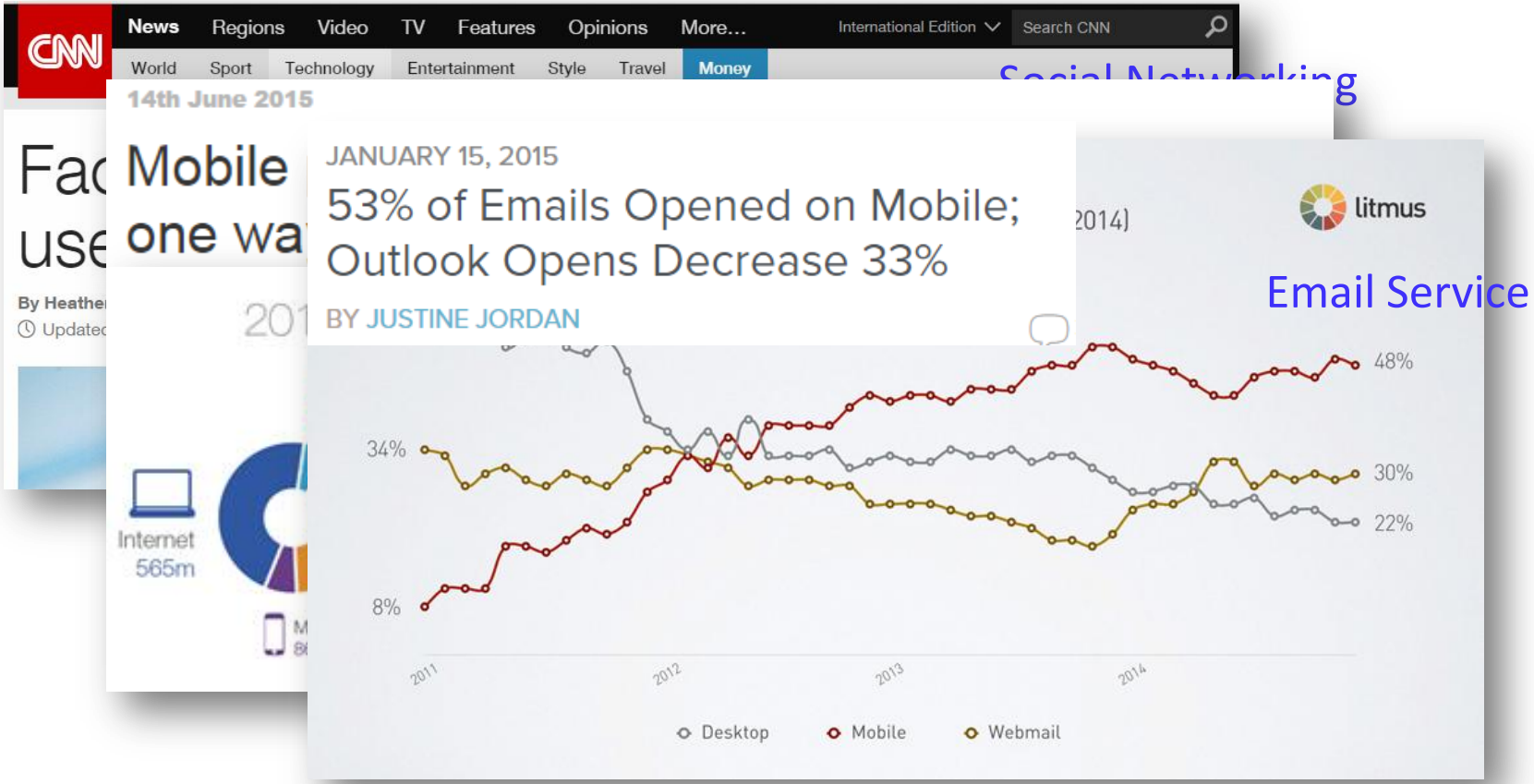
Authenticator Leakage Through Backup Channels on Android

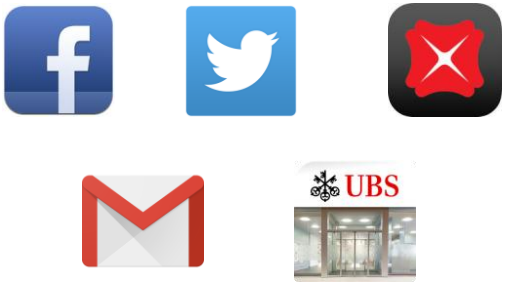
Guangdong Bai

National University of Singapore

Web services are increasingly delivered through mobile apps …

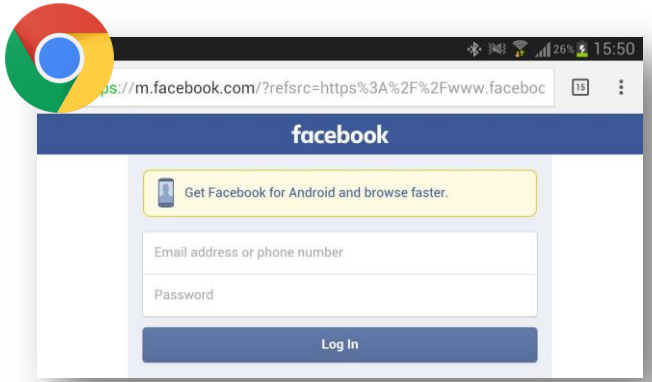


Can’t we simply use mobile browsers?



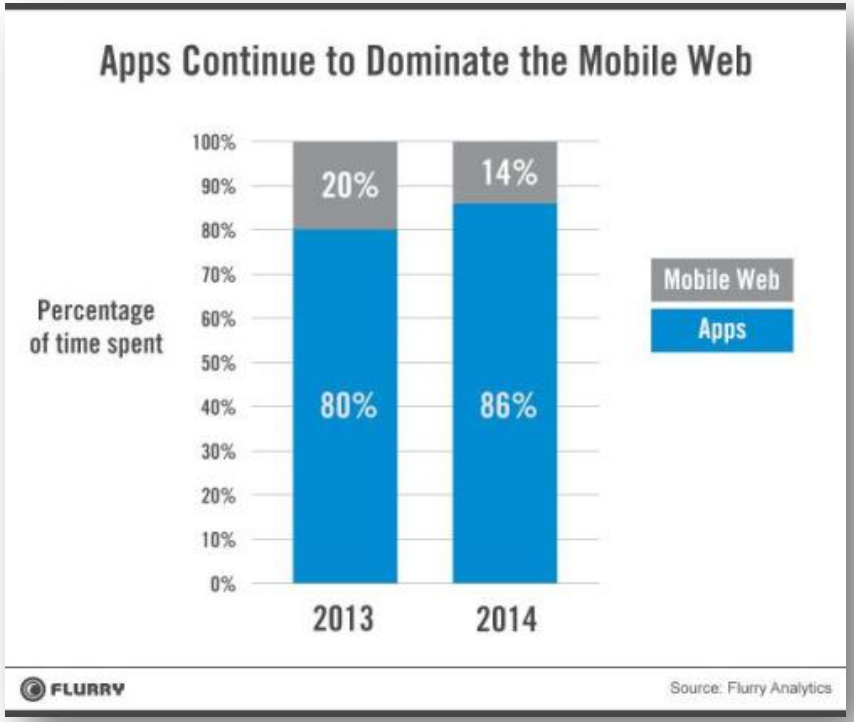
- ✓ Full use of device/APIs
- ✓ Less programming limitation
- ✓ Running faster

V.S.



- ✓ Cross platforms
- ✓ Reusable browser functionality (JS engine, …)
- ✓ Developed faster

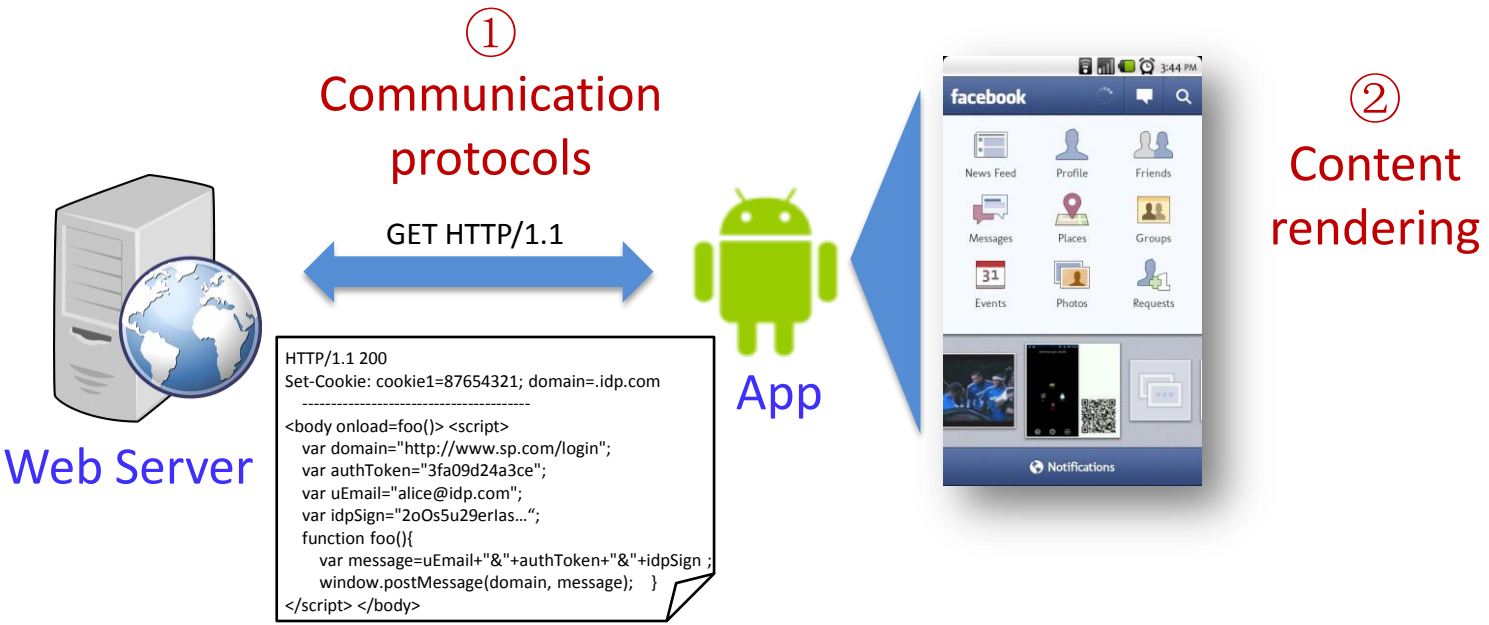
Can’t we simply use mobile browsers?



… the (mobile) browser has become a single application swimming in a sea of apps.

-- Flurry Insights

Therefore, mobile apps play the same role as web browsers



However, this is a non-trivial task…

- Security of communication protocols
  - Novel attack surface
  - Novel Trusted Computing Base (TCB)

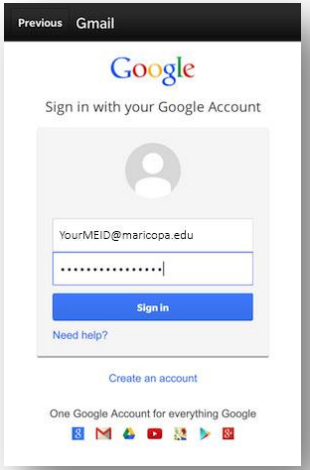
- Code injection attacks
  - Have been extensively studied [CCS'13, CCS'14, ESORICS'15]

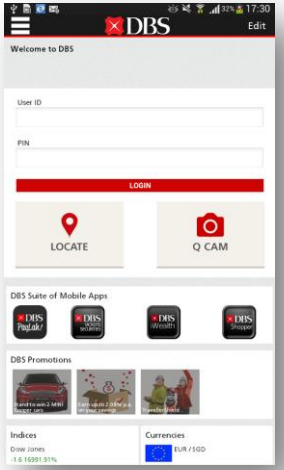
Focus of this talk: web authentication protocols on Android

- Implementation of web authentication schemes on Android
  - Authentication process
  - How authentication credentials (authenticators) are managed
- Backup channel: a new attack surface against web authentication on Android platform
  - Why backup is a dangerous functionality on Android
  - How to abuse backup channels
- Case studies and mitigations

Section 1. Web Authentication on Android

Web authentication: safeguard to web accounts





- Web Authentication
  - A process by server to confirm whether an entity (client) is who it declared
  - One of the mostly used web functionalities

How Android apps implement web authentication?

