixes” as end of life (OEM)
- “Ghost” vendors (or ostriches?)
  - lot of vendors simply fail to reply after 90 days

Classification of the vendors

- Various vendor profiles
- Responsive vendors
  - top ones, often in managed program but not only
  - issues acknowledged
  - issues acknowledged with “1 year license to the solution”
- Non trustworthy vendors
  - issues get rejected quickly as “out of scope”
  - issues get rejected quickly as “Won’t fix” (too much work to fix)
  - issues acknowledged but “Won’t fixes” as end of life (OEM)
- “Ghost” vendors (or ostriches?)
  - lot of vendors simply fail to reply after 90 days
- (soon) escalation to Google Play team

Classification of the vendors (2)

- Vendor view is “complex”

Classification of the vendors (2)

- Vendor view is “complex”
- Explaining the issue is sometimes really complex
  - really, need to illustrate why disclosure of user PII is an issue?
  - really, need to illustrate how sensitive data disclosure impacts user business?

Classification of the vendors (2)

- Vendor view is “complex”
- Explaining the issue is sometimes really complex
  - really, need to illustrate why disclosure of user PII is an issue?
  - really, need to illustrate how sensitive data disclosure impacts user business?
  - really, need to illustrate what path transveral vulnerability means?

Classification of the vendors (2)

- Vendor view is “complex”
- Explaining the issue is sometimes really complex
  - really, need to illustrate why disclosure of user PII is an issue?
  - really, need to illustrate how sensitive data disclosure impacts user business?
  - really, need to illustrate what path transveral vulnerability means?
    - avoid end-user business vs vendor business conflicts

Classification of the vendors (2)

- Vendor view is “complex”
- Explaining the issue is sometimes really complex
  - really, need to illustrate why disclosure of user PII is an issue?
  - really, need to illustrate how sensitive data disclosure impacts user business?
  - really, need to illustrate what path transveral vulnerability means?
    - avoid end-user business vs vendor business conflicts
- Often, it does not worth it (cf. rewards)

Classification of the vendors (2)

- Vendor view is “complex”
- Explaining the issue is sometimes really complex
  - really, need to illustrate why disclosure of user PII is an issue?
  - really, need to illustrate how sensitive data disclosure impacts user business?
  - really, need to illustrate what path transveral vulnerability means?
    - avoid end-user business vs vendor business conflicts
- Often, it does not worth it (cf. rewards)
- Often, changing the vision of the vendor is “hard”
  - no matter how critical the issue is

Conclusion

- “Dumpster diving” on Android is
  - similar to looking for a needle in a haystack except you do not know what the needle looks like

Conclusion

- “Dumpster diving” on Android is
  - similar to looking for a needle in a haystack except you do not know what the needle looks like
- Analysis is
  - partially automated on the pattern side
  - fully manual on the install side (and app selection)
  - once found, often reproducible

Conclusion

- “Dumpster diving” on Android is
  - similar to looking for a needle in a haystack except you do not know what the needle looks like
- Analysis is
  - partially automated on the pattern side
  - fully manual on the install side (and app selection)
  - once found, often reproducible
- Future work still numerous
  - real automation tool
  - extend to other types of disclosure and criteria
  - extend to multiple step attacks (API)

Conclusion

- “Dumpster diving” on Android is
  - similar to looking for a needle in a haystack except you do not know what the needle looks like
- Analysis is
  - partially automated on the pattern side
  - fully manual on the install side (and app selection)
  - once found, often reproducible
- Future work still numerous
  - real automation tool
  - extend to other types of disclosure and criteria
  - extend to multiple step attacks (API)
- Things strongly depends on the vendor replies
  - bug fixes
  - future work (your requests at goo.gl/5qzwAL)