Who am I?

Zsombor Kovács, IT security Geek, Founder of Hackersuli

“I was in the IT industry when Duke Nukem Forever was announced for the first time!”

In mobile app security since iOS 4.0

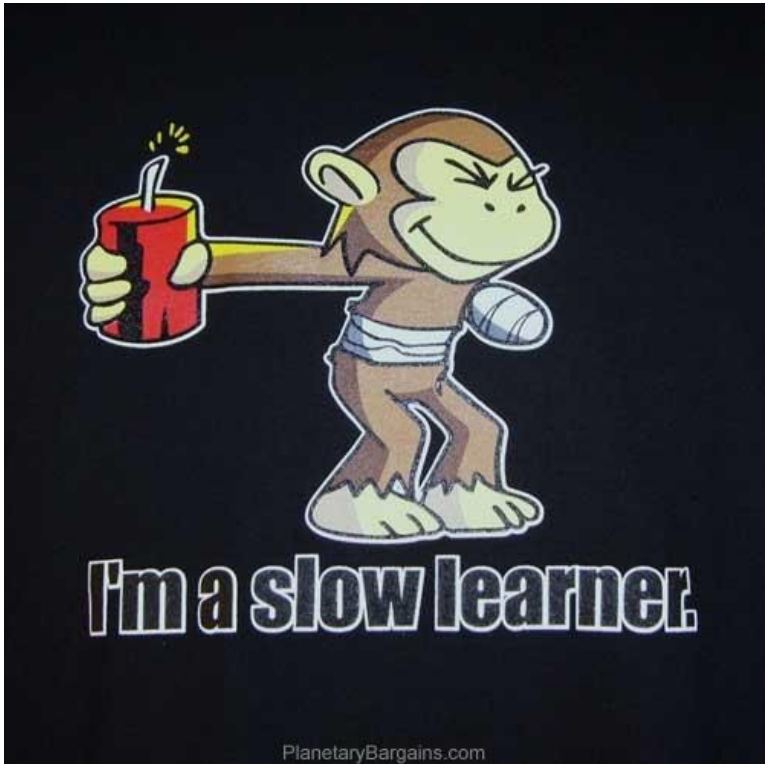
Don't Will Till the End, Fail Right at the Start!

Mis-categorize apps with sensitive data as Low-risk apps

- Mobile banking apps ('hey, that's just a web browser')
- OTP applications ('that's just only one factor')
- Anything that is not in App Store
- Anything that does not look sensitive at first sight

...and later on, reason with…

- “There is no requirement for cert pinning on Low-risk apps!”
- “This is a Low-risk app, why would anyone be interested?!”
- “It’s for internal use, no need to encrypt stuff”

