Insecure Internal Storage in Android

Claud Xiao

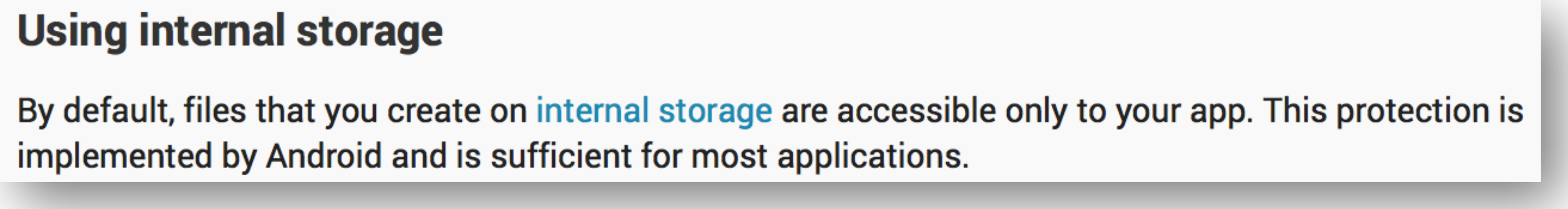
HITCON, Taipei

2014.08

paloalto networks.

It’s well known that in Android

- external storage is globally read/writable thus not secure;
- internal storage is isolated for each apps by sandbox thus is secure enough.
  - By Google’s suggestion, applications store sensitive data and configurations here.



Using internal storage

By default, files that you create on internal storage are accessible only to your app. This protection is implemented by Android and is sufficient for most applications.

http://developer.android.com/training/articles/security-tips.html#StoringData

Today, we're going to

- Present an **attack** to read/write data in internal storage
  - by combination of disclosed attacks and vulnerabilities.
- Explain why **94.2%** of popular apps are all vulnerable
- Disclose one category of apps storing password in plaintext
  - which are under the attack above,
  - affect **billions** of Android users,
  - and may lead to enterprise or server account leaking.
- Discuss some ideas of mitigation.

Attacks

ADB backup and restore

- Android Debug Bridge
- ADB backup
  - Fully backup almost all apps' internal data from device to PC.
  - Password to encrypt backup archive is optional but not enforced.
- ADB restore
  - Restore a backup archive to device.
  - Can modify data in the archive before restore it.

- More details on archive format:
  - http://nelenkov.blogspot.com/2012/06/unpacking-android-backups.html

Exceptions

- These apps won't be backup or restored:
  - whose "`android:allowBackup`" is `false` in AndroidManifest.xml
  - who implemented a `BackupAgent` by themselves.

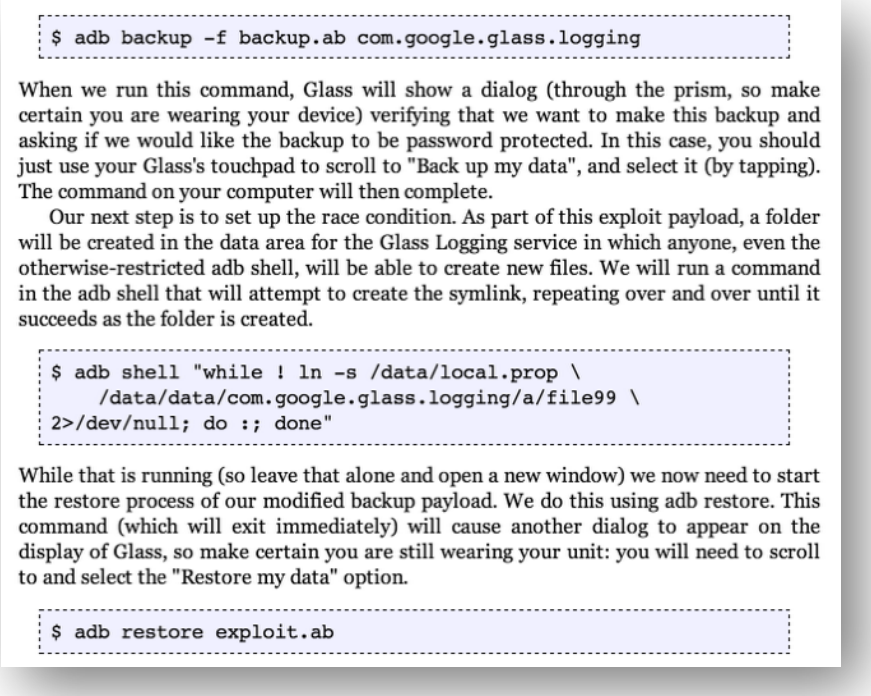
- When developers not set "`android:allowBackup`" manually, its value will be `true` by default!

- How many apps can be backup? Will be discussed later.

It’s a known “attack surface”

- Used to root Android devices like
  - some phone/tablet models (on XDA Developers)
  - and even Google Glass

- But these methods are NOT designed for real attacks
  - need user interactions
  - only for rooting your own devices



```
$ adb backup -f backup.ab com.google.glass.logging
```

When we run this command, Glass will show a dialog (through the prism, so make certain you are wearing your device) verifying that we want to make this backup and asking if we would like the backup to be password protected. In this case, you should just use your Glass's touchpad to scroll to "Back up my data", and select it (by tapping). The command on your computer will then complete.

Our next step is to set up the race condition. As part of this exploit payload, a folder will be created in the data area for the Glass Logging service in which anyone, even the otherwise-restricted adb shell, will be able to create new files. We will run a command in the adb shell that will attempt to create the symlink, repeating over and over until it succeeds as the folder is created.

```
$ adb shell "while ! ln -s /data/local.prop \
    /data/data/com.google.glass.logging/a/file99 \
2>/dev/null; do :; done"
```

While that is running (so leave that alone and open a new window) we now need to start the restore process of our modified backup payload. We do this using adb restore. This command (which will exit immediately) will cause another dialog to appear on the display of Glass, so make certain you are still wearing your unit: you will need to scroll to and select the "Restore my data" option.

```
$ adb restore exploit.ab
```

http://www.saurik.com/id/16

Restrictions of abusing ADB backup/restore

1. Connect to target device through an USB cable.
2. The system supports ADB backup/restore.
3. ADB debugging is enabled.
4. The device's screen is unlocked.
5. The PC can pass ADB authentication.
6. Click "Back up my data" button in ADB backup interface.