- Our investigation
  - Goal: to learn approaches contemporary apps use to implement their authentication schemes
  - Focus: how authenticators are managed
  - Methodology: we have manually analyzed top-ranked 100 apps on Google Play (by reverse engineering and traffic analysis)

Result summary

66 with authentication schemes

- Basic authentication (40)
- Single Sign-on (40)
- Android Account Manager (16)

34 without authentication schemes

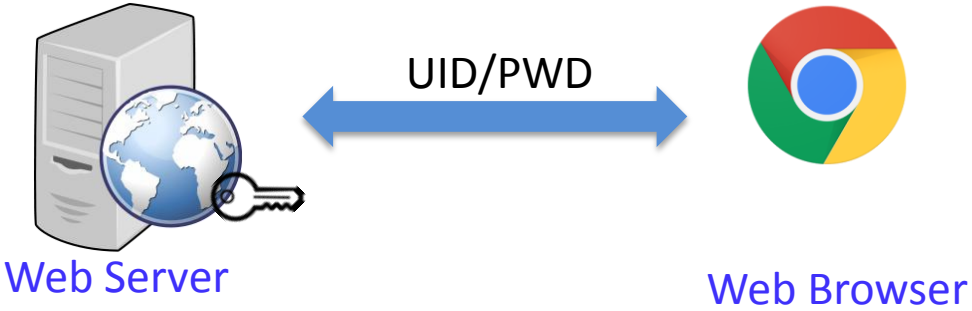
Standalone apps e.g., news browsers, maps and video players

Figure source: http://geektechreviews.com/wp-content/uploads/2015/07/Top-10-Free-Android-Apps-Must-Have.jpg

Web authentication scheme #1: Basic authentication

- Basic authentication stands for traditional authentication schemes on the basis of
  - Knowledge (e.g., a password and security questions)
    - 34 out of 40 apps use password-based schemes
  - Ownership (e.g., a hardware token and a mobile phone)
    - 6 out of 40 apps use SMS-based one time password schemes
  - Inherence (e.g., fingerprint and retinal pattern)
    - None is found
    - Fingerprint confidentiality at Black Hat US 2015 by Dr. Wei Tao

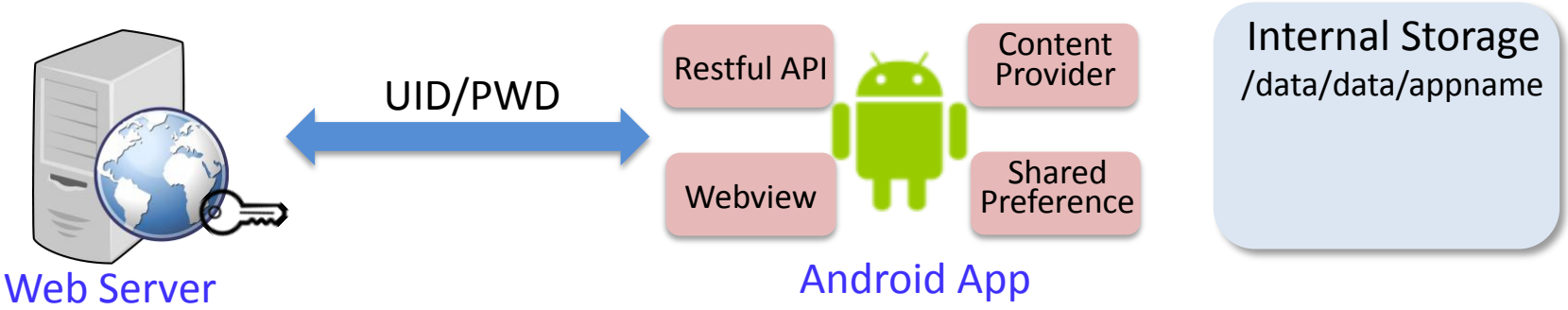
General process of basic authentication on desktop browsers



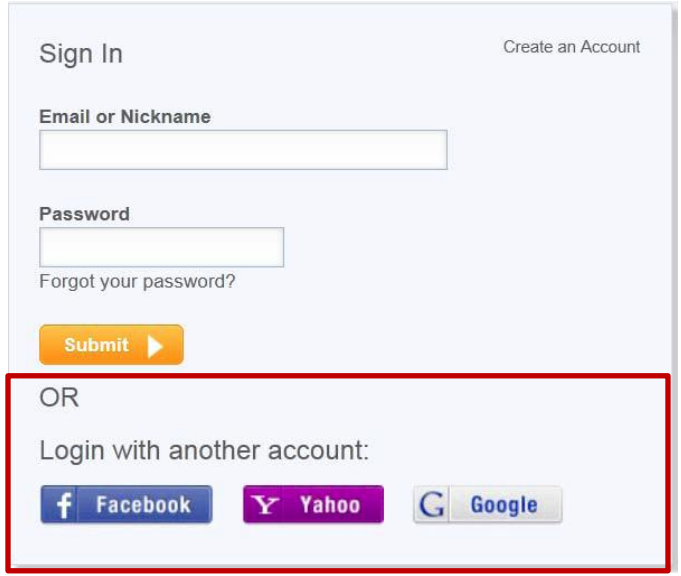
- ✓ Same origin policy (SOP)
- ✓ Cookie Protection
- ✓ …

- Authenticator
  - An authentication credential indicating client's login session
  - E.g., cookies, session ID, OAuth Token and OAuth Code

General process of basic authentication on Android apps

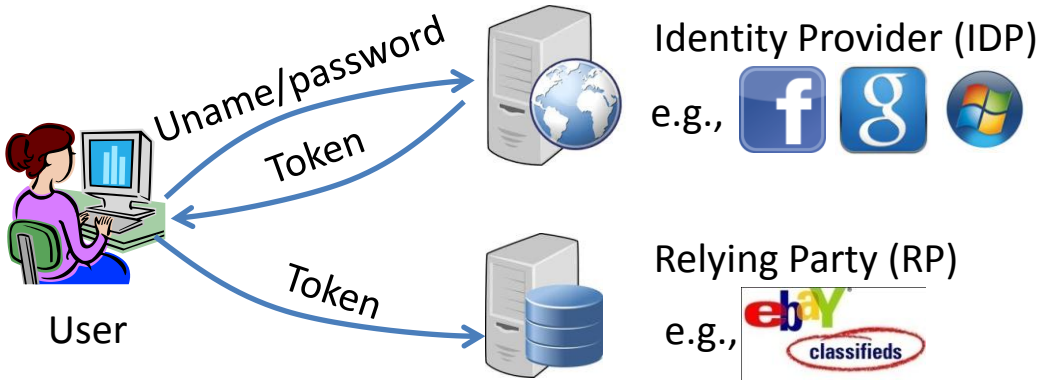


Web authentication scheme #2: Single Sign-on



- Single Sign-On (SSO)
  - A kerberos-like single credential authentication scheme
  - BrowserID (Mozilla)
  - Facebook Connect
    - 250+ Million users, 2,000,000 websites
  - OpenID
    - one billion users, 50,000 websites
  - …

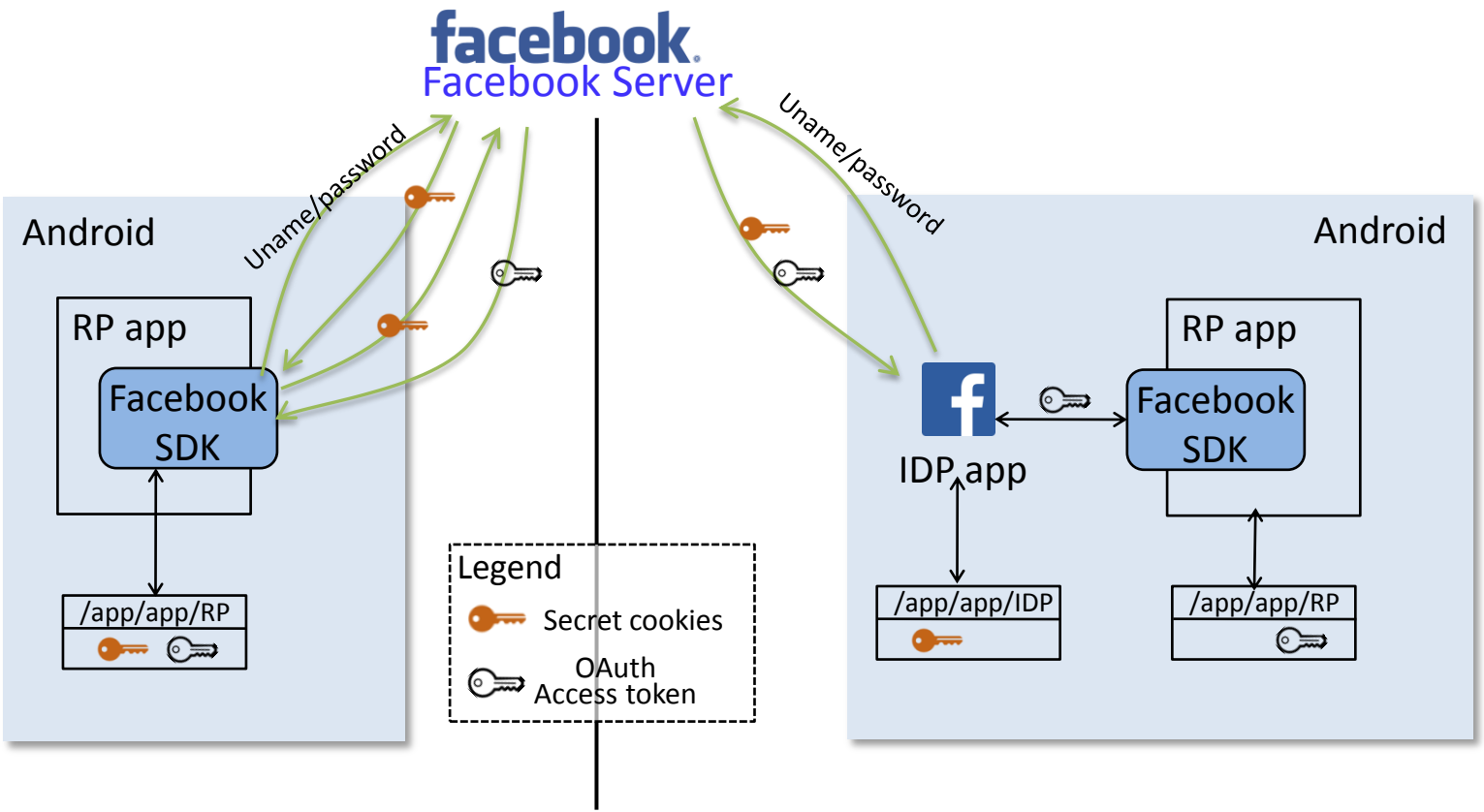
Three parties in SSO



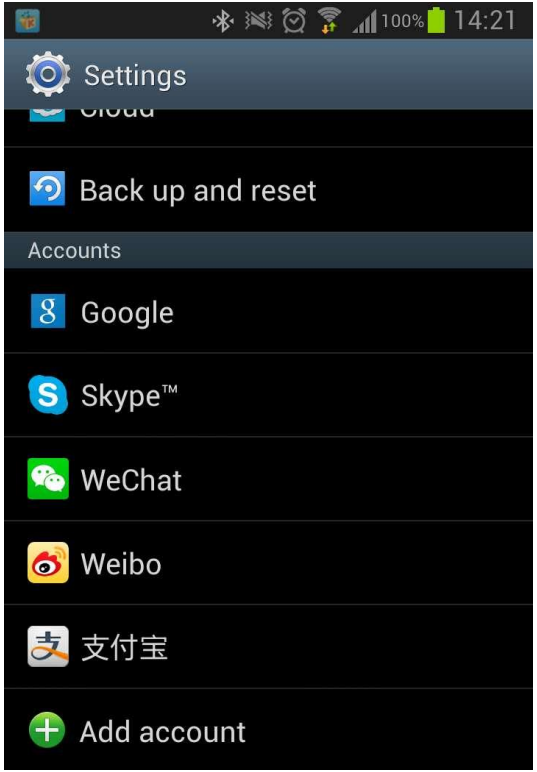
SSO in Android

- Relying Party (RP)
  - Application
- Identity provider (IDP)
  - SSO Service is released in form of SDK
  - E.g., Facebook Connect, Twitter ID

