



평양 2407

Hacking North Korea's Android

1

[ @hackerfantastic ] [ @myhackerhouse ] [ <https://hacker.house> ]

```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161     host = gethostbyname(server);
162
163
164
```

# Hello my name is



## Introductions

2

```
exit(0);
```

[ @hackerfantastic ] [ @myhackerhouse ] [ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
165
166}
```

```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
```

# 평양 2407 Smart Phone

... or surveillance tool? We explore, you decide!



- 34C3 - KCC release “평양 2407” disk image
- Locating and obtaining a phone
- Hardware teardown & booting disk image
- Android ROM reverse engineering
- Security features, user privacy & exploits/jailbreaking

```
161     host = gethostbyname(server);
162
163
164
```

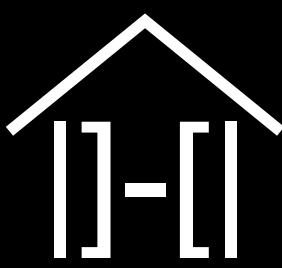
## 평양 2407 Overview

3

```
165     exit(0);
166 }
```

[ @hackerfantastic ] [ @myhackerhouse ] [ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
```



```
161     host = gethostbyname(server);
162
163
164
```

# North Korea smart phones

4

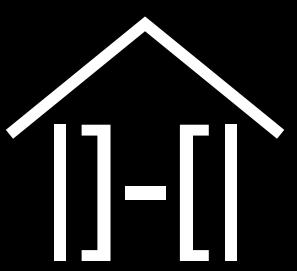
```
exit(0);
```

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
```



```
161     host = gethostbyname(server);
162
163
164
```

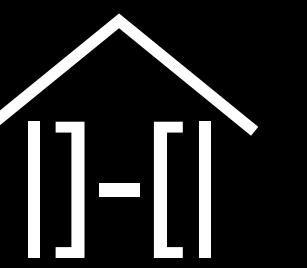
“MediaTek” made in China

5

```
165         exit(0);
166     }
```

[ @hackerfantastic ] [ @myhackerhouse ] [ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161     host = gethostbyname(server);
162
163
164
165
166 }
```