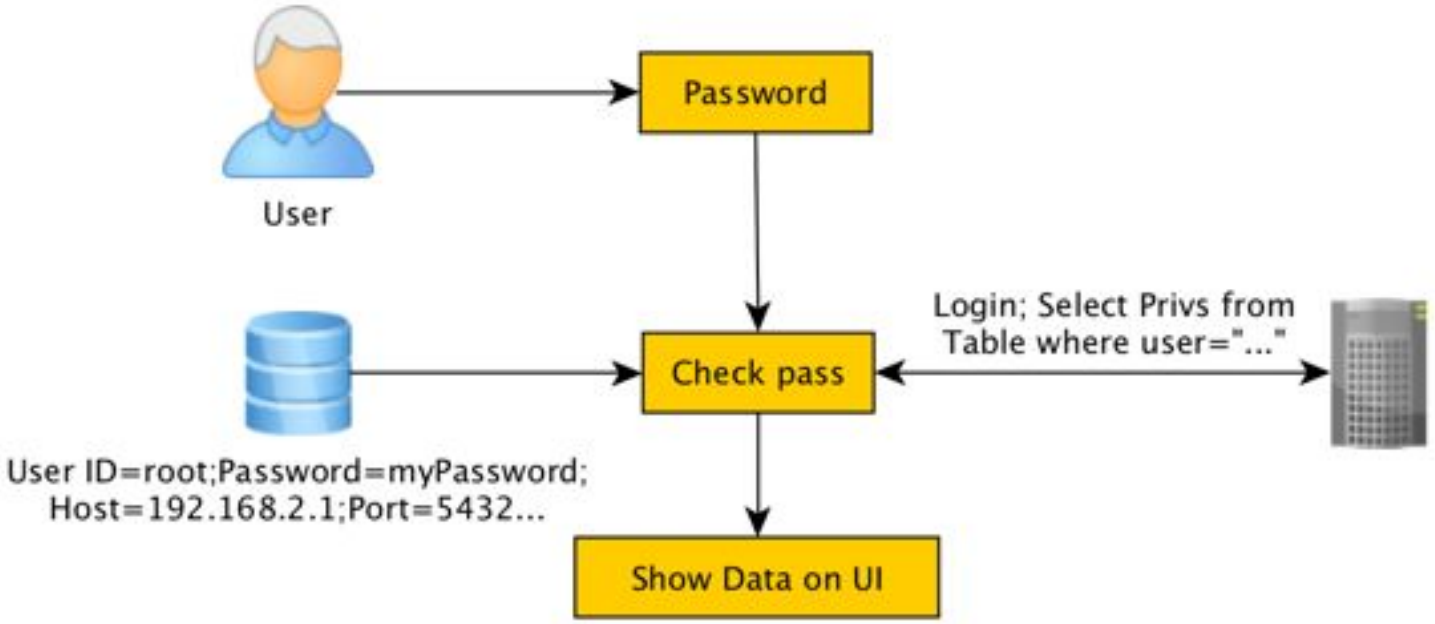


Insecure Architecture, the Root of All Evil

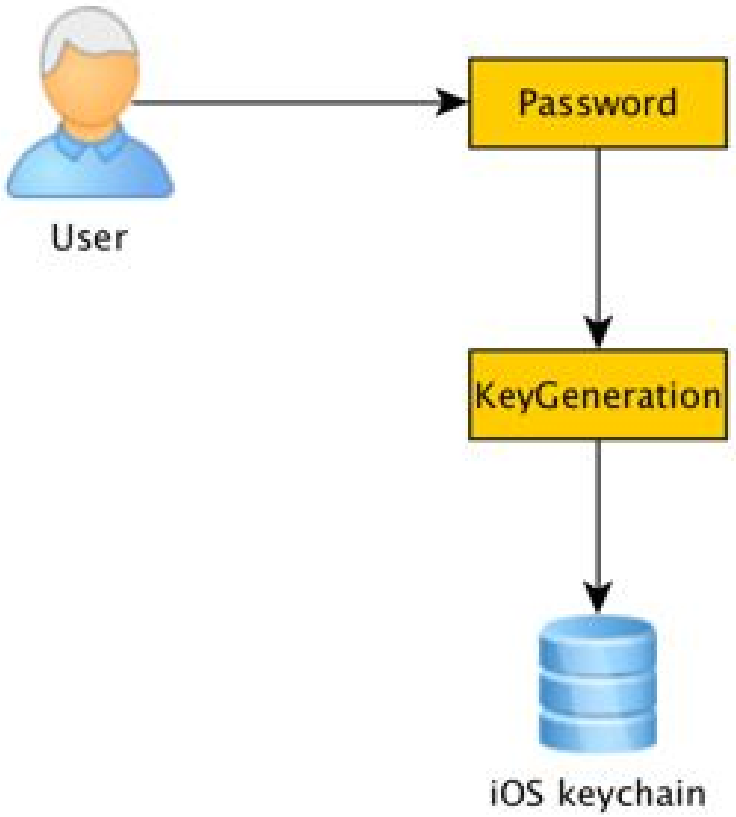
Mobile apps present new challenges

- Devices can be stolen
- Devices should not be trusted (hello iOS keychain!)
- Network environments should not be trusted
- Users can be *sigh*