A concrete process: Facebook connect



Web authentication scheme #3: Android Account Manager



- Account Manager
  - An Android service which provides a delegated authentication service and centralized account/authenticator control
  - Pros
    - Simplifies the process for the developer
      - By implementing some interface
    - Can handle multiple token types for a single account
    - Automatically background update (SyncAdapters)

http://blog.udinic.com/2013/04/24/write-your-own-android-authenticator/

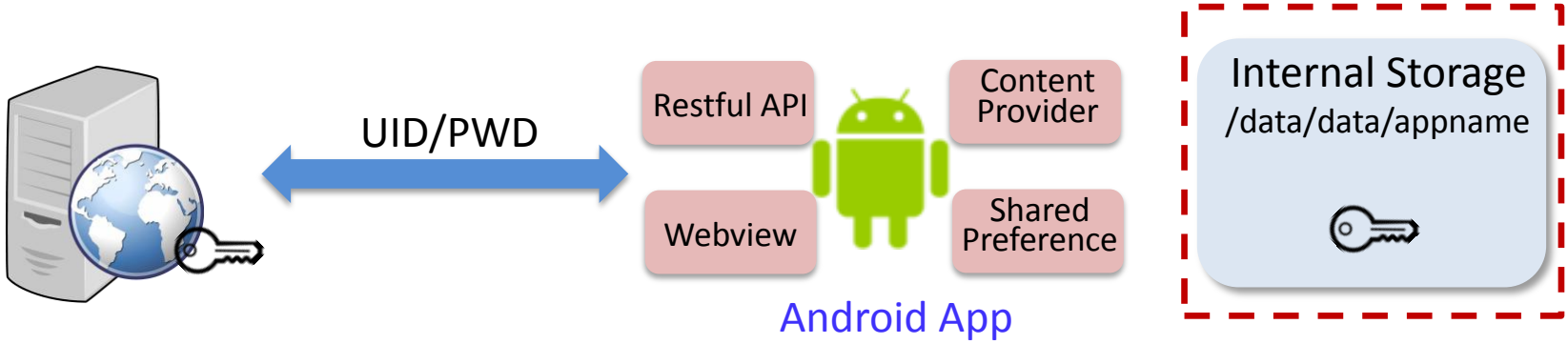
Briefing how Account Manager works

- Developer needs only to …
  - To create an *AccountAuthenticator*
    - Add accounts, account types, auth token
  - To create an Activity
    - Through which users enter credentials
- Account manager will …
  - Manage authenticators
    - Located in *account.db* in /data/system/users/0
  - Update authenticators on background

Security of authentication schemes

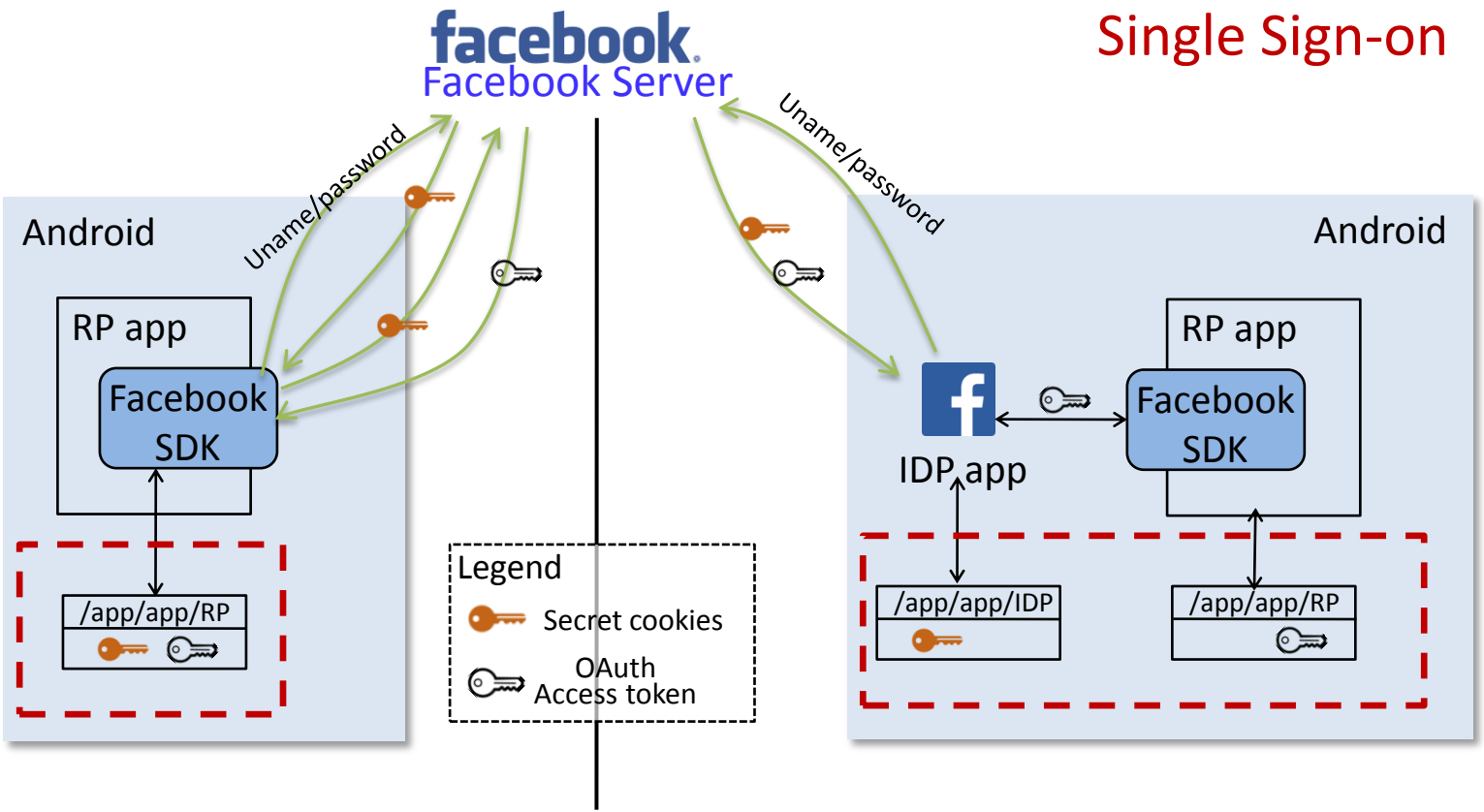
- Security of protocols in three layers
  - Design-level security: design and logic flaws
    - A notorious example: flaws in Needham-Schroeder protocol
    - Protocol verification: theorem proving (Proverif), model checking (PAT)
  - Implementation-level security
    - Implementation errors/bugs in the code
    - E.g., Google ID flaw: not all messages are cover in signature (IEEE S&P'12)
      Guessable authenticators (NDSS'13)
  - Infrastructure-level security
    - Exploits in the software stack (e.g., OS, file system) that the protocols rely upon
    - A previous study: password leakage through compromised ADB (Claud Xiao on HITCON'14)

Let's look at infrastructure-level security of web authentication on Android



Basic Authentication

Let's look at infrastructure-level security of web authentication on Android



Let’s look at infrastructure-level security of web authentication on Android

Basic Authentication

Single Sign-on

The owner app’s proprietary directory

/app/app/appname

```
root@grouper:/ # ls -l /data/data
drwx------ system   system            2014-04-25 16:53 android
drwxr-x--x u0_a116  u0_a116           2015-10-30 20:21 cal.byzm
drwxr-x--x u0_a345  u0_a345           2015-10-30 20:21 com.RZStudio.cube
drwxr-x--x u0_a356  u0_a356           2015-10-30 20:21 com.ahmetkizilay.a
drwxr-x--x u0_a78   u0_a78            2015-10-30 20:21 com.alipay.android
drwxr-x--x u0_a0    u0_a0             2014-03-26 14:21 com.android.backup
drwxr-x--x bluetooth bluetooth          2014-04-25 19:18 com.android.blue
drwxr-x--x u0_a28   u0_a28            2014-04-28 13:42 com.android.browse
drwxr-x--x u0 a29   u0 a29            2014-04-28 13:41 com.android.calcul
```