Let's "bypass" them all ☺

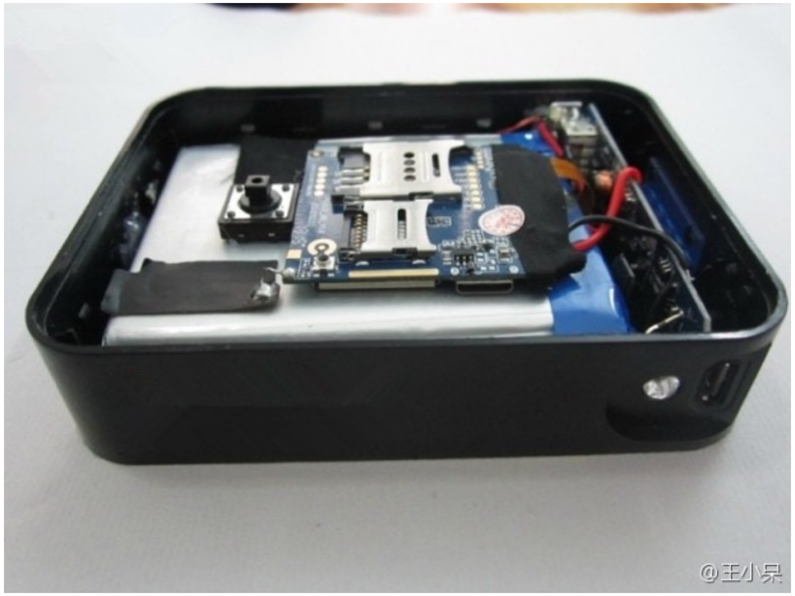
Connect to the device

- *Bridge-way:* use victim's PC as a bridge/proxy
  - Suppose attacker has controlled victim's PC by malware or phishing and plans to attack remotely.
  - Need to automate all further steps.
- *Direct-way*: directly attack victim's Android device
  - Suppose attacker can physically touch the target device temporarily.
  - Thus allow his interactions with device in further steps.

To find a bridge/proxy is not hard

- Cross infection between PC and mobile devices
  - Mobile -> PC:`USBCleaver`, `Ssucl`, …
  - PC -> Mobile: `Zitmo`, `Droidpak`, `WinSpy/GimmeRat`, …

- PC isn't the only bridge
  - May 2014, a customer bought a portable charger from *Taobao*, which was then found to be a customized remote control spy box with SIM card embedded.
  - Just like a real version of *Mactans* presented in Black Hat 2013
  - Or the "Juice-Jacking" attack



http://weibo.com/1705901331/B2AP6ihs2

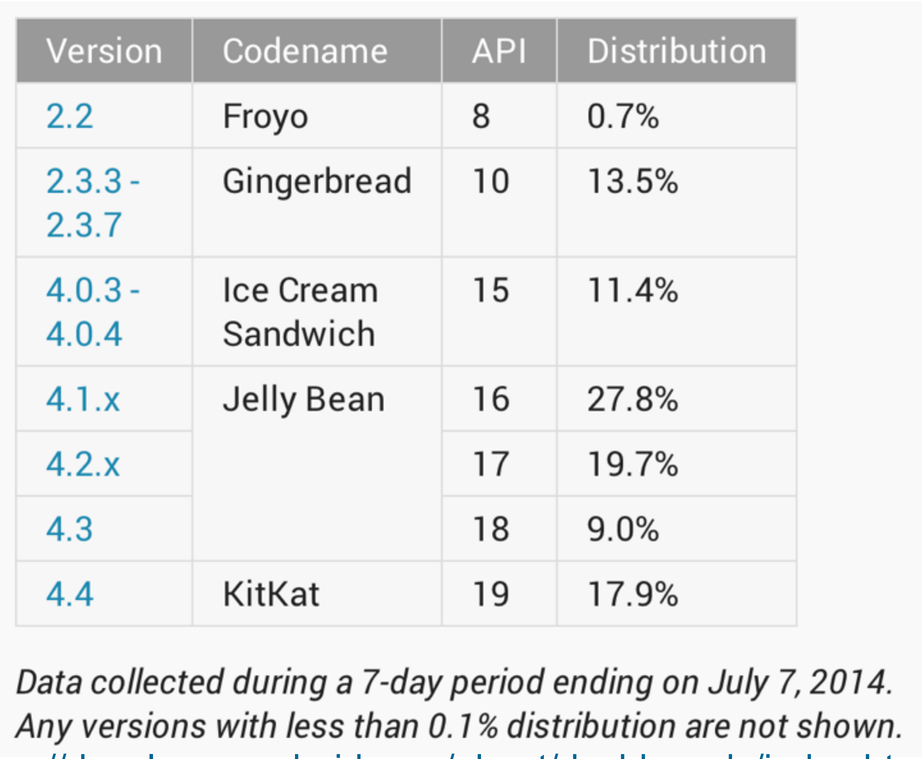
To physically touch a device is also not hard

- Intentionally (target someone)
  - steal it
  - temporarily borrow it
  - or even buy it from victim’s family

- Unintentionally
  - buy second-hand devices from resellers
  - find a lost phone
  - touch some public Android embedded devices

System’s support to ADB backup/restore

- Introduced in Android ICS 4.0.
- ~85.8% devices support it (Jul 7, 2014).
  - Google announced there’re over 1 billion 30-day active users on the Android platform at Jun 2014.



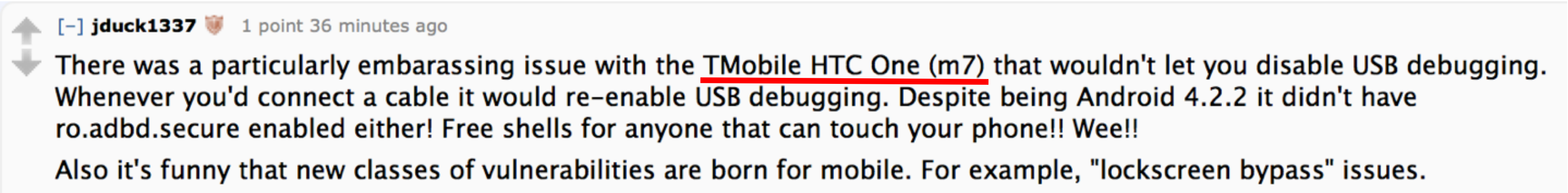
Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	13.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	11.4%
4.1.x	Jelly Bean	16	27.8%
4.2.x		17	19.7%
4.3		18	9.0%
4.4	KitKat	19	17.9%

Data collected during a 7-day period ending on July 7, 2014.
Any versions with less than 0.1% distribution are not shown.

https://developer.android.com/about/dashboards/index.html

Enable the ADB debugging

- Some enthusiasts have enabled it.
- Most of PC auxiliary tools ask and guide users to enable it.
- Some vendors even enable it by default.