# 평양 2407 Internals

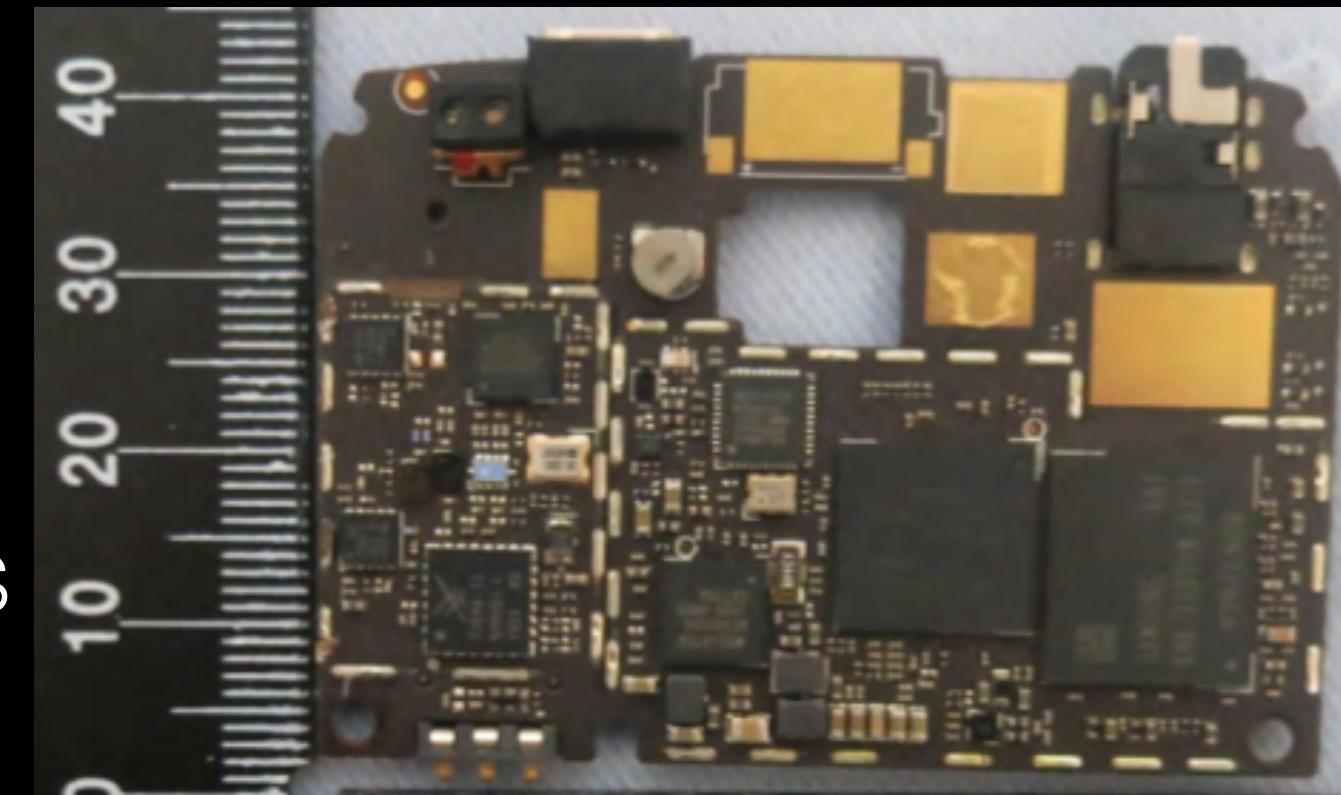
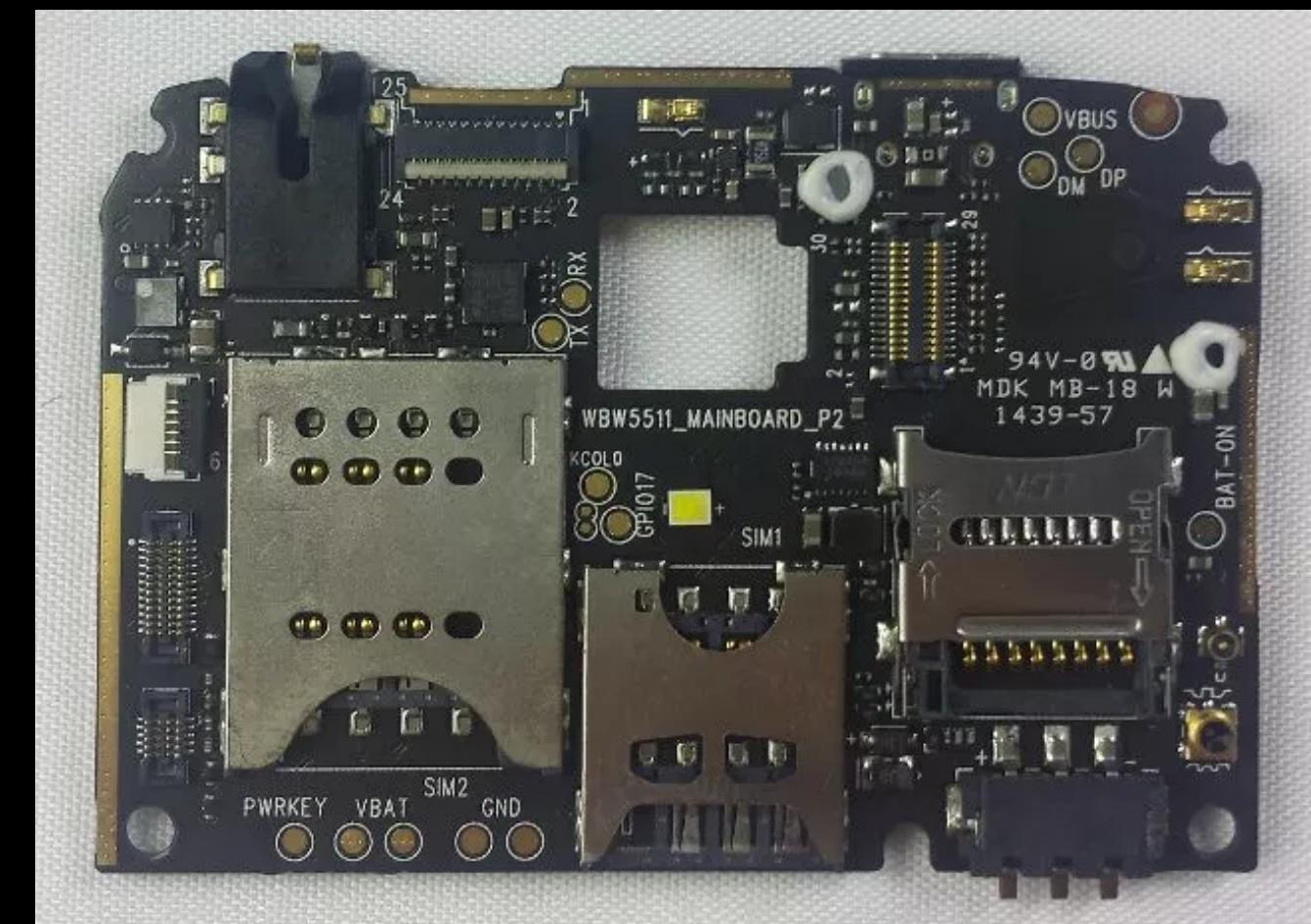
## WBW5511\_MAINBOARD\_P2

Mass-produced Mediatek device for overseas markets using Android, vendor deploys aftermarket Android with custom overlays & branding.

Several vendors supply identical WBW5511 based Android devices.

- **Gionee Ctrl V5 (India)**
- **Walton Primo H3 (Egypt / India)**
- **BLU Life Play 2 (USA)**

Exported to USA, certified in China for FCC <https://fccid.io/YHLBLULIFEPLAY2> - ROMS are compatible between hardware devices.