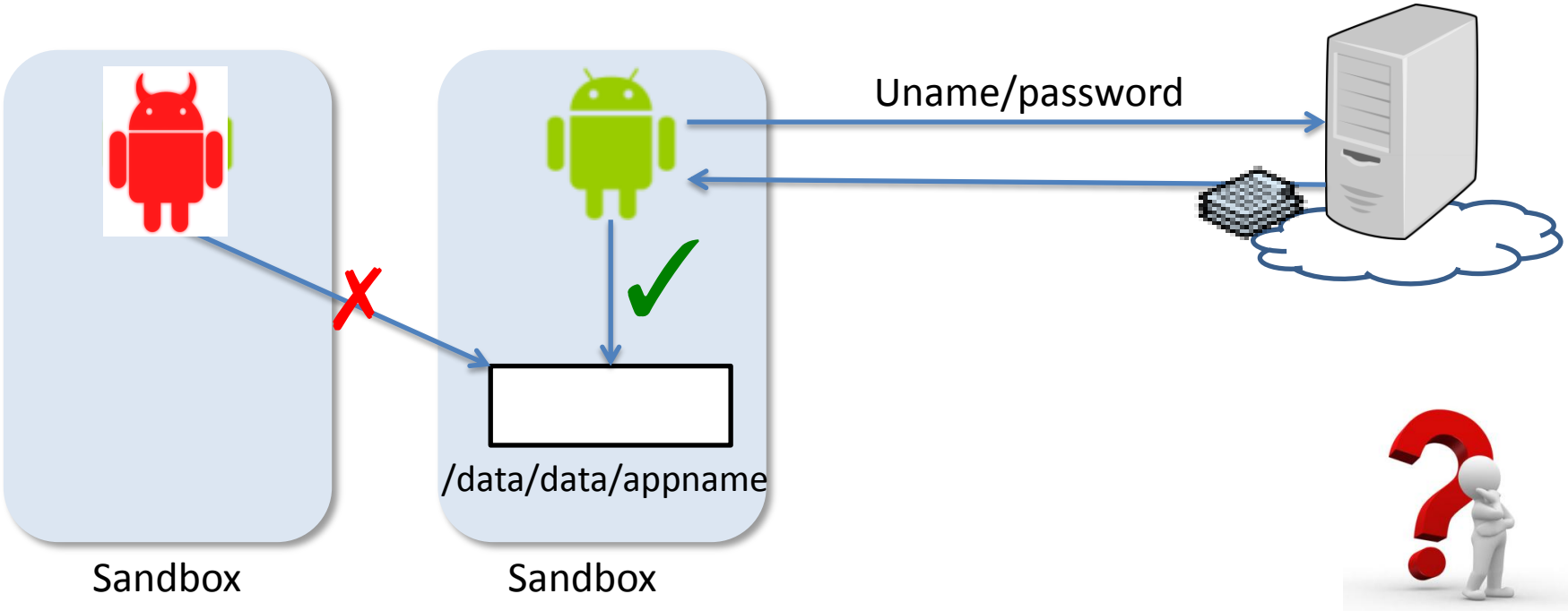
Account Manager

System directory

/data/system/users/0

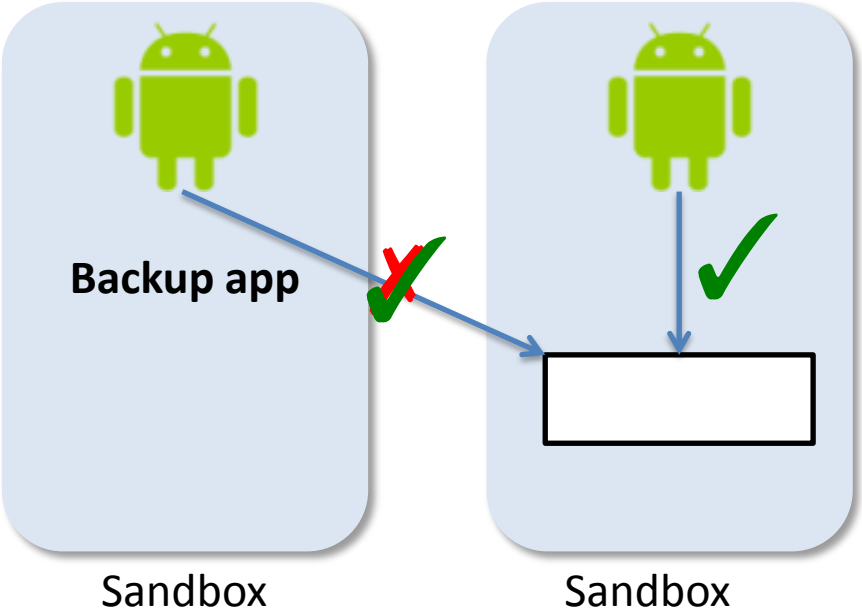
```
root@grouper:/ # ls -l /data/system/users/0
-rw-rw---- system   system     139264 2015-07-20 18:39 accounts.db
-rw------- system   system      62072 2015-07-20 18:39 accounts.db
-rw------- system   system        538 2015-11-01 15:37 appwidgets.x
-rw-rw---- system   system      67867 2015-11-01 15:36 package-rest
-rw------- system   system       3825 2014-03-25 17:05 photo.png
-rw------- system   system         99 2015-10-30 20:21 wallpaper in
```

Isolation Mechanism in Android



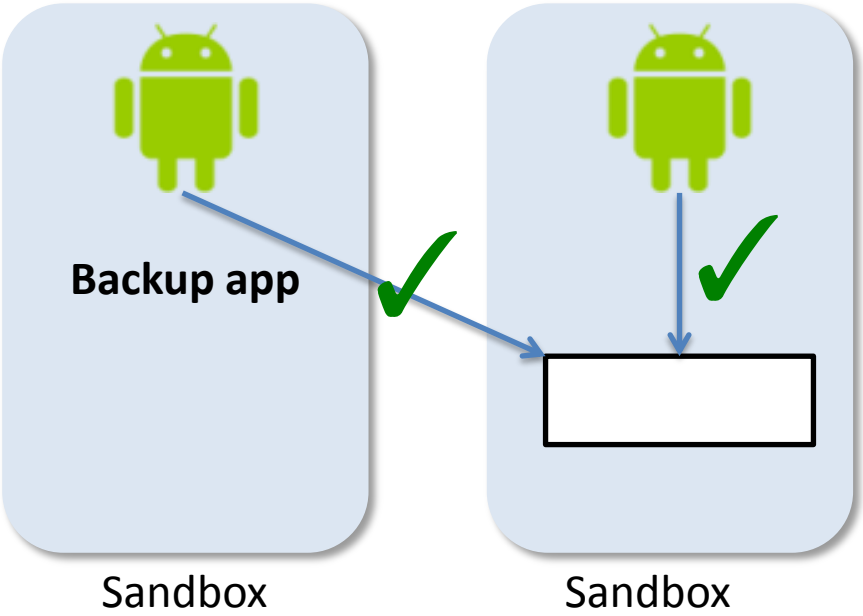
What if the sandbox is bypassed?

Backup functionality has to violate sandbox mechanism



Section 2. Backup on Android

Two ways to implement backup on Android

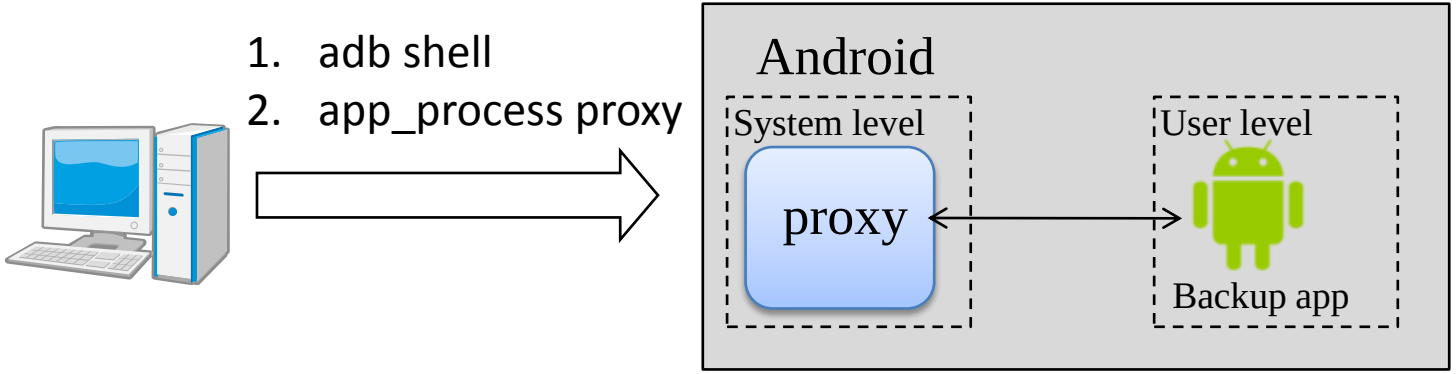


- Root-based backup
  - Root the device and grant root privilege to the backup apps
- ADB-based backup

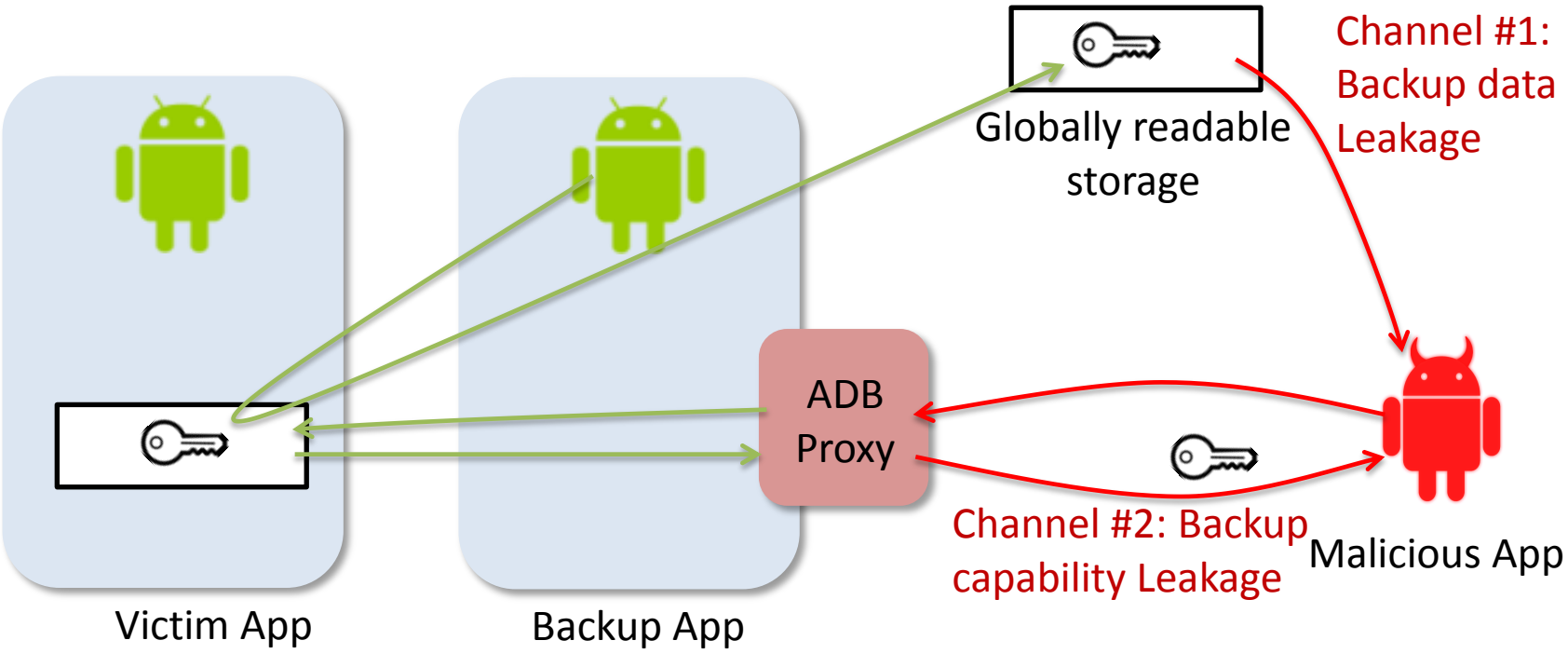
We consider only to backup an app's data located in its proprietary folder, instead of the user's data can be accessed through APIs like contacts and SMS messages

ADB-based backup

- ADB (Android Debug Bridge)
  - ADB is a versatile command line tool that lets users communicate with an emulator instance or connected Android-powered device.
  - Running on **system** level privilege
    - Root > system > user
- How does ADB-based backup work?



How backup can be a threat to authentication?