[–] jduck1337 1 point 36 minutes ago

There was a particularly embarassing issue with the TMobile HTC One (m7) that wouldn't let you disable USB debugging. Whenever you'd connect a cable it would re-enable USB debugging. Despite being Android 4.2.2 it didn't have ro.adbd.secure enabled either! Free shells for anyone that can touch your phone!! Wee!!

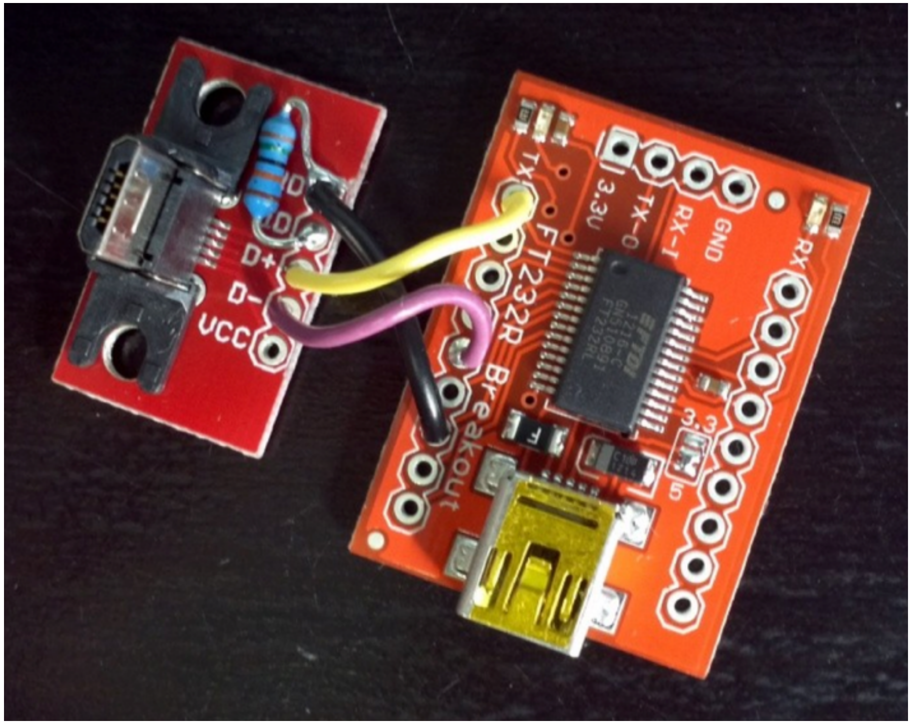
Also it's funny that new classes of vulnerabilities are born for mobile. For example, "lockscreen bypass" issues.

http://www.reddit.com/r/netsec/comments/27zdxc/android_hackers_handbook_ama

- When using utilities like `adbWireless`, even a normal app can use ADB debugging locally
  - Interesting. Apps may bypass sandbox in this way.

Enable the ADB debugging (cont.)

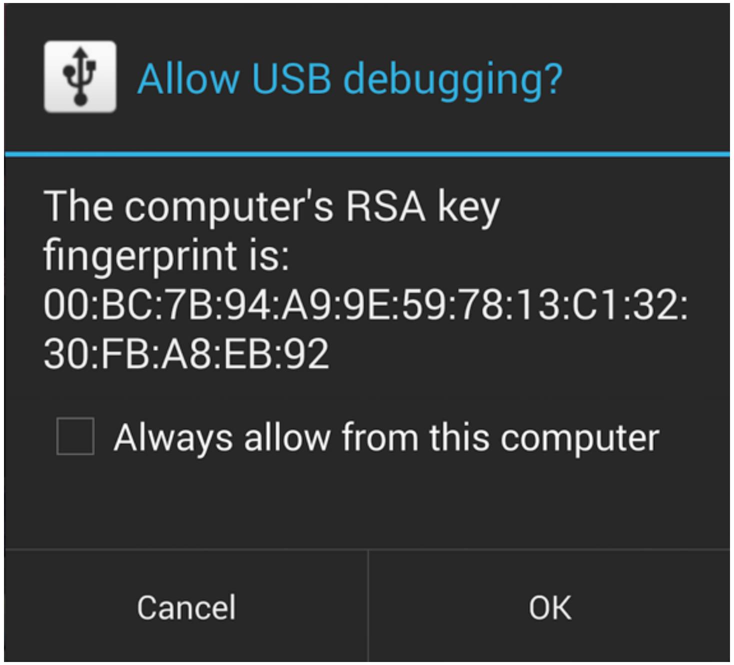
- For the rest devices, we can still try to enable it by an *USB multiplexer*
  - in just **tens of dollars** for hardware cost
  - refers Kyle Osborn and Michael Ossman's *Two Timing Data Connectors* in Infiltrate 2013



http://greatscottgadgets.com/infiltrate2013/infiltrate-osborn-ossmann.pdf

ADB authentication

- Introduced in Android 4.2.2



Allow USB debugging?

The computer's RSA key fingerprint is: 00:BC:7B:94:A9:9E:59:78:13:C1:32:30:FB:A8:EB:92

□ Always allow from this computer

- For preventing unauthorized devices (e.g., PC, portable recharger) connect to Android in ADB mode.