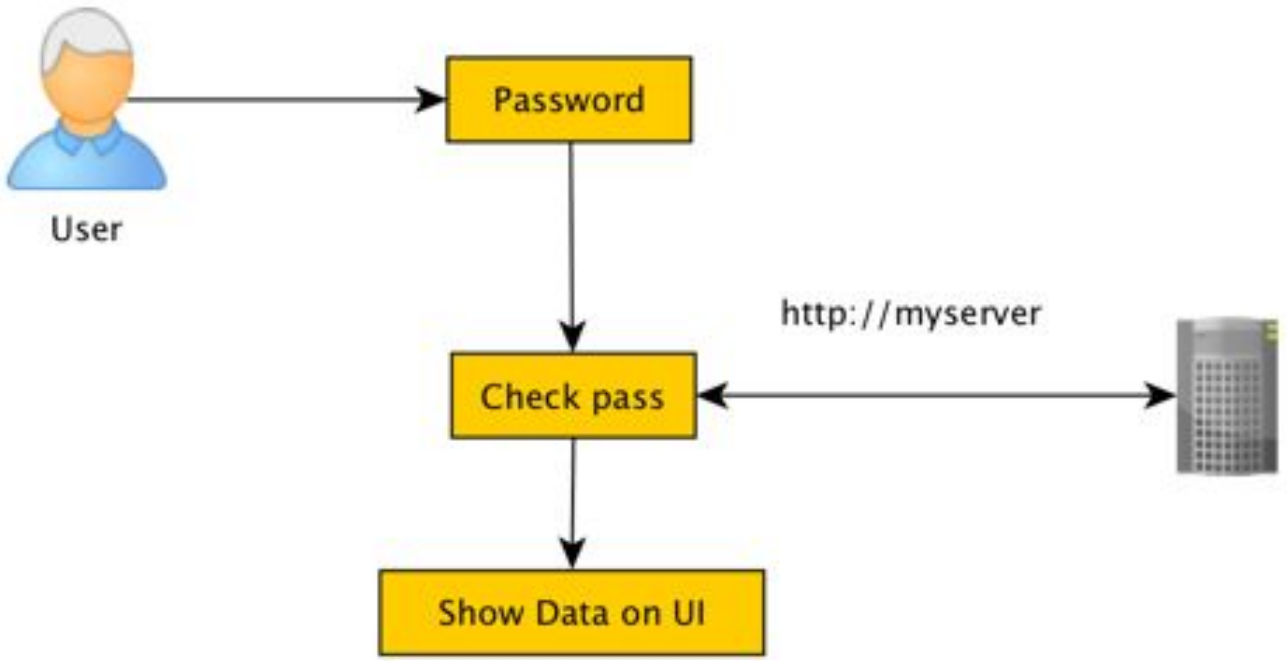
"The One Written by Thick Client Folks"