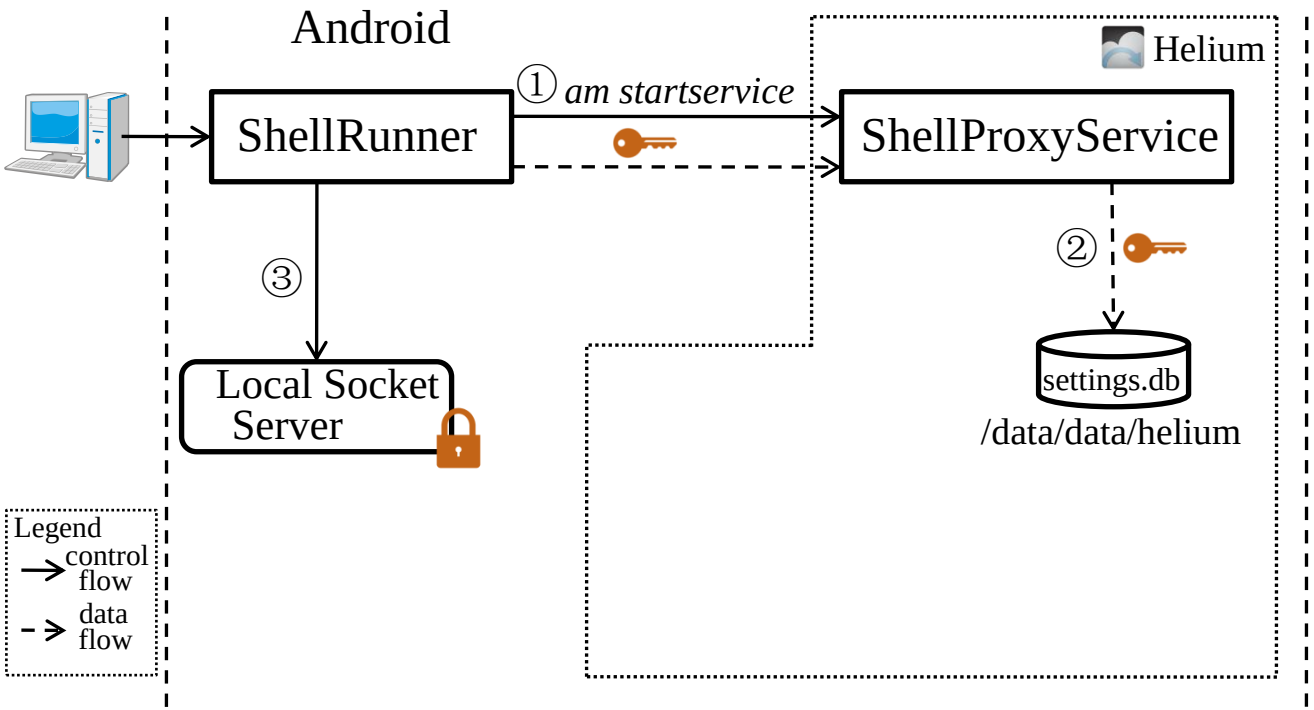
A summary of leakage through the existing backup apps

Category	Apps	Installs	Publicly accessible?	Backup data encrypted?	Compromised interfaces?	Leakage possible?
Root-based	My Backup	1,000,000 - 5,000,000	SD card	✗	--	✓
	Ultimate Backup	500,000 - 1,000,000	SD card	✗	--	✓
	Ease Backup	100,000 - 500,000	SD card	✗	--	✓
	Titanium Backup	10,000,000 - 50,000,000	SD card	✗	--	✓
ADB-based	Helium	1,000,000 - 5,000,000	SD card	✗	✓	✓

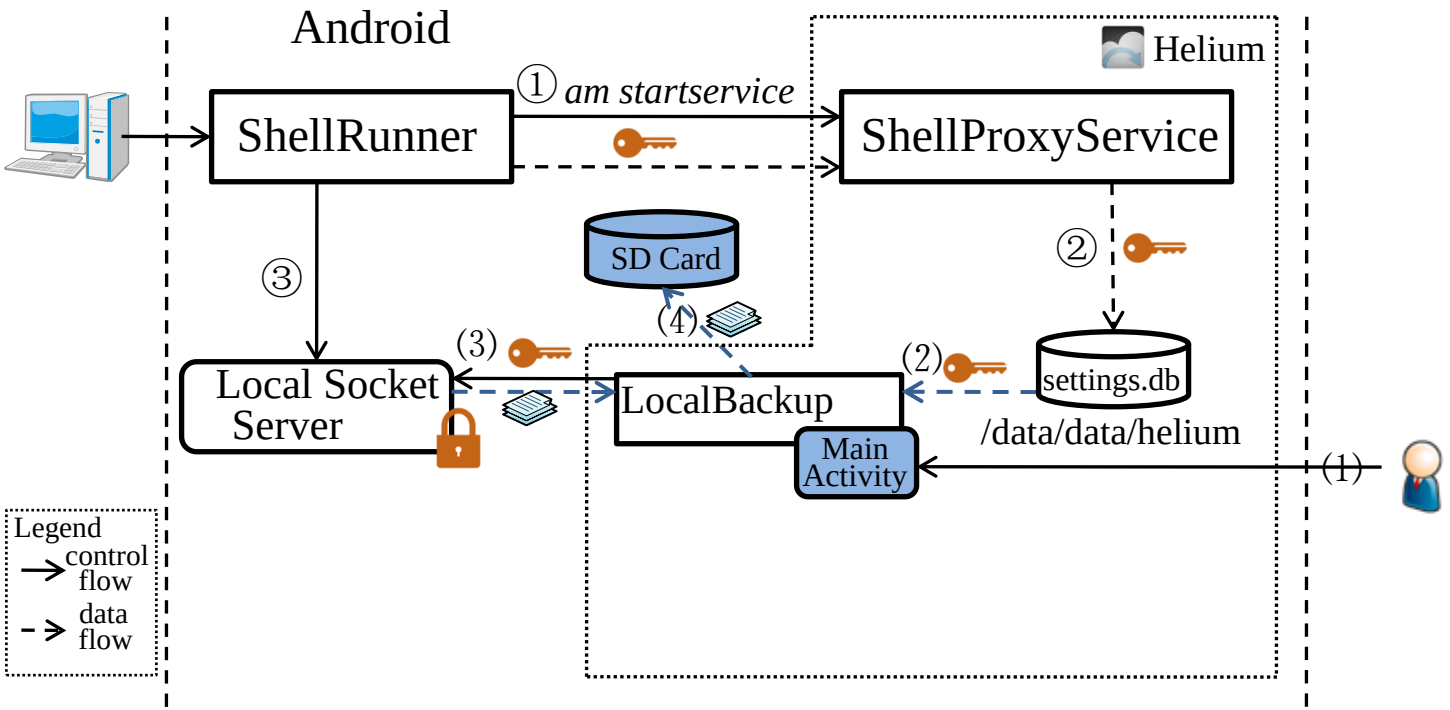
Analyzing an ADB-based Backup App

- Helium 
  - One of the best apps in 2013 (www.gizmap.com/best-android-apps-2013/30238)
  - Developer: ClockworkMod
    - Developer of CyanogenMod Android OS
    - Has released 19 apps on Google Play, 15 million installs

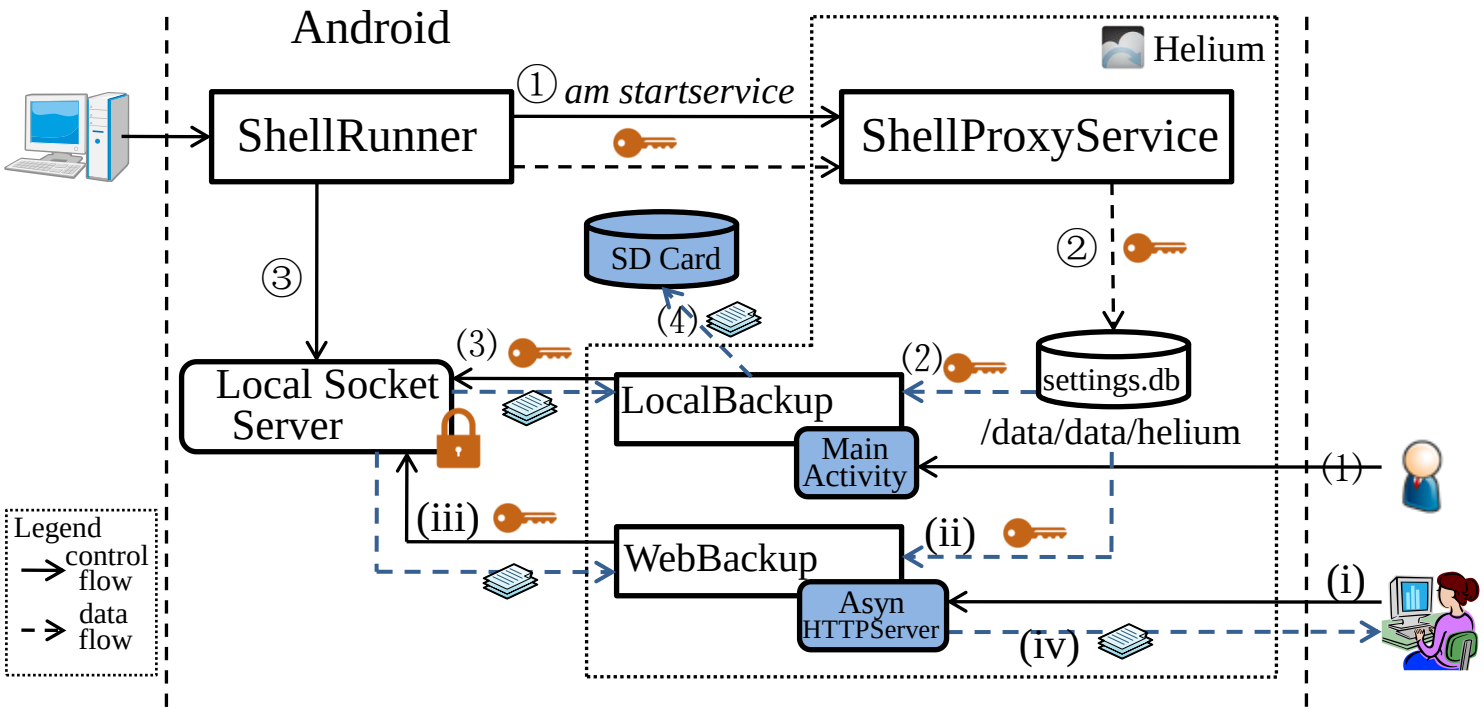
Internals of Helium (obtained by reverse engineering)



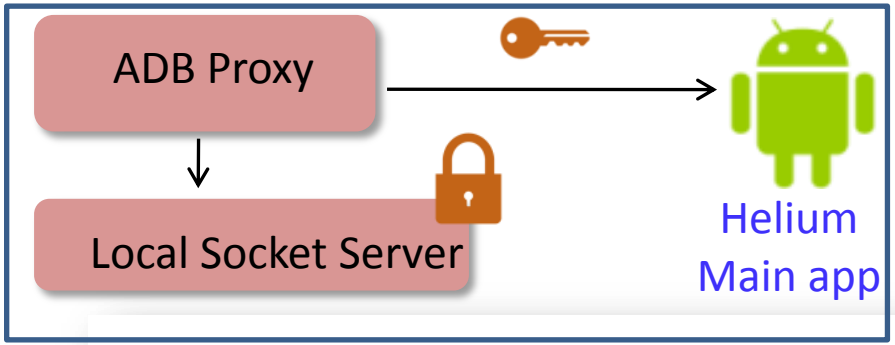
Internals of Helium (obtained by reverse engineering)

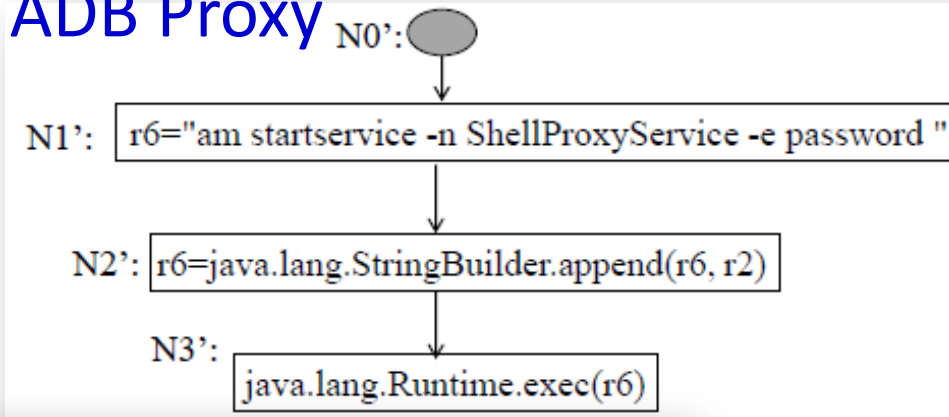


Internals of Helium (obtained by reverse engineering)