## 평양 2407 (MEDIATEK)

6

```
exit(0);
```

[ @hackerfantastic ]

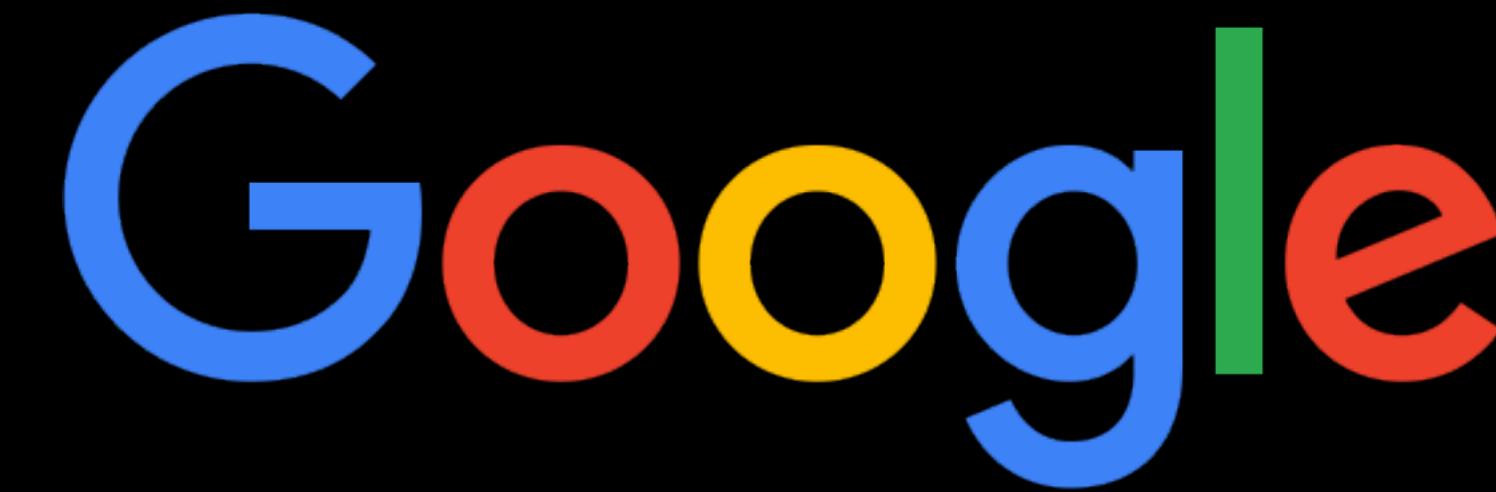
[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea

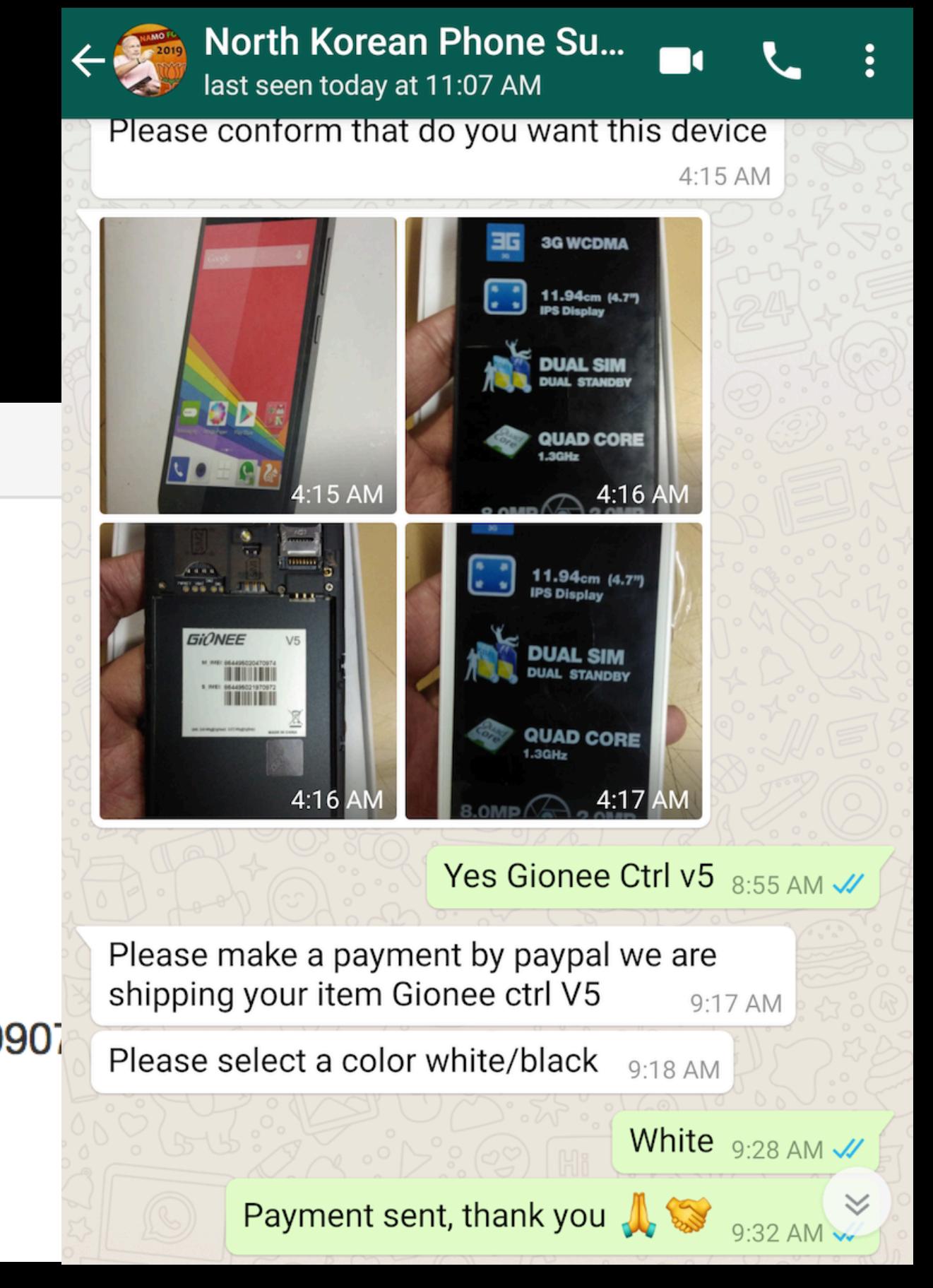


```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
```