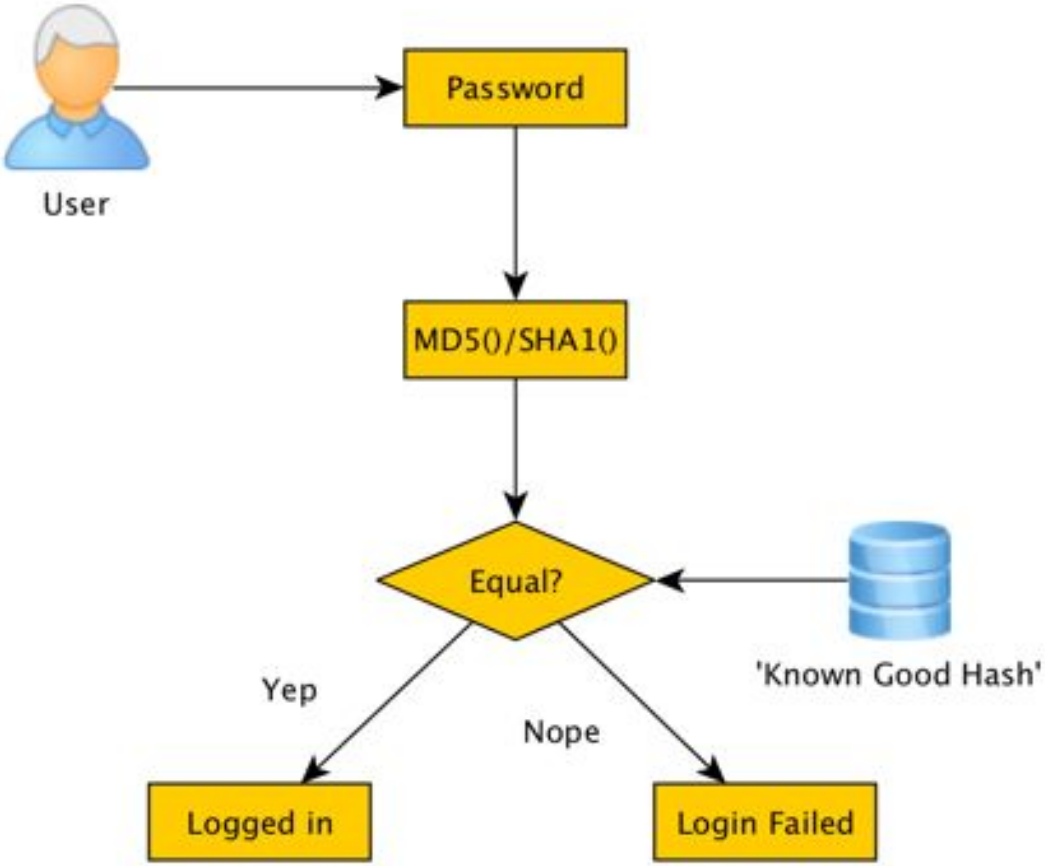
"The One With iOS Keychain"



"The One With the Open Door"



"The One With Local Password Check"



Insecure Architecture - Trusting the OS

Rule 1. Do not trust the OS.

Rule 2. Do NOT trust the OS.

- Don’t rely on root/jailbreak detection
- Don’t rely solely on OS cryptography services
- Don’t rely on OS ‘secure’ storage locations
- Don’t rely on OS CA store
- Don’t believe anything in general

“We Do Support Android 4.0”

Mobile OS’s are under constant development

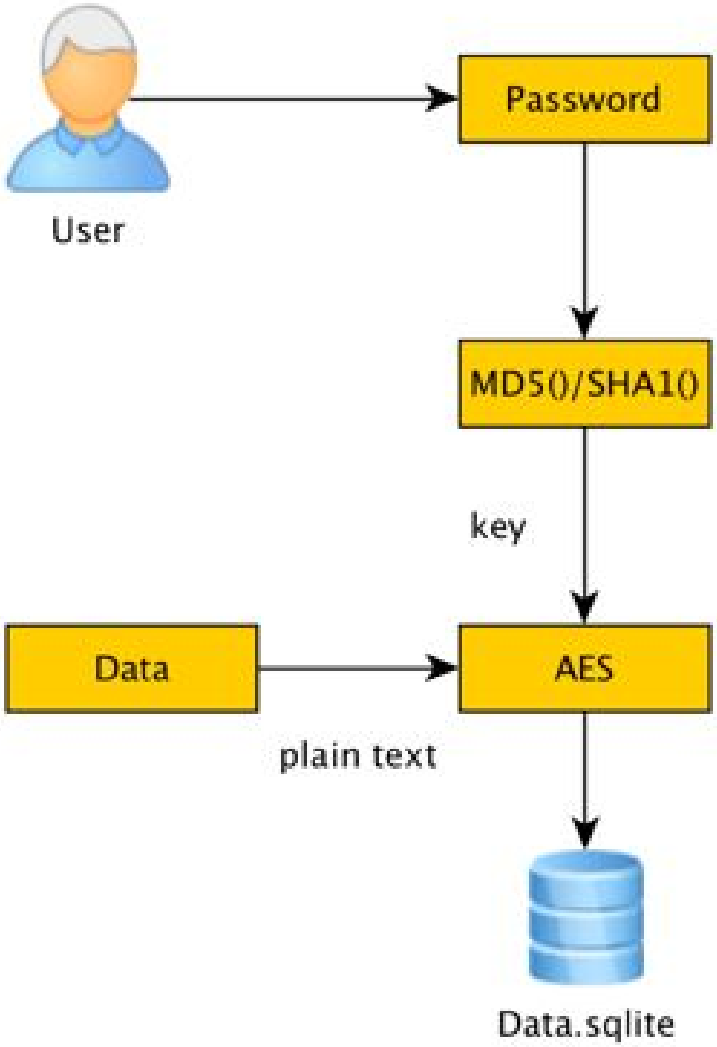
- Mostly an Android Issue
  - New features are available on later versions
  - Newer API revisions change
  - Newer versions are mostly backward compatible
  - So why use new features?
- Deprecated API calls are worked around