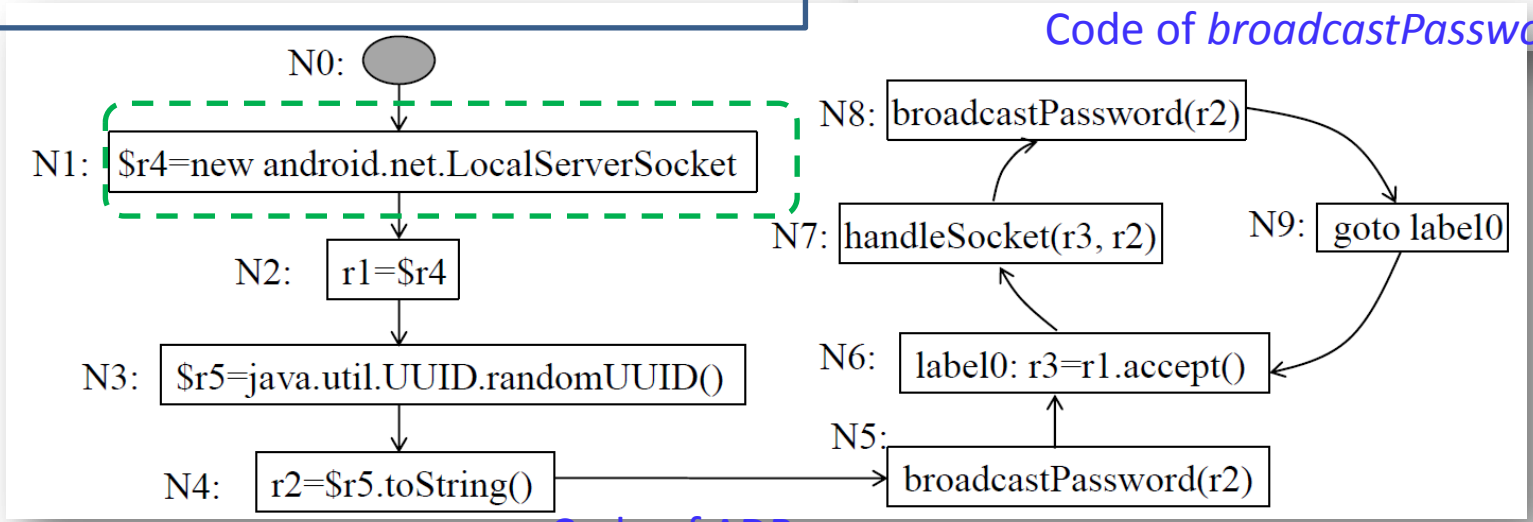


Access Control Protocol in the ADB Proxy



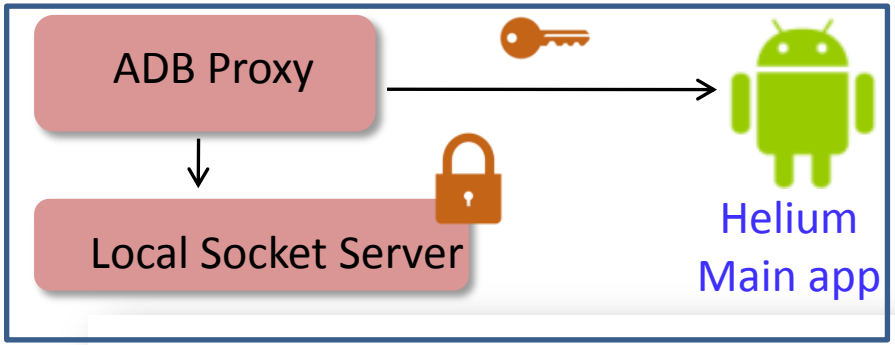


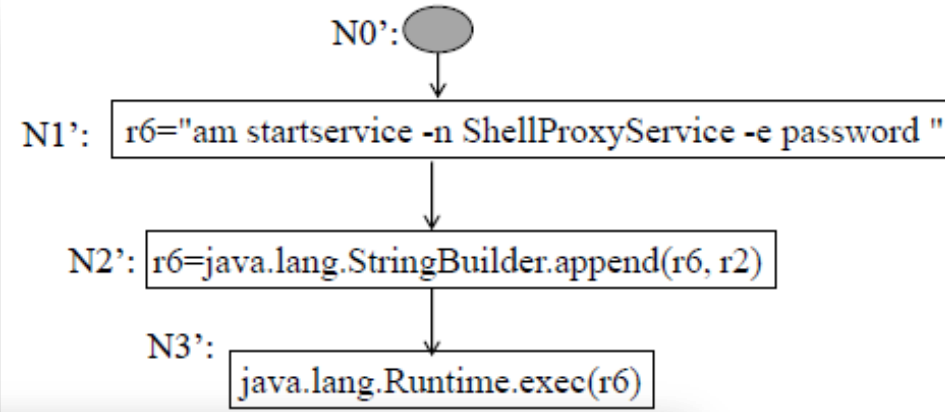
Code of *broadcastPassword()*



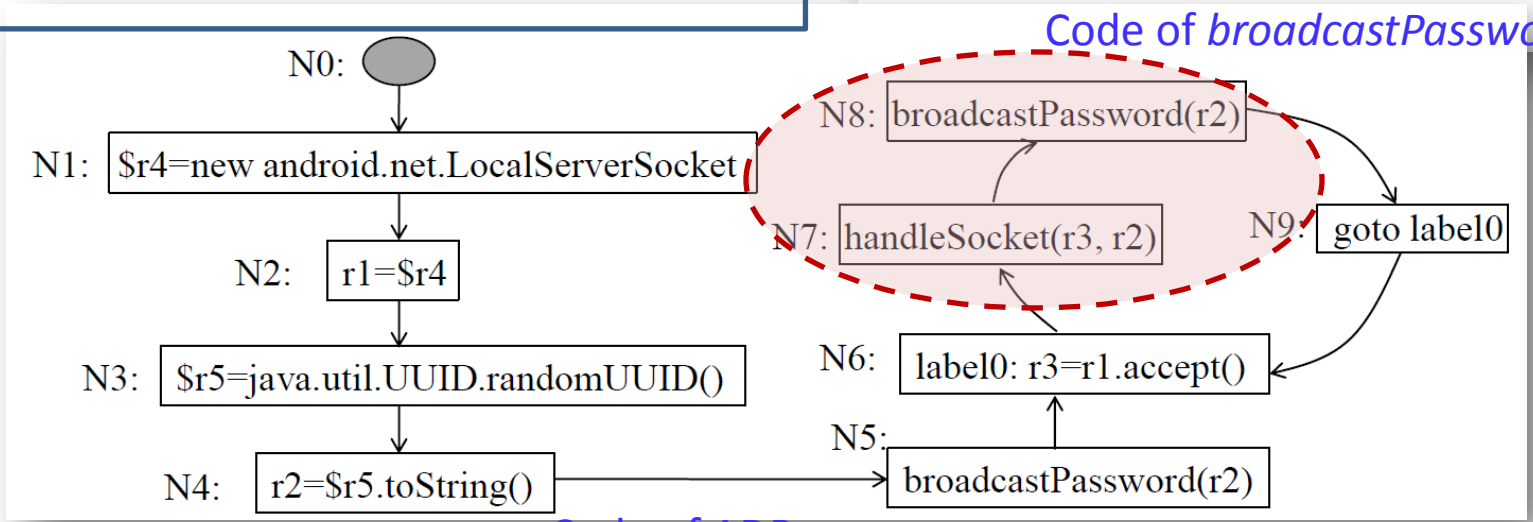
Code of ADB proxy

A logic flaw





Code of *broadcastPassword()*

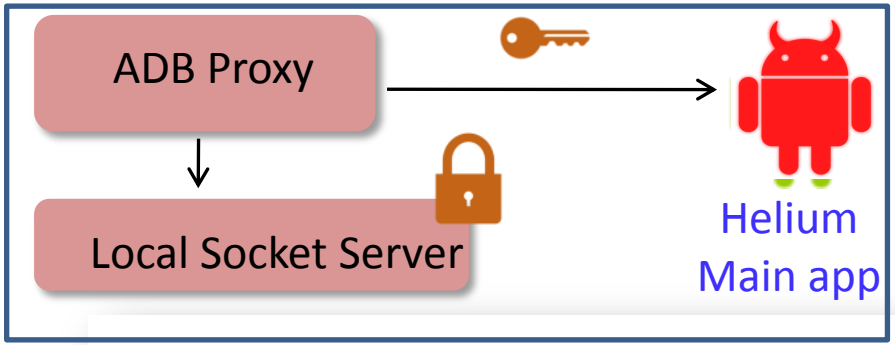


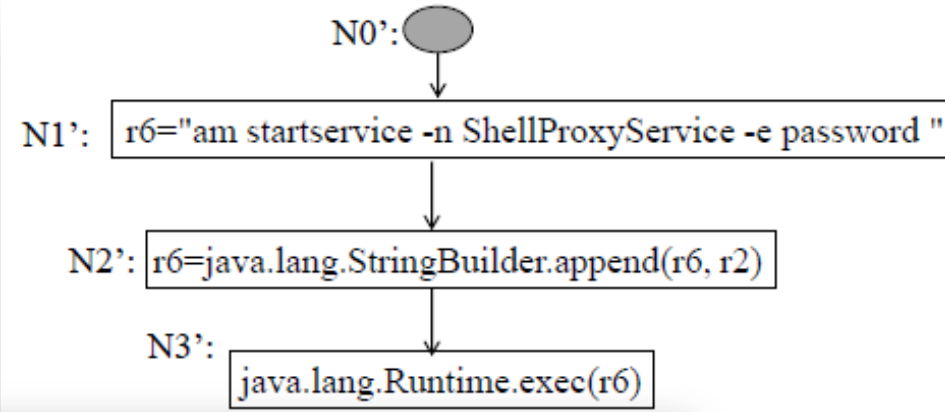
Code of ADB proxy

How *handleSocket()* works?