Bypass ADB authentication

- In all devices which support ADB, 45.7% don't have ADB authentication thus not need to bypass. (till July 7, 2014)
- For the rest 54.3%,
  - in "bridge-way", we can suppose the victim's PC passed authentication before.
  - in "direct-way", if device screen is unlocked, we can manually approve it.
  - in "direct-way", if device screen is locked, we can use a new disclosed vulnerabilities to bypass it.
    - affect Android <= 4.4.2.
    - https://labs.mwrinfosecurity.com/advisories/2014/07/03/android-4-4-2-secure-usb-debugging-bypass/

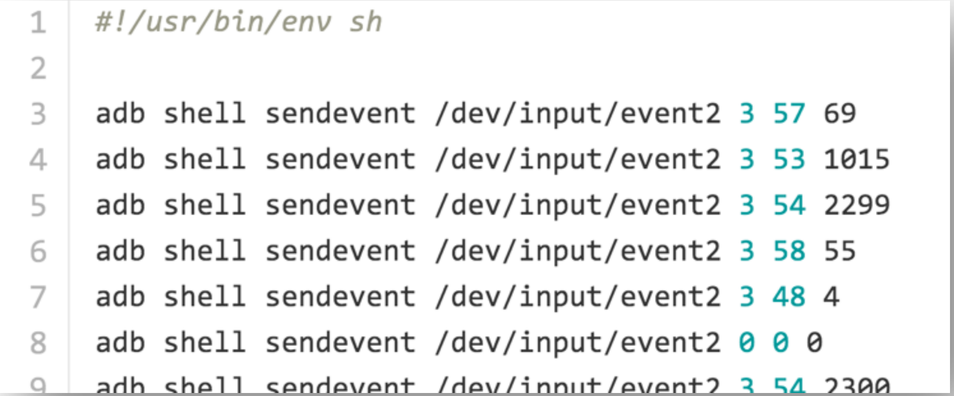
Unlock device’s screen lock

- Not all users use screen lock

- If it’s locked, since we can use ADB now:
    - Disable it by CVE 2013-6271
        - affect Android 4.0 - 4.3
        - http://seclists.org/fulldisclosure/2013/Nov/204
    - *(Optional)* use exists exploit to get root privilege, then disable it
        - like CVE-2014-3153 or Android bug #12504045
        - affect Android <= 4.4.4 (almost all Android devices)
        - Notice: root exploit isn’t essential for the whole attack

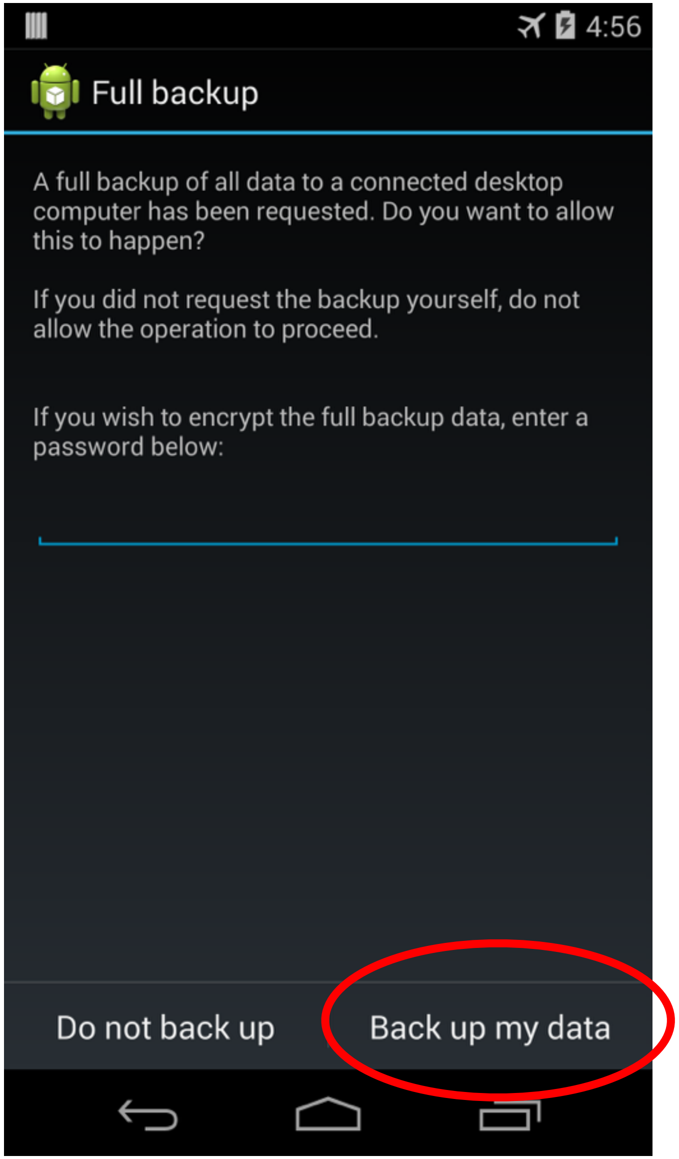
Click confirmation button

- In “direct-way”, just manually click it.
- In “bridge-way”, can simulate user’s click by `adb shell sendkey` to automate it in background:



```
#!/usr/bin/env sh

adb shell sendevent /dev/input/event2 3 57 69
adb shell sendevent /dev/input/event2 3 53 1015
adb shell sendevent /dev/input/event2 3 54 2299
adb shell sendevent /dev/input/event2 3 58 55
adb shell sendevent /dev/input/event2 3 48 4
adb shell sendevent /dev/input/event2 0 0 0
adb shell sendevent /dev/input/event2 3 54 2300
```



Full backup

A full backup of all data to a connected desktop computer has been requested. Do you want to allow this to happen?

If you did not request the backup yourself, do not allow the operation to proceed.

If you wish to encrypt the full backup data, enter a password below:

Conclusions

1. Highly possible to attack through USB cable via PC or touching device.
2. 85.8% of 1 billion devices support ADB backup.
3. In plenty of devices, ADB debugging has been enabled, or can be enabled by special hardware.
4. ADB authentication can be bypassed in almost all of them.
5. Screen lock can be bypassed in most of them.
6. User interaction can be performed automatically.

Jobs done ☺

DEMOs

Bridge-way

- Nexus 4
- Android 4.3
- ADB debugging is enabled
- The PC has been authenticated
- Screen is locked
- Totally automatically

Direct-way

- Nexus 4
- Android 4.4.2
- ADB debugging is enabled
- The PC has not been authenticated
- Screen is locked

Impact

If an app can be backup/restore

- attackers can read its internal sensitive data
  - e.g., password, tokens, etc
- or modify these sensitive data or configurations
  - e.g., login URL of banking
- Serious.