(Don't) Roll Your Own Crypto, a.k.a Re-Invent the Wheel

(Don't) Roll Your Own Crypto

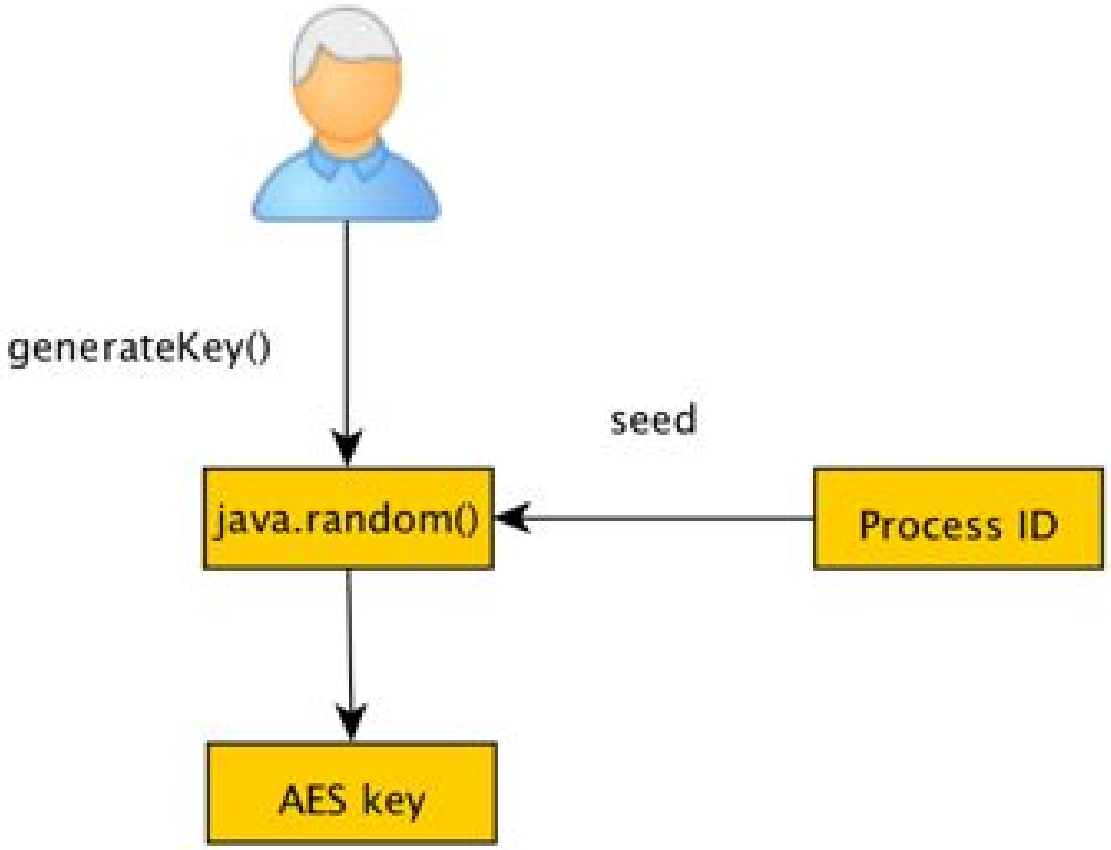
- Developers are (usually) not crypto folks
  - Crypto can be hard and to get it right, takes time and practice
  - Crypto documentation is mostly for crypto folks
- Use existing blocks
  - No substitute for understanding!
- Use cryptographic random API for crypto purposes
  - Do not forget to seed properly
  - Be careful what happens to the random numbers