### BLU Products, Inc. FCC Registration Number

FRN	0019890946
Registration Date:	06/02/2010 03:15:00 AM
Last Updated:	
Business Name:	CT Asia
Business Type:	Foreign Entity
Contact Organization:	
Contact Position:	Product Manager
Contact Name:	Mr Chris Chan
Contact Address:	Unit 01, 15/F, Seaview Centre, 139-141 Hoi Bun Road, Kwun Tong, Kowloon, Hong Kong 99907
Contact Email:	chris@ctasiahk.com
ContactPhone:	852-27931198
ContactFax:	852-27931197



```
host = gethostbyname(server);
```

# Where to buy phones?

7

```
exit(0);
```

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
```



```
161     host = gethostbyname(server);
162
163
164
```

Booting 평양 2407 ROM

8

```
exit(0);
```

[ @hackerfantastic ]

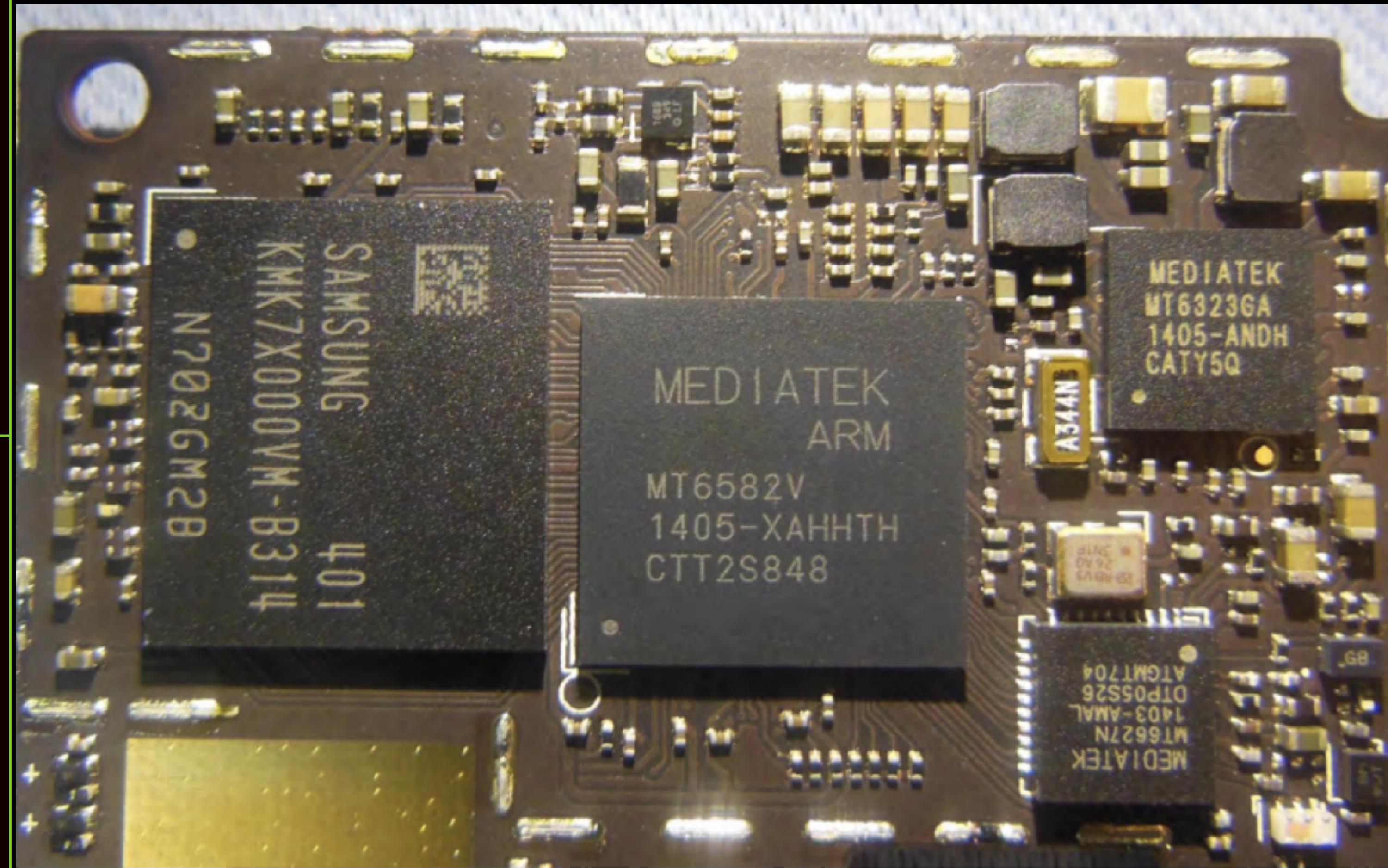
[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
```



```
host = gethostbyname(server);
```