How many apps can be backup/restore?

- Analyzed 12,351 most popular apps from Google Play.
  - 556 of them explicitly set `android:allowBackup` to `false`.
  - 156 of the rest implement an `BackupAgent` to restrict backup.
  - The rest 11,639 apps can be fully backup/restore.

Statistics of Installations of Fully Backup-able Popular Apps in Google Play

installation counts	# of backup-able apps
500,000,000 - 1,000,000,000	4
100,000,000 - 500,000,000	35
50,000,000 - 100,000,000	38
10,000,000 - 50,000,000	524
5,000,000 - 10,000,000	766
1,000,000 - 5,000,000	5043
500,000 - 1,000,000	5229

Unimaginable Result

94.2%

of the most popular Android apps

are under threat of the attack.

What’s next?

- From this perspective, everyone can easily find tons of vulnerabilities in them.

- Here we just discuss the most serious case:
  Plaintext Storage of a Password

How to “store password”, or, remember account?

A balance between user experience and security.

1. Friendly but not secure:
   - store password in plaintext
   - store password in other reversible way, or obscure it
2. Secure but not friendly:
   - not store password at all
   - store password with symmetric-key algorithm, and request user to input passphrase every time

Using token as a replacement of password

- Pros
  - restrict resource access and control privilege more precisely
    - e.g., when login by token, users can't change password or security questions
  - expiration and renewing mechanism
- Cons
  - A non-restricted, full-privileged, non-expired token nearly equals to a password
    - e.g., Amazon's Android app
  - Can also be stolen by our attack

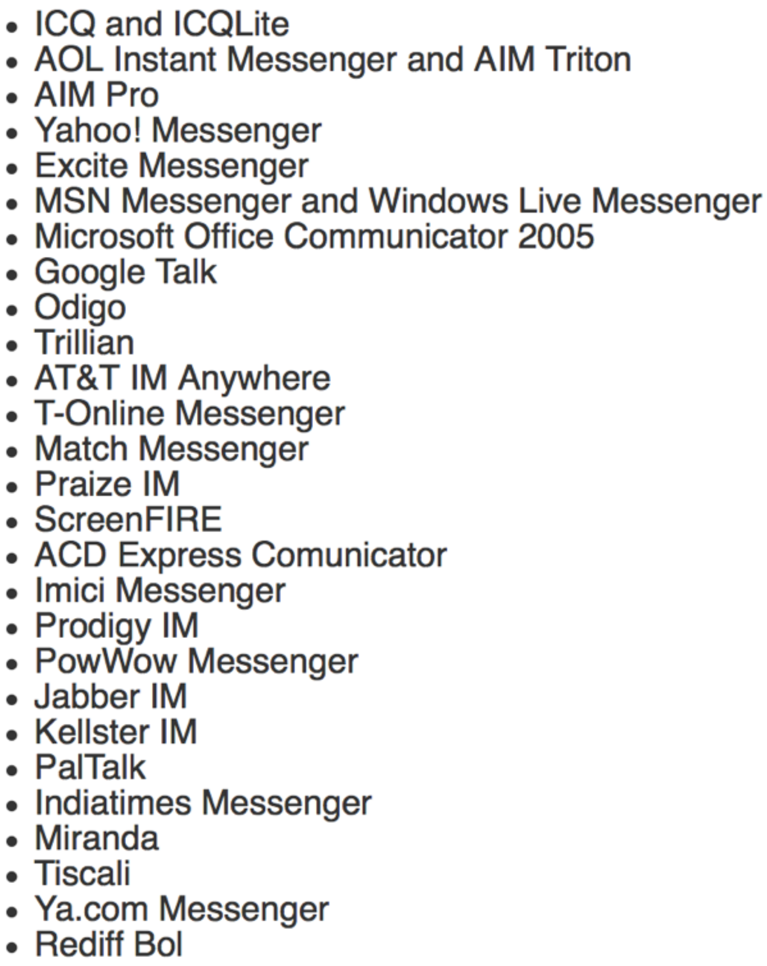
  ➢ Developers have to control and deploy servers to cooperate with their client applications

Token is not the silver-bullet

- For client applications of standard network services, developers can NOT expect which specific server users will connect to.
  - IMAP, SMTP, POP3
  - SSH, Telnet, FTP
  - IRC
  - HTTP
  - ……
- These apps must follow standard network protocols and their authentication methods.
  - Almost always password-based, e.g., IMAP-PLAIN and CRAM-MD5.

Common choice: user experience comes first

- Store or not store?
- In PC systems, the debate can be traced back to 1995 or earlier.
- Finally, most of popular IM clients, mail clients and browsers chose to store it.
  - Why?

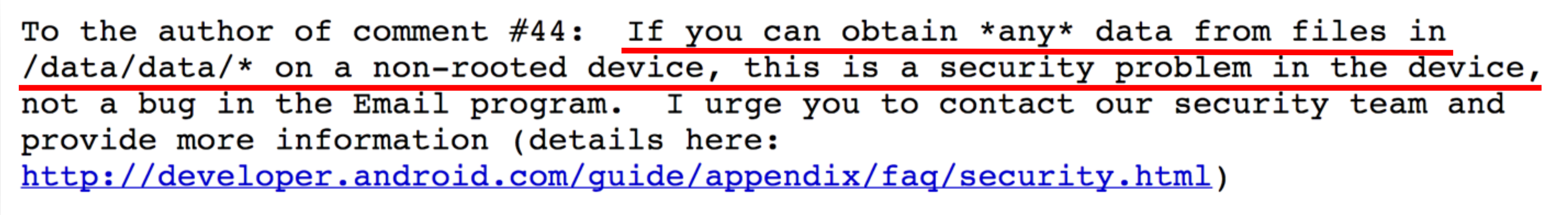


- ICQ and ICQLite
- AOL Instant Messenger and AIM Triton
- AIM Pro
- Yahoo! Messenger
- Excite Messenger
- MSN Messenger and Windows Live Messenger
- Microsoft Office Communicator 2005
- Google Talk
- Odigo
- Trillian
- AT&T IM Anywhere
- T-Online Messenger
- Match Messenger
- Praize IM
- ScreenFIRE
- ACD Express Comunicator
- Imici Messenger
- Prodigy IM
- PowWow Messenger
- Jabber IM
- Kellster IM
- PalTalk
- Indiatimes Messenger
- Miranda
- Tiscali
- Ya.com Messenger
- Rediff Bol