Roll Your Own Crypto "The One With Key Generation"



Roll Your Own Crypto

"The One With So Many Things Wrong"



Roll Your Own Crypto - "The One With Misleading API calls"

```
1   CreatePasswordViewController - (void)okButtonPressed:(id)
2   [lots of checks for password length, complexity etc.]
3   MOV             R0, (_OBJC_IVAR_$_CreatePasswordViewController.leftPasswordField - 0x2FFC6)
4   LDR             R1, [SP,#0xC0+var_94]
5   ADD             R0, PC  ; UITextField *leftPasswordField;
6   LDR             R0, [R0] ; UITextField *leftPasswordField;
7   LDR             R0, [R6,R0]
8   BLX             _objc_msgSend
9   MOV             R7, R7
10  BLX             _objc_retainAutoreleasedReturnValue
11  MOV             R4, R0
12  MOV             R0, (selRef_hash - 0x2FFE0) ; selRef_hash
13  ADD             R0, PC ; selRef_hash
14  LDR             R1, [R0] ; "hash"
15  MOV             R0, R4
16  BLX             _objc_msgSend
17  MOV             R2, R0
18  MOV             R0, (selRef_storePassword_ - 0x2FFF4) ; selRef_storePassword_
19  ADD             R0, PC ; selRef_storePassword_
20  LDR             R1, [R0] ; "storePassword:"
21  MOV             R0, R6
22  BLX             _objc_msgSend
23  [...]
```

`CreatePasswordViewController.storePassword(leftPasswordField.hash())`

hash

Returns an integer that can be used as a table address in a hash table structure. (required)

```
- (NSUInteger)hash
```

Return Value

An integer that can be used as a table address in a hash table structure.

Insecure Data Storage

Data needs to be stored, right?

- Is it really necessary?
- Application sandboxes are not secure
  - Malware (especially with root privs)
  - Backups to Google, desktop etc.
  - OS encryption is transparent!
- SQLite
  - Most NAND flash based devices do *not* vacuum after ‘delete from’ statements

Insecure Data Storage "It Happens Even When Unexpected"

On iOS, whenever the home button is pressed, an automated screen shot is created

- Can contain sensitive stuff
- In later iOS versions, an AppDelegate function is invoked prior to the screen shot
  - `- (void)applicationWillResignActive:(UIApplication *)application`

Insecure Data Storage — iOS Keychain and Android Keystore

iOS keychain is not to be considered as a secure storage location

- When the app is removed, contents remain on the keychain
- On jailbroken instances, trivial to read out the contents

Android KeyStore is not much better either

- On rooted devices, it is trivial to read out the contents
- (Used to be) somewhat cumbersome to use for general data storage

Reliance on OS Services - WebView Issues

WebView is a popular and easy-to-use feature to build thick (-looking) apps using HTML5

- It’s a web browser though!
  - Use HTTPS with certificate pinning
  - Disable JavaScript, if not needed
- Cookies are stored in plain-text sqlite dbs
  - Strict session timeout is essential
- Disable caching!
  - Pragma: no-cache
  - Cache-control: no-cache

Reliance on OS Services - WebView Issues

All data is stored in %Appdir% sandbox

```
root@vbox86p:/data/data/com.myapp.uat/app_webview # ls
Cache
Cookies
Cookies-journal
Web Data
Web Data-journal
paks
```