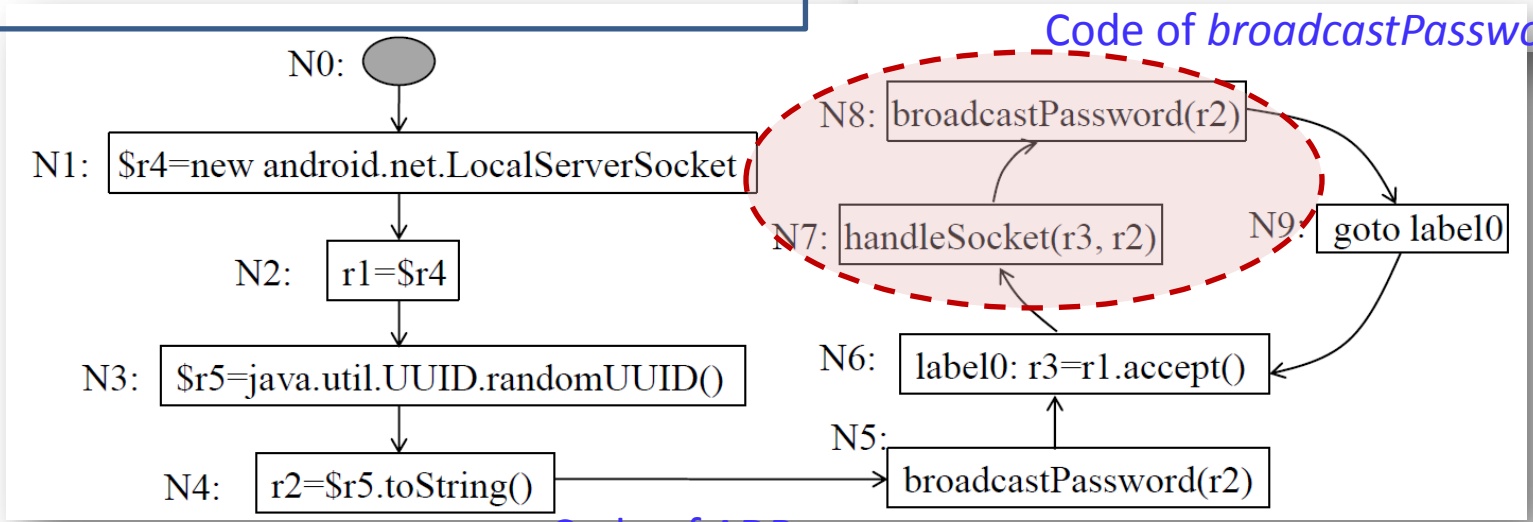
```
handleSocket()
{
    try {
        while(true) {
            r = getRequest();
            if (checkOTP(r))
                serve(r);
            else
                throw exception;
        }
    catch {
            // not terminate
    }
}
```

A logic flaw



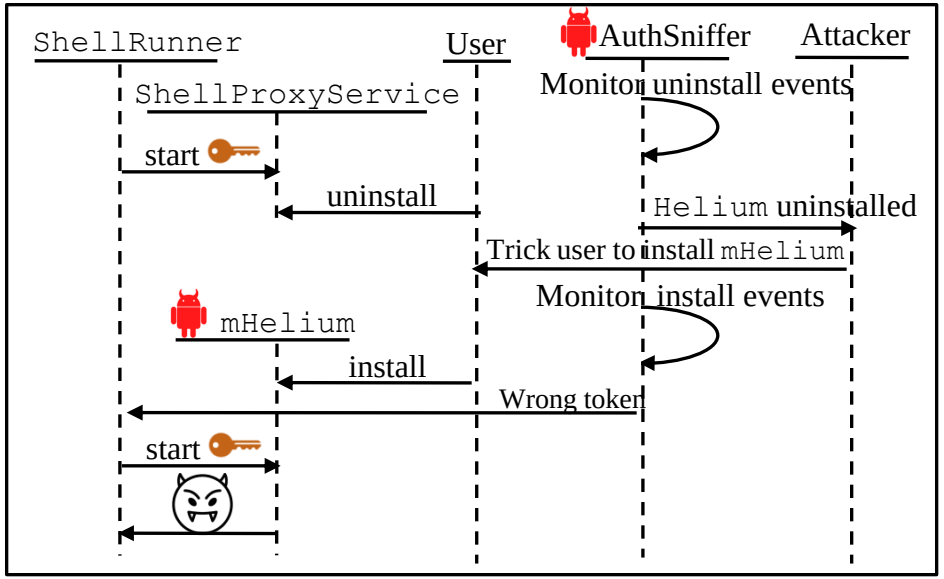


Code of *broadcastPassword()*



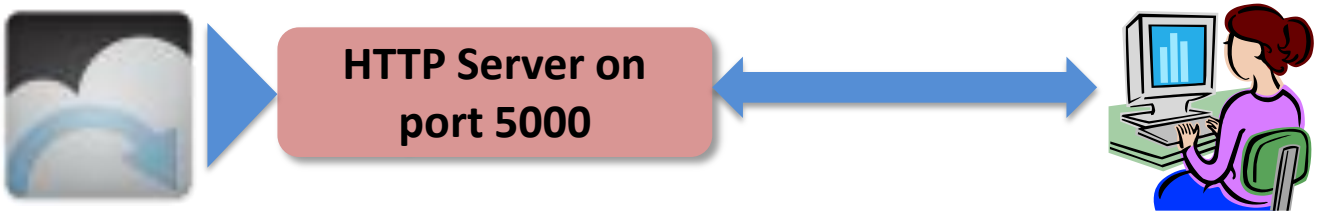
Code of ADB proxy

Attack #1: Exploit the logic flaw



- Disadvantage of the attacker
  - Helium needs to be uninstalled
  - Attacker needs to install an malware with the same name as Helium
- Advantage of the attacker
  - Once obtaining the OTP, the attacker is able to backup the victim app at any time (active attack)
  - Once obtaining the OTP, the attacker is able to conduct other high-privileged actions (see http://developer.android.com/tools/help/adb.html)

Attack #2: Invoke the Web interface



URL	Method	HTTP Body	Description
http://IP:5000/api/package	GET	NULL	Fetch the list of installed apps
http://IP:5000/api/backup.zip	POST	Name of the app to backup	Backup
http://IP:5000/api/restore.zip	POST	Backup data	Restore

Attack #2: Invoke the Web interface

- Disadvantage of the attacker
  - The HTTP server is closed by default and only open when web backup is used (semi-active attack)
  - Needs INTERNET permission
- Advantage of the attacker
  - Can backup target victim
  - Easier to implement than Attack #1

Attack #3: Access backup data on external storage

- Disadvantage of the attacker
  - Cannot chose target victim (passive attack)
- Advantage of the attacker
  - Easy to implement

Section 3. Impact and Case studies

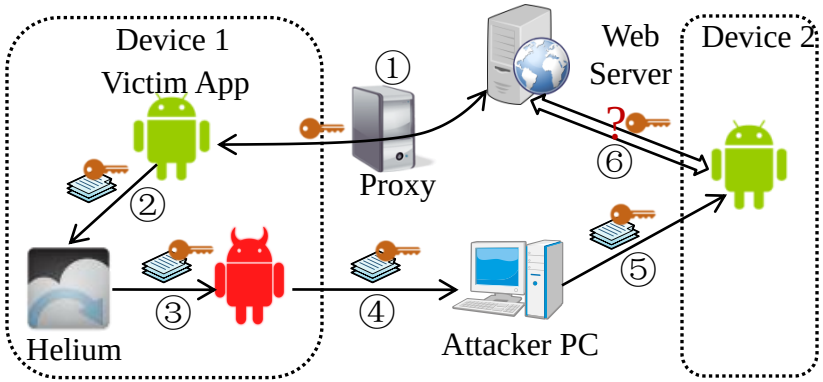
Extent of the ADB backup

- The apps won't be backup by ADB proxy when
  - Using Android Account Manager for authentication
  - *Android:allowBackup* is *false*

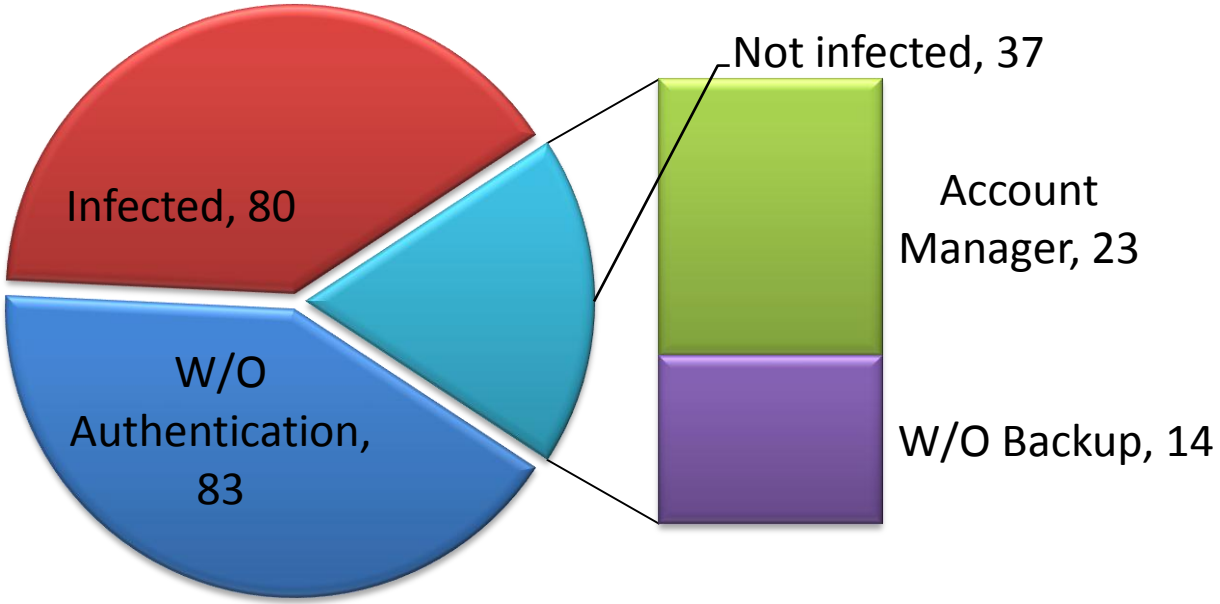
- If a developer does not specify it in *AndroidManifest.xml*, it is true by default!!
  - Our study reveals that only ~10% apps specify it false.

How many apps are subject to these attacks?

- Data Set I
  - Top ranked 100 apps
- Data Set II
  - Randomly chosen 10 Categories of apps from Goolge Play
  - Top 10 apps from each category

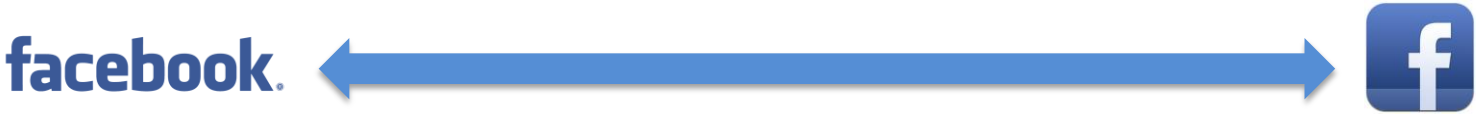


How many apps are subject to these attacks?



Case study #1: Facebook App

POST https://b-api.facebook.com/method/auth.login HTTP/1.1
...
User-Agent: [FBAN/FB4A;FBAV/9.0.0.26.28;FBBV /2403143;FBDM/
email=alice.tester%40gmail.com&password=pwd&sig
=452aca050cdce967a699e96907696 2f0&...