https://developer.pidgin.im/wiki/PlainTextPasswords

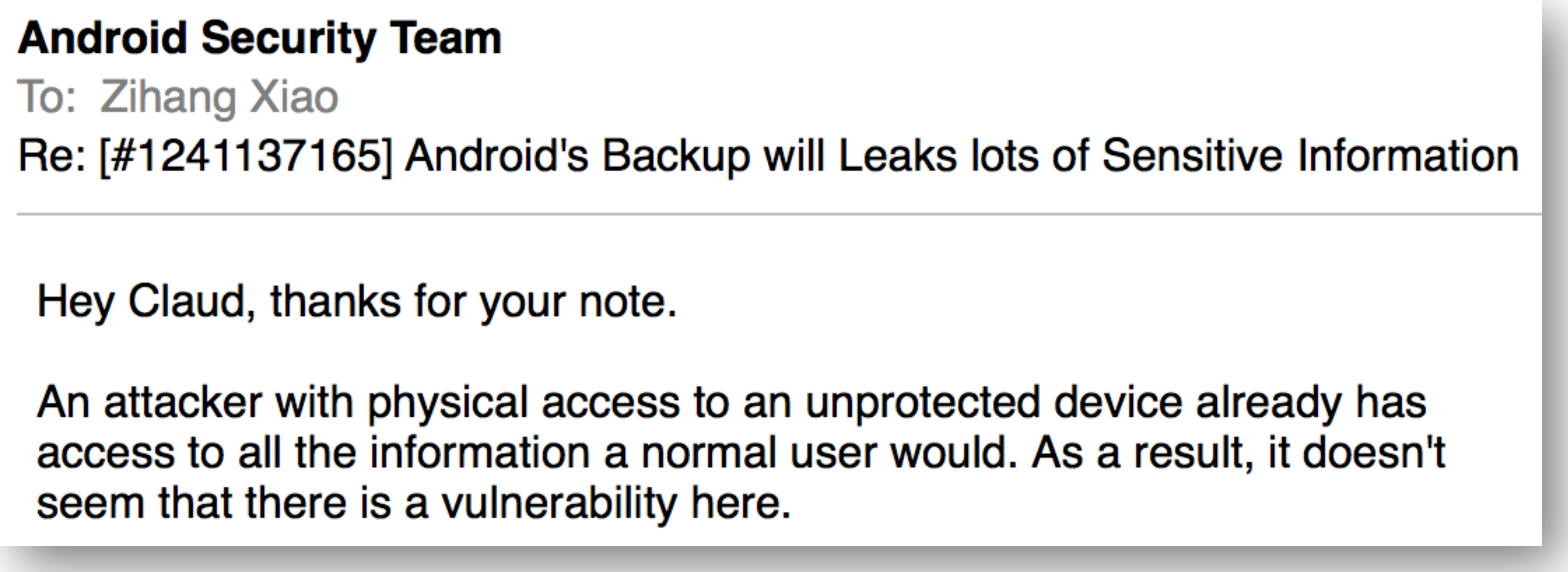
Security assumption

- Rely password storage's security on system's access control mechanisms.
  - Classic explanation: https://developer.pidgin.im/wiki/PlainTextPasswords
- Google's opinion https://code.google.com/p/android/issues/detail?id=10809
  tl;dr:
  1. They're old, insecure protocols;
  2. We trust the system's security.



To the author of comment #44: If you can obtain *any* data from files in /data/data/* on a non-rooted device, this is a security problem in the device, not a bug in the Email program. I urge you to contact our security team and provide more information (details here: http://developer.android.com/guide/appendix/faq/security.html)

So we reported to Google…



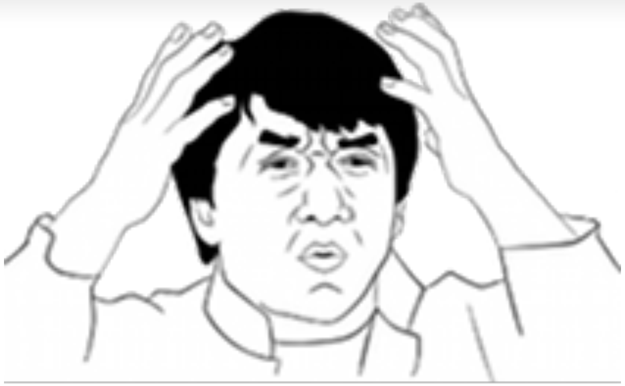
Android Security Team

To: Zihang Xiao

Re: [#1241137165] Android's Backup will Leaks lots of Sensitive Information

Hey Claud, thanks for your note.

An attacker with physical access to an unprotected device already has access to all the information a normal user would. As a result, it doesn't seem that there is a vulnerability here.