Insecure Network Protection - Certificate Pinning

Certificate Pinning is hard to implement properly

- Common mistakes
  - Plain text connections
  - No pinning, reliance on OS cert store
  - Behold! So many things to check!
    - Check the ‘CN’ field
    - Check the name of the issuer (i.e. “Verisign, INC”)
    - Check the date of the certificate
  - No pinned cert? No problem, let’s proceed anyway
  - Some HTTPS connection are pinned, some are not

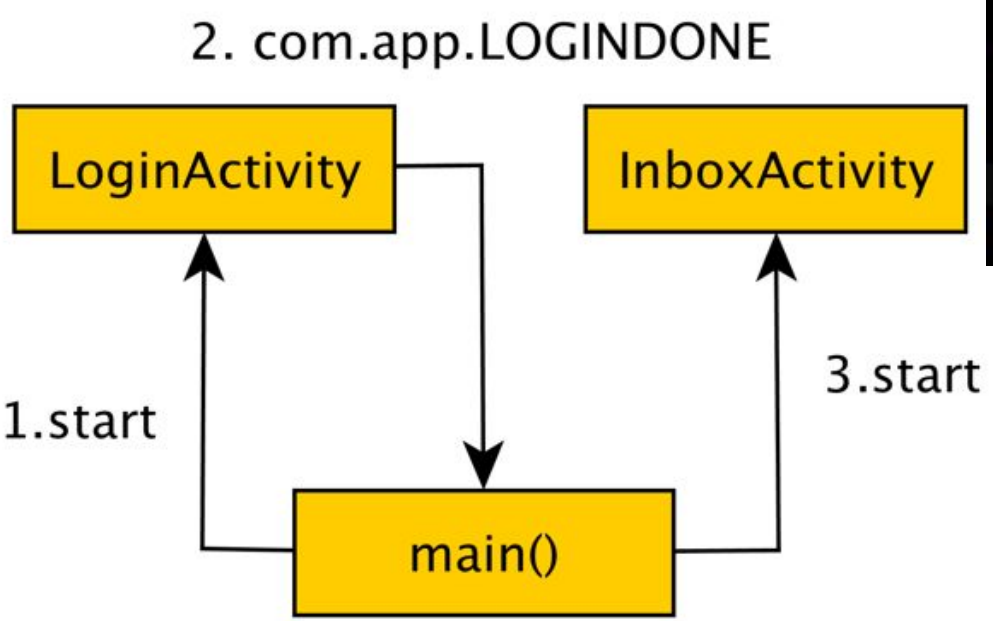
Insecure Use of OS Features — Android IPC Issues

- On Android, IPC is one of the key features, interaction between
  - Activities (GUI element)
  - Services (no GUI)
  - Content Providers
  - Broadcast Listeners
- Exportedness is a key concept
  - An exported component can be invoked from another app
  - Can be difficult to tell if a component is exported
- Injection issues
  - Some Content Providers can be queried (SQLi, anyone?)

Insecure Use of OS Features

Android IPC Issues, an example

The basic flow




```
<activity   android:name = "LoginActivity"
            android:label = "@2347261511">
</activity>

<activity   android:name = "InboxActivity"
            android:label = "@2323961231">
</activity>
```

```
root@android:/# am start com.example.myapp/com.example.myapp.InboxActivity
Starting: Intent { act=android.intent.action.MAIN cat=[
    android.intent.category.LAUNCHER] cmp=com.example.myapp/.InboxActivity}
```

App Fortification - When It's Really Needed

- Implement root/jailbreak detection
  - Don't just use one '+ (bool) AppDelegate isJailbroken'-like function. Implement the logic in many places, slightly changed for each one
- For protection against dynamic hooking, check the address of relevant functions
- Add self checks on the binary
  - Use runtime class integrity checks
  - Use integrity checks of the binary itself

Thank you

- Questions?