# MEDIATEK MT6582 SoC

9

```
exit(0);
```

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea

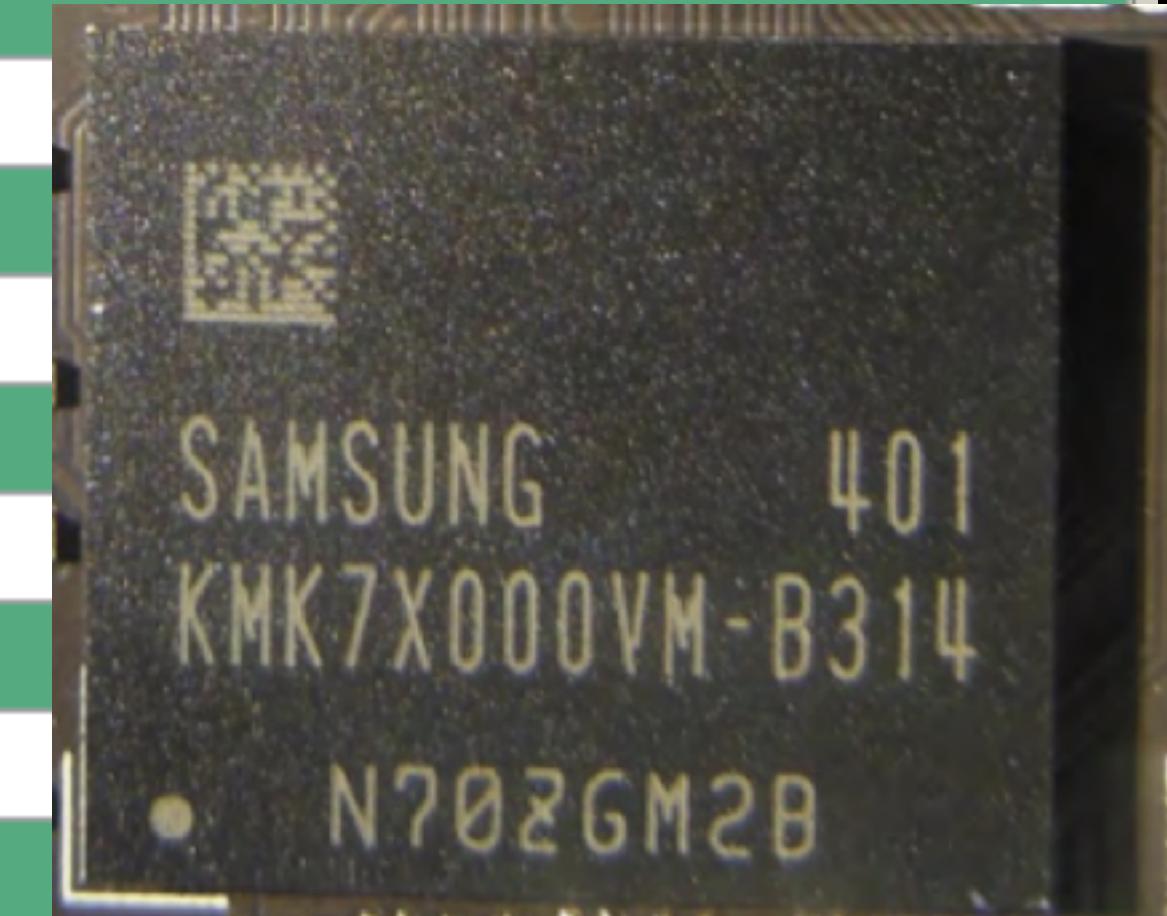


165
166

```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
```

Format All + Download ▾

X	Name	Begin Address	End Address	Location
X	PRELOADER	0x0000000000000000	0x000000000001ba3b	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/preloader_gionee82_wet_jb5.bin
X	MBR	0x0000000000880000	0x00000000008801ff	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/MBR
X	EBR1	0x0000000000900000	0x00000000009001ff	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/EBR1
X	UBOOT	0x00000000025a0000	0x00000000025dfcaf	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/lk.bin
X	BOOTIMG	0x0000000002600000	0x0000000002a78fff	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/boot.img
X	RECOVERY	0x0000000003600000	0x0000000003af77ff	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/recovery.img
X	SEC_RO	0x0000000004600000	0x0000000004620fff	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/secro.img
X	LOGO	0x00000000046c0000	0x00000000047054f5	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/logo.bin
X	EBR2	0x00000000049c0000	0x00000000049c01ff	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/EBR2
X	SPDATA	0x0000000005440000	0x000000000586e093	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/spdata.img
X	ANDROID	0x0000000005c40000	0x00000000043456c77	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/system.img
X	CACHE	0x00000000065c40000	0x0000000006643a0e7	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/cache.img
X	USRDATA	0x0000000007d440000	0x0000000007f3cf297	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/userdata.img
X	FAT	0x000000000f1440000	0x000000000f3d6507b	/home/fantastic/Projects/pyongyang/pyongyang_hacked_rom/WBW5511GI_0202_T5752/fat.img



```
161     host = gethostbyname(server);
162
163
164
```