Android applications storing password in plaintext

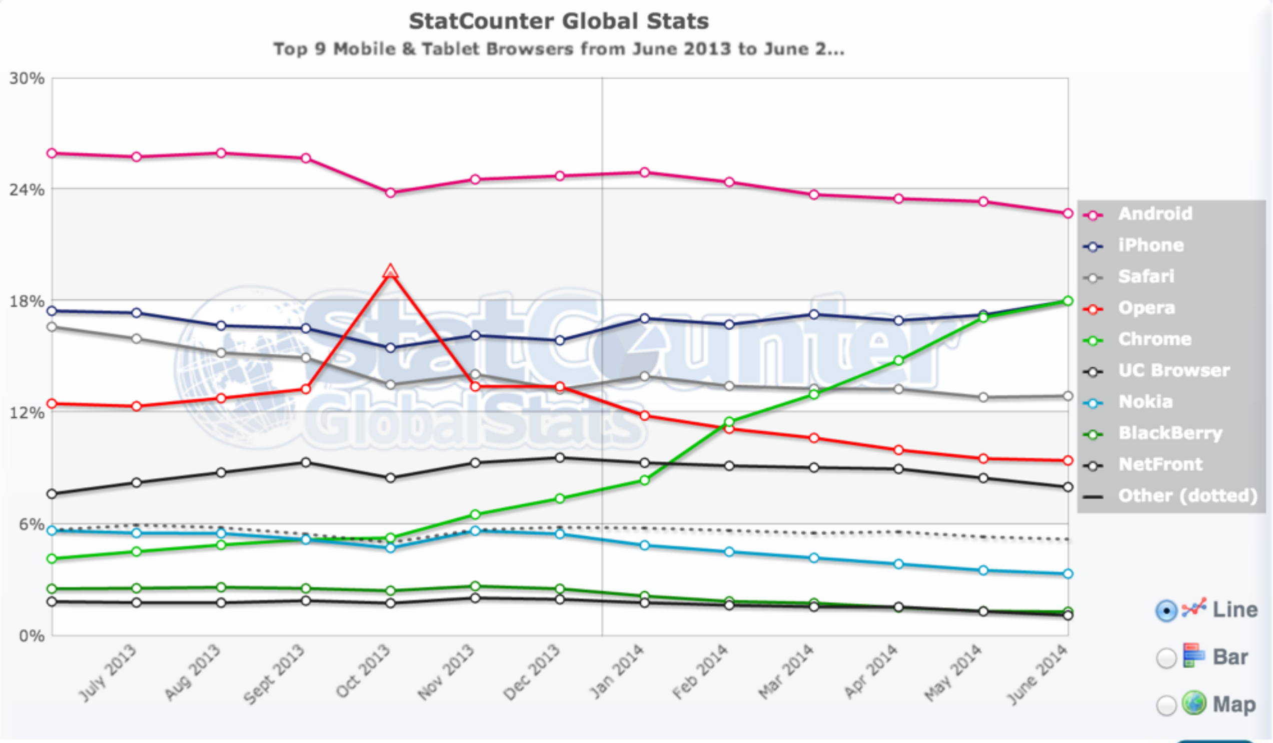
Application	Package Name	Installations
K9 Mail	`com.fsck.k9`	5,000,000 – 10,000,000
Blue Mail	`com.trtf.blue`	100,000 – 500,000
MailDroid	`com.maildroid`	1,000,000 – 5,000,000
myMail	`com.my.mail`	100,000 – 500,000
SSH Tunnel	`org.sshtunnel`	100,000 – 500,000
Unix Admin: FTP SFTP SSH FTPS	`org.kidinov.unixadmin`	10,000 – 50,000
SSH Autotunnel	`cz.sde.tunnel`	10,000 – 50,000
BotSync SSH SFTP	`com.botsync`	10,000 – 50,000
AndFTP (your FTP client)	`ysesoft.andftp`	1,000,000 – 5,000,000
FtpCafe FTP Client	`com.ftpcafe.trial`	100,000 – 500,000

Pre-installed apps is also vulnerable

- Email (`com.android.email` and `com.google.android.email`)
  - Passwords of POP3, SMTP and IMAP accounts are stored in `EmailProvider.db` and `EmailProviderBackup.db`.
  - The Email application is pre-installed in almost every Android devices.
    - Again, there're 1 billion active Android users till Jun 2014.

Pre-installed apps is also vulnerable (cont.)

- Browser (`com.android.browser`)
  - Remembered passwords of websites are stored at `webview.db`.
  - 22.7% of all mobile phone users use it (Jun 2014, by *StatCounter*).
  - Some other third-party browsers have the same problem.



http://gs.statcounter.com/

Specially for Taiwanese users

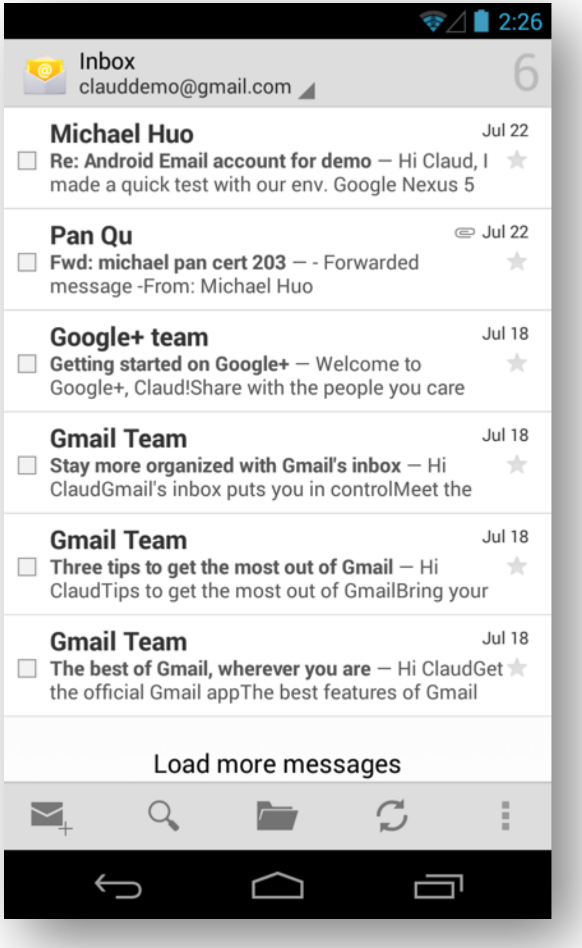
- PTT(批踢踢) is the most popular BBS in Taiwan
  - base on TELNET

Vulnerable PTT clients

Application	Package Name	Installations
Mo PTT	mong.moptt	500,000 – 1,000,000
Miu Ptt	sg.xingzhi.miu_ptt	100,000 – 500,000
touchPTT	com.yuandroid.touchPTT	100,000 – 500,000
JPTT	com.joshua.jptt	50,000 – 100,000
……		