HTTP/1.1 200 OK
...
Content-Type: application/json
{"session_key":"5_71T...411696",
"access_token":"CAAAAUaZA...XW8ZD",
"session_cookies":[{"name":"c_user","value":"100003708411696","expir
es":"Thu, 28 May 2015 10:11:48 GMT","domain":".facebook.com"},
{"name":"xs","value":"201:71TTJlPmwZwjXQ :2:1401271908:10025","expi
res":"Thu, 28 May 2015 10:11:48 GMT","domain":".facebook.com"},
...]
...}

Identifying authenticators

access_token

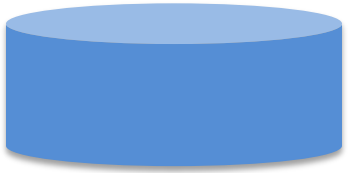
Credentials in subsequent requests, e.g., posting a new post

c_user

xs

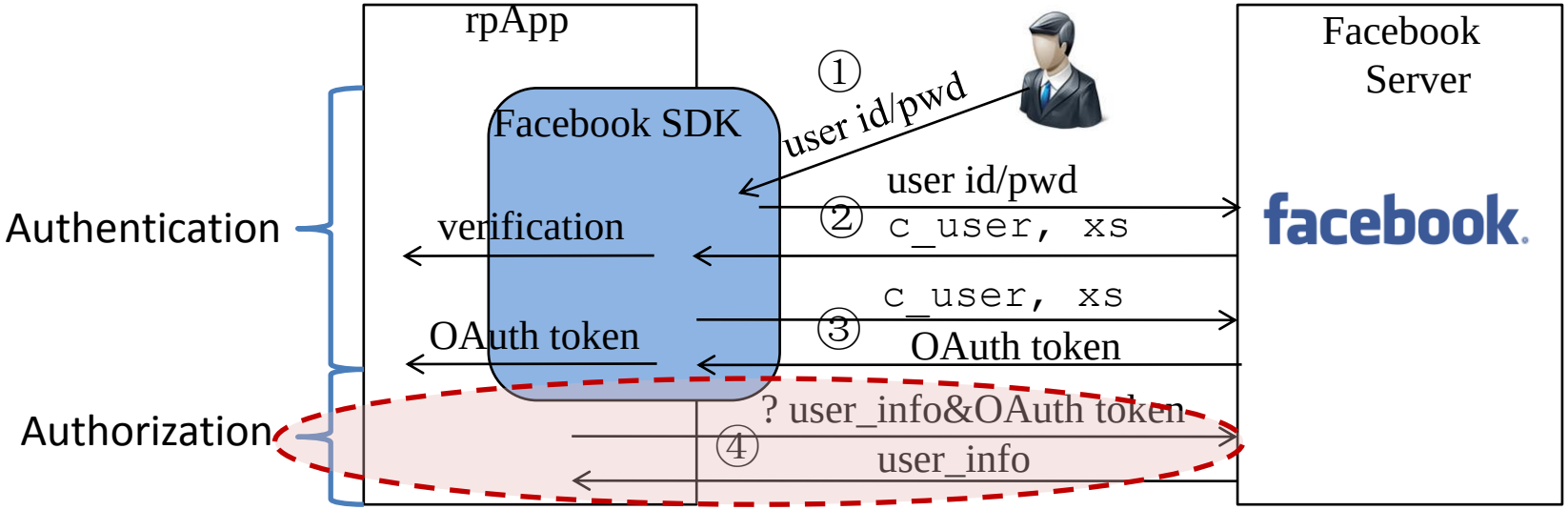
Credentials indicating the user's login state

prefs_db



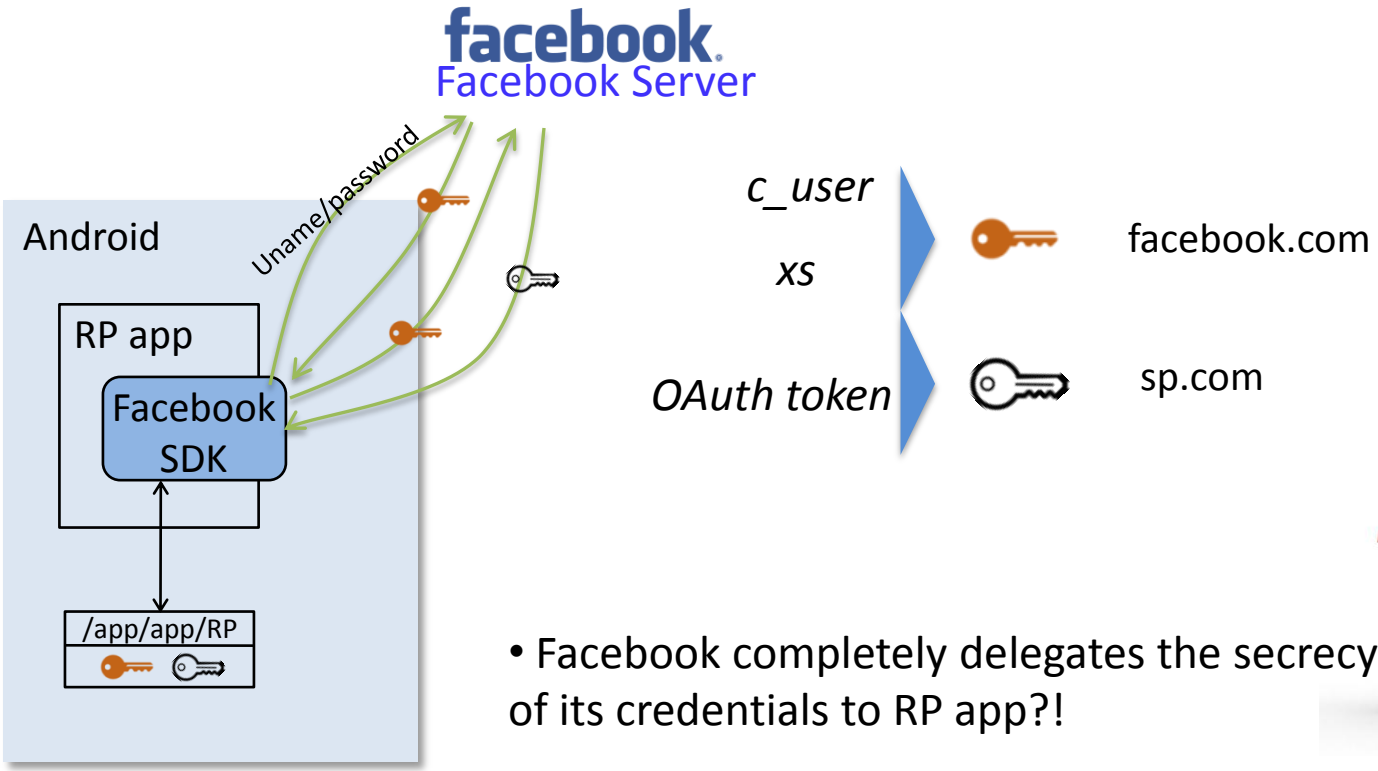
/data/data/com.facebook.katana

Case study #2: Facebook Single Sign-on



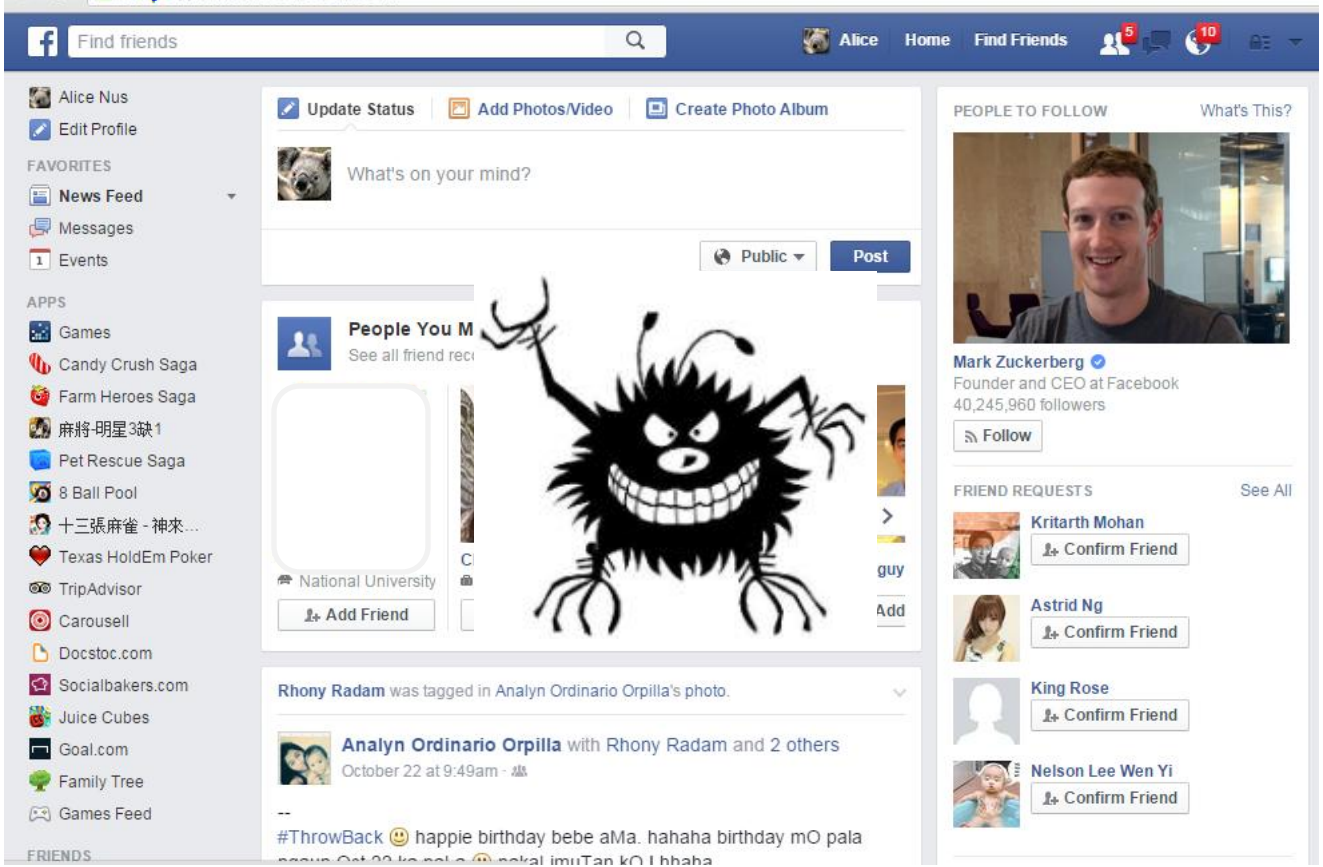
- Authorization: the user can control what information can be accessed by the rpApp.

Authenticators belonging to two origins?



- Facebook completely delegates the secrecy of its credentials to RP app?!

Using *c_user* and *xs* to log into user's account and completely violate authorization …



Facebook’s opinion

Facebook Security

But couldn't a malicious application with a WebView also steal usernames and passwords as they're submitted?
Once the user is entering their credentials outside of a trusted browser, there's very little that we can do from our end to protect them.
That's why it's so important that marketplaces like Google Play and Apple's App Store take steps to protect users from malicious applications.

Section 4. Mitigation

Suggestions to backup app developers

- Build secure ADB-based Backup
  - Prevent backup privilege from exposure
    - Verified Access control of the ADB proxy
    - Secrecy of backup data
  - Follow the principle of least privilege
    - Expose only backup/restore functionality
  - Manage lifecycle of ADB proxy
    - ADB proxy never outlives the main app

Suggestions to web app developers

- Protect authenticators
  - Disable *android:allowBackup* if not necessary
  - Avoid storing password
  - Shorten authenticator lifetime
- Avoid implementation own authenticator management
  - Use Android Account Manager

Summary and Take-away

- The dilemma
  - Backup functionality v.s. Confidentiality
  - Push the boundary or break the sandbox?

- Authentication
  - Awareness of infrastructure-level attacks

References

- [CCS’13] Wang, Rui, et al. "Unauthorized origin crossing on mobile platforms: Threats and mitigation."
- [CCS’14] Jin, Xing, et al. "Code injection attacks on HTML5-based mobile apps: Characterization, detection and mitigation."
- [ESORICS’15] Hassanshahi, Behnaz, et al. "Web-to-Application Injection Attacks on Android: Characterization and Detection."
- [IEEE S&P’12] Wang, Rui, et al. "Signing me onto your accounts through facebook and google: A traffic-guided security study of commercially deployed single-sign-on web services.“
- [NDSS’13] Bai, Guangdong, et al. “AUTHSCAN: Automatic Extraction of Web Authentication Protocols from Implementations.”

Thank you!

Bai Guangdong

baiguangdong@gmail.com