# Flash Format Layout

10

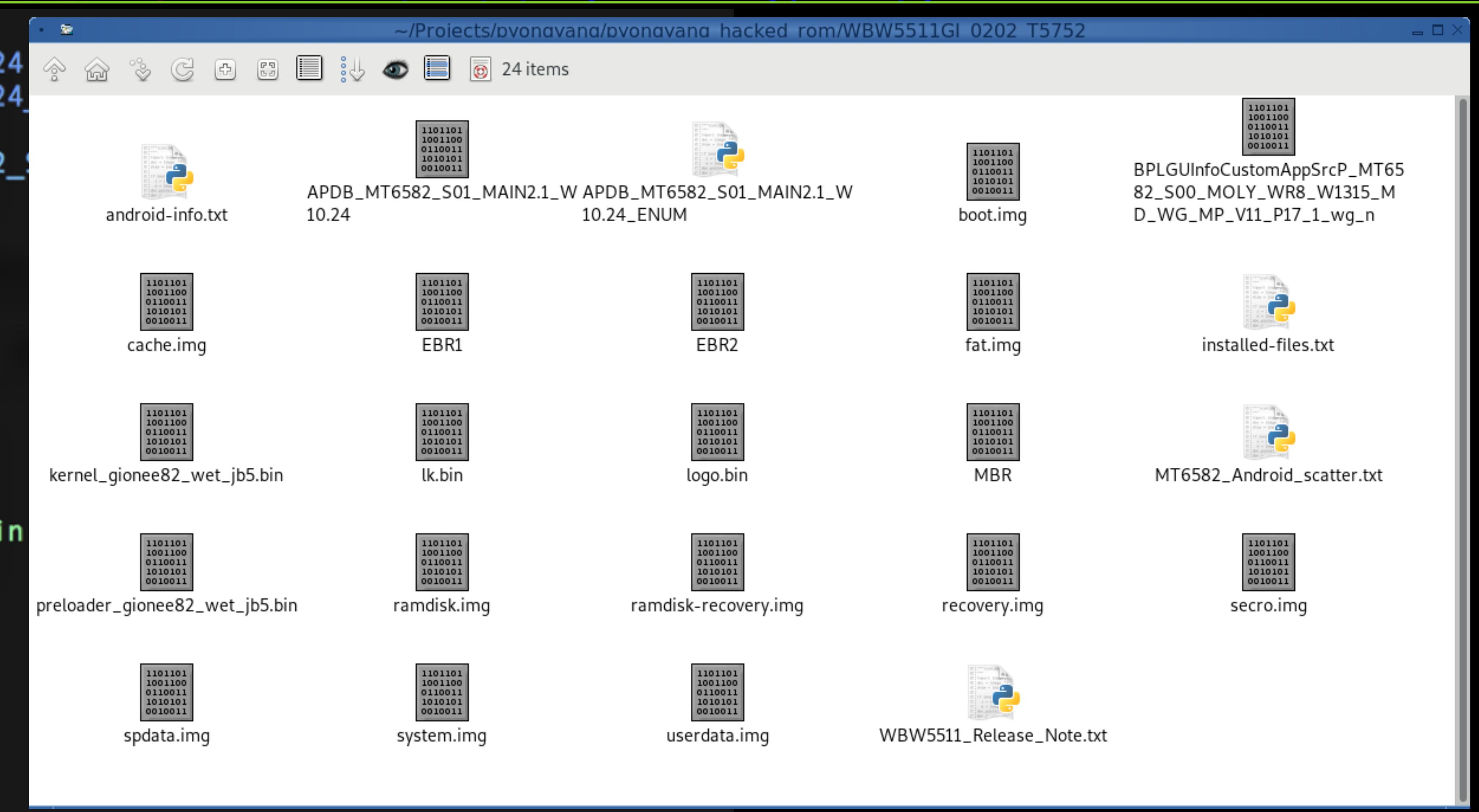
Pyongyang 2407  
Hacking North Korea



```
165     exit(0);
166 }
```

[ @hackerfantastic ] [ @myhackerhouse ] [ https://hacker.house ]

```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147 android-info.txt
148 APDB_MT6582_S01_MAIN2.1_W10.24
149 APDB_MT6582_S01_MAIN2.1_W10.24
150 boot.img
151 BPLGUIInfoCustomAppSrcP_MT6582_
152 cache.img
153 EBR1
154 EBR2
155 fat.img
156 installed-files.txt
157 kernel_gionee82_wet_jb5.bin
158 lk.bin
159 logo.bin
160 MBR
161 MT6582_Android_scatter.txt
162 preloader_gionee82_wet_jb5.bin
163 ramdisk.img
164 ramdisk-recovery.img
165 recovery.img
166 secro.img
167 spdata.img
168 system.img
169 userdata.img
170 WBW5511_Release_Note.txt
```



```
161 host = gethostbyname(server);
162
163 ROM WBW5511GI_0202_T5752
```