DEMOs

- Android Email



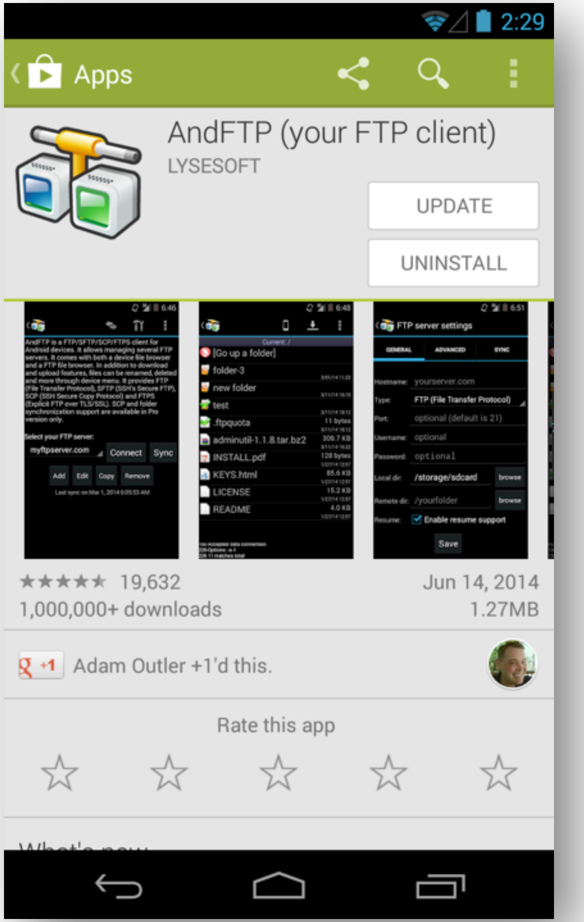
2:26

Inbox
clauddemo@gmail.com
6

□	**Michael Huo** **Re: Android Email account for demo** — Hi Claud, I made a quick test with our env. Google Nexus 5	Jul 22 ☆
□	**Pan Qu** **Fwd: michael pan cert 203** — - Forwarded message -From: Michael Huo	Jul 22 ☆
□	**Google+ team** **Getting started on Google+** — Welcome to Google+, Claud!Share with the people you care	Jul 18 ☆
□	**Gmail Team** **Stay more organized with Gmail's inbox** — Hi ClaudGmail's inbox puts you in controlMeet the	Jul 18 ☆
□	**Gmail Team** **Three tips to get the most out of Gmail** — Hi ClaudTips to get the most out of GmailBring your	Jul 18 ☆
□	**Gmail Team** **The best of Gmail, wherever you are** — Hi ClaudGet the official Gmail appThe best features of Gmail	Jul 18 ☆

Load more messages

- AndFTP



2:29

Apps

AndFTP (your FTP client)
LYSESOFT

UPDATE

UNINSTALL

6:46

AndFTP is a FTP/SFTP/SCP/FTPS client for Android devices. It allows managing several FTP servers. It comes with both a device file browser and a FTP file browser. In addition to download and upload features, files can be renamed, deleted and more through device menu. It provides FTP (File Transfer Protocol), SFTP (SSH's Secure FTP), SCP (SSH Secure Copy Protocol) and FTPS (Explicit FTP over TLS/SSL). SCP and folder synchronization support are available in Pro version only.

Select your FTP server:
myftpserver.com
Connect Sync
Add Edit Copy Remove
Last sync on Mar 1, 2014 6:05:53 AM

6:48

Current: /

[Go up a folder]	
folder-3	
new folder	
test	
.ftpquota	11 bytes
adminutil-1.1.8.tar.bz2	306.7 KB
INSTALL.pdf	128 bytes
KEYS.html	85.6 KB
LICENSE	15.2 KB
README	4.0 KB

6:51

FTP server settings

GENERAL ADVANCED SYNC

Hostname:	yourserver.com
Type:	FTP (File Transfer Protocol)
Port:	optional (default is 21)
Username:	optional
Password:	optional
Local dir:	/storage/sdcard browse
Remote dir:	/yourfolder browse
Resume:	☑ Enable resume support

Save

★★★★★ 19,632
1,000,000+ downloads

Jun 14, 2014
1.27MB

+1 Adam Outler +1'd this.

Rate this app
☆ ☆ ☆ ☆ ☆

What's new

Conclusions

- 94.2% of popular Android apps can be backup thus are affected by the attack
- Network services clients and some pre-installed apps will store password in plaintext and become vulnerable by the attack.
- Affect almost all Android users, and may lead personal and enterprise's account leaking

Mitigations

Easiest mitigation ways

- From OS's perspective,
  - Set default value of `R.attr.allowBackup` to `false`.
    - Just a description in document is not enough. Most of developers ignored it.

- From Developer's perspective,
  - In AndroidManifest.xml, set `android:allowBackup` to `false`.
  - Or implement a `BackupAgent` to specify what data to be backup.

Easiest mitigation ways (cont.)

- From user's perspective,
  - Disable ADB debugging when not need it
    - still not secure enough
  - Avoid to lost it
    - how?
  - Update to the newest system
    - Android L may be enough
  - Reset useless phone to factory
    - still vulnerable
  - Encrypt the whole disk.

Third-party hot patching

- Need root privilege at first
  - an dilemma
- Discussed before by Collin Mulliner et al at ACSAC '13
  - *PatchDroid: scalable third-party security patches for Android devices*
- Develop with existing dynamic instrumentation frameworks
  - `Xposed`: http://repo.xposed.info/
  - `adbi/ddi`: http://www.mulliner.org/android/
  - `Cydia Substrate` for Android: http://www.cydiasubstrate.com/
  - There're lots of great security enhancement apps based on them.

Some simple patching ideas

- Disable ADB debugging when screen is locking
  - Just like what FirefoxOS did.
- Fix the CVE-2013-6271 for Android <= 4.3
- Fix the CVE-2014-3153 for Android <= 4.4.4
- Disable "`adb shell sendkey`" when doing backup
- Transparently encrypt all internal data by a master passphrase

DEMO

Summary

- An attack to read or modify internal data in high success rate.
- 94.2% of popular applications are influenced.
- Network services clients and pre-install apps store password in plaintext internally.
- Easy to mitigate the problem from different perspectives.

Thank you!

- Claud Xiao, Senior Security Researcher at Palo Alto Networks
- @claud_xiao, iClaudXiao@gmail.com
- DEMO code: http://github.com/secmobi/BackupDroid

Special thanks to:

Elad Wexler, Zhi Xu, Ryan Olson, Bo Qu, visualwu, irene, tombkeeper

Greets:

Nikolay Elenkov, Collin Mulliner, Kyle Osborn, Michael Ossman,
MWR Labs, rovo89, jduck, Jay Freeman(saurik)