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146 =====
147 Build Notes
148 =====
149 1.版本目的:
150 验证前置群辉摄像头偏黄
151 ESD问题
152 2.修改的重大问题点:
153 3.测试重点:
154 4.应用数据库是否改变:
155 =====
156 Release Location
157 =====
158 FTP://18.8.8.2/software_release/WBW5511/WBW5511GI_0202_T5752.zip
159 =====
160 CR List
161 =====
162 [Repository Url]: http://192.168.110.97/svn/android_mtk_jb5_6582_wet/branches/branch_oversea_wbw5506_5511_5508_t5531_rel
163 CR01031315      Opened   WBW5511A      zhanghe 5511 sub camera 代码合入      svn revision(8991)
164 CR01227613      Opened   WBW5511GS     changlei      5511GS资源配置及菜单摆放      svn revision(8992,8990)
165 Tag url: http://192.168.110.97/svn/android_mtk_jb5_6582_wet/tags/TAG_WBW5511_T5752
166 Previous tag: TAG_WBW5511_T5751
167 Code difference: svn diff -r 8992:8987 http://192.168.110.97/svn/android_mtk_jb5_6582_wet/branches/branch_oversea_wbw5506_5511_5508_t5531_rel
168 =====
169 [Repository Url]: http://192.168.110.97/svn/gionee_packages_apk_amigo/branches/branch_oversea_rom4.2.4_rel
170 CR01245664      Opened   WBW5885GI(Amigo)    liuliang      [5885GI_amigo]印度反馈:随变->壁纸中有Lockscreen和Menu wallpaper两项,这两项在navi l
171 auncher项目中无作用,应去掉      svn revision(29171)
172 Tag url: http://192.168.110.97/svn/gionee_packages_apk_amigo/tags/TAG_WBW5511_T5752
173 Previous tag: TAG_WBW5511_T5751
174 Code difference: svn diff -r 29171:29025 http://192.168.110.97/svn/gionee_packages_apk_amigo/branches/branch_oversea_rom4.2.4_rel
175 =====
```

```
161 host = gethostbyname(server);
162
163
164
165
166
```

# Release\_Note FTP Info Leak

12

```
exit(0);
```

[ @hackerfantastic ]

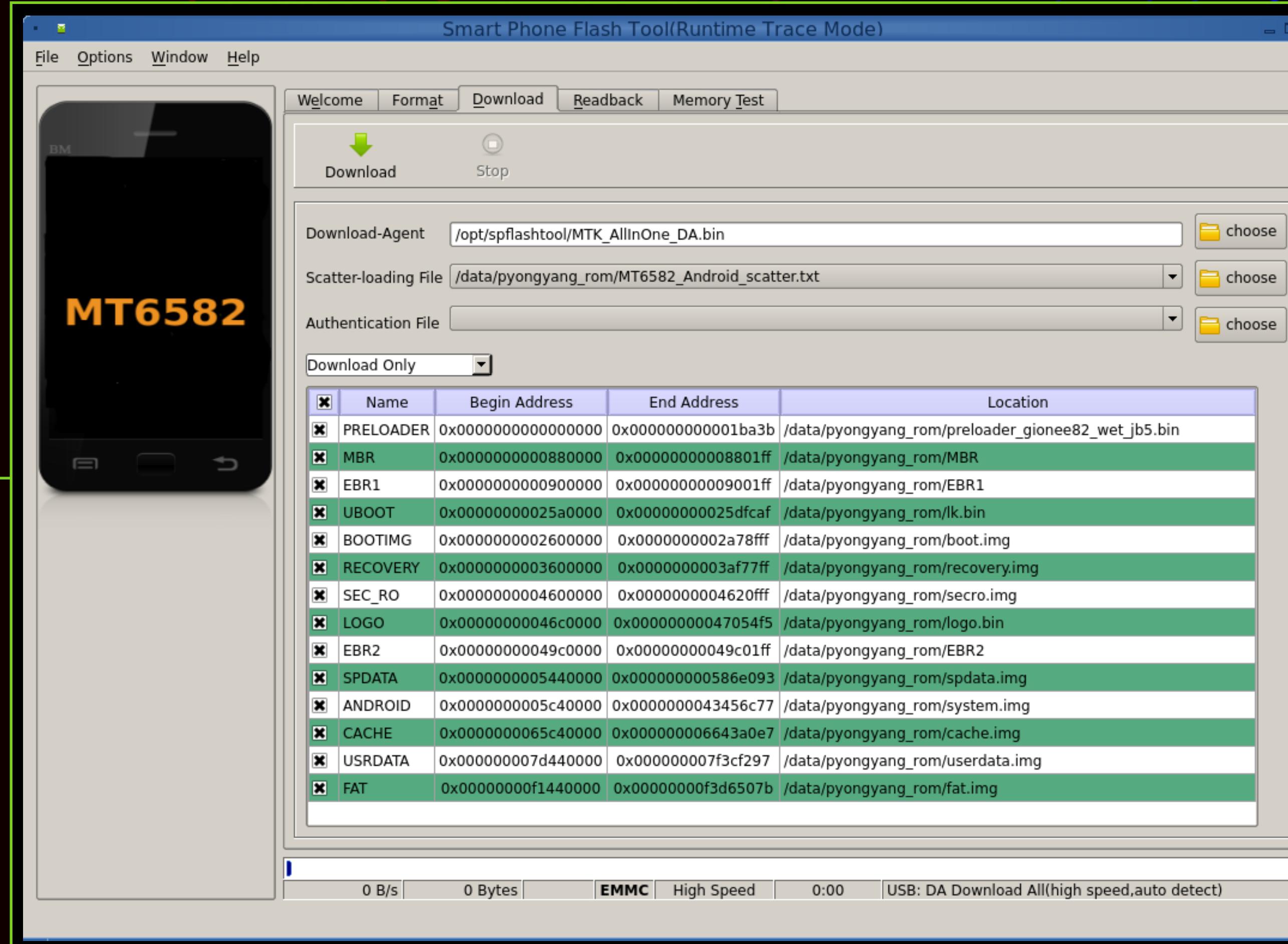
[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161     host = gethostbyname(server);
162
163
164
```



```
[134315.811873] usb 1-5: new high-speed USB device number 40 using xhci_hcd
[134315.954481] usb 1-5: New USB device found, idVendor=0e8d, idProduct=2000, bcdDevice= 1.00
[134315.954488] usb 1-5: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[134315.954492] usb 1-5: Product: MT65xx Preloader
[134315.954496] usb 1-5: Manufacturer: MediaTek
[134316.018783] cdc_acm 1-5:1.1: ttyACM0: USB ACM device
[134317.649985] usb 1-5: USB disconnect, device number 40
```

MediaTek provides tools and support for their overseas & mass-produced boards/devices.

Plugging in the device without a battery will briefly show a USB device...

“spflashtool” is MediaTek way of flashing any MT6582 based device ..

It works early in preloader boot process and allows raw memory & storage hardware access ...

A binary “agent” is transmitted to the device to run in RAM.

# MediaTek SP Flash Tool

13

```
exit(0);
```

[ @hackerfantastic ]

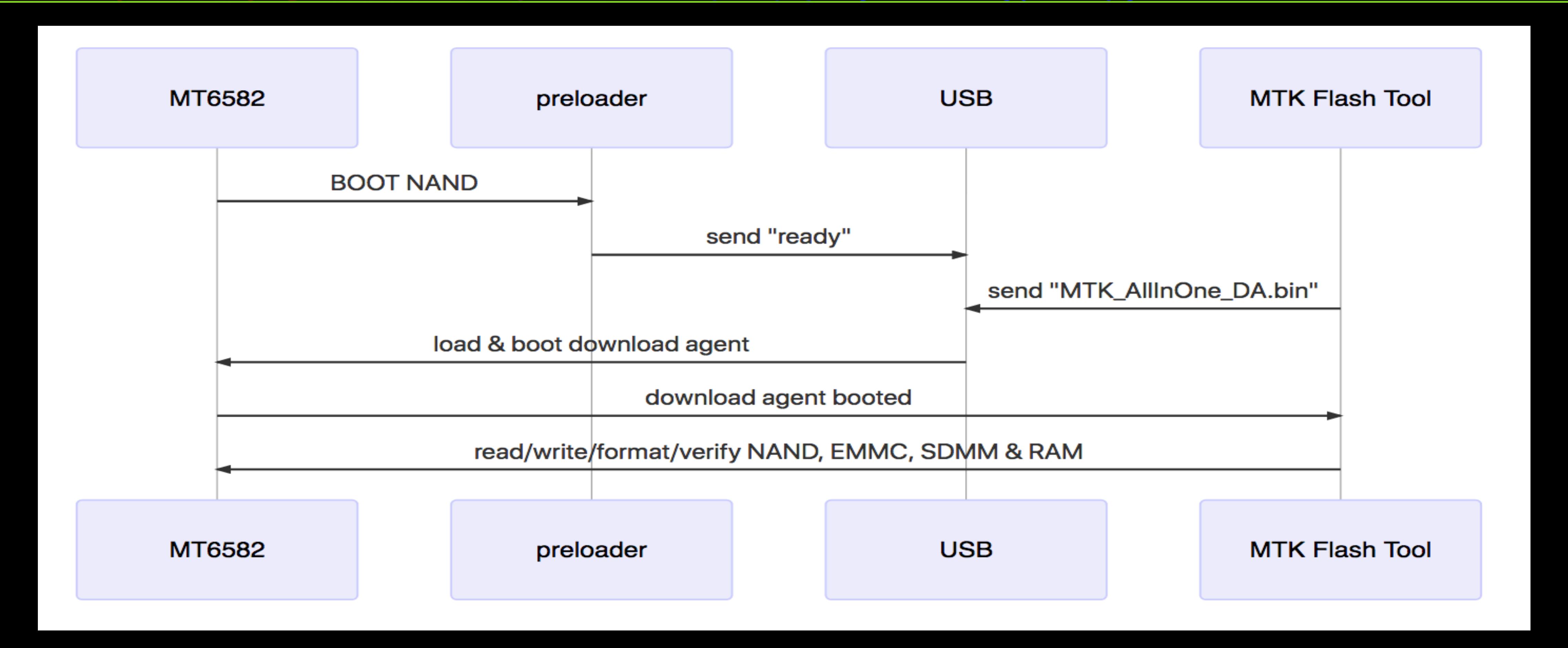
[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147     MT6582
148     preloader
149     USB
150     MTK Flash Tool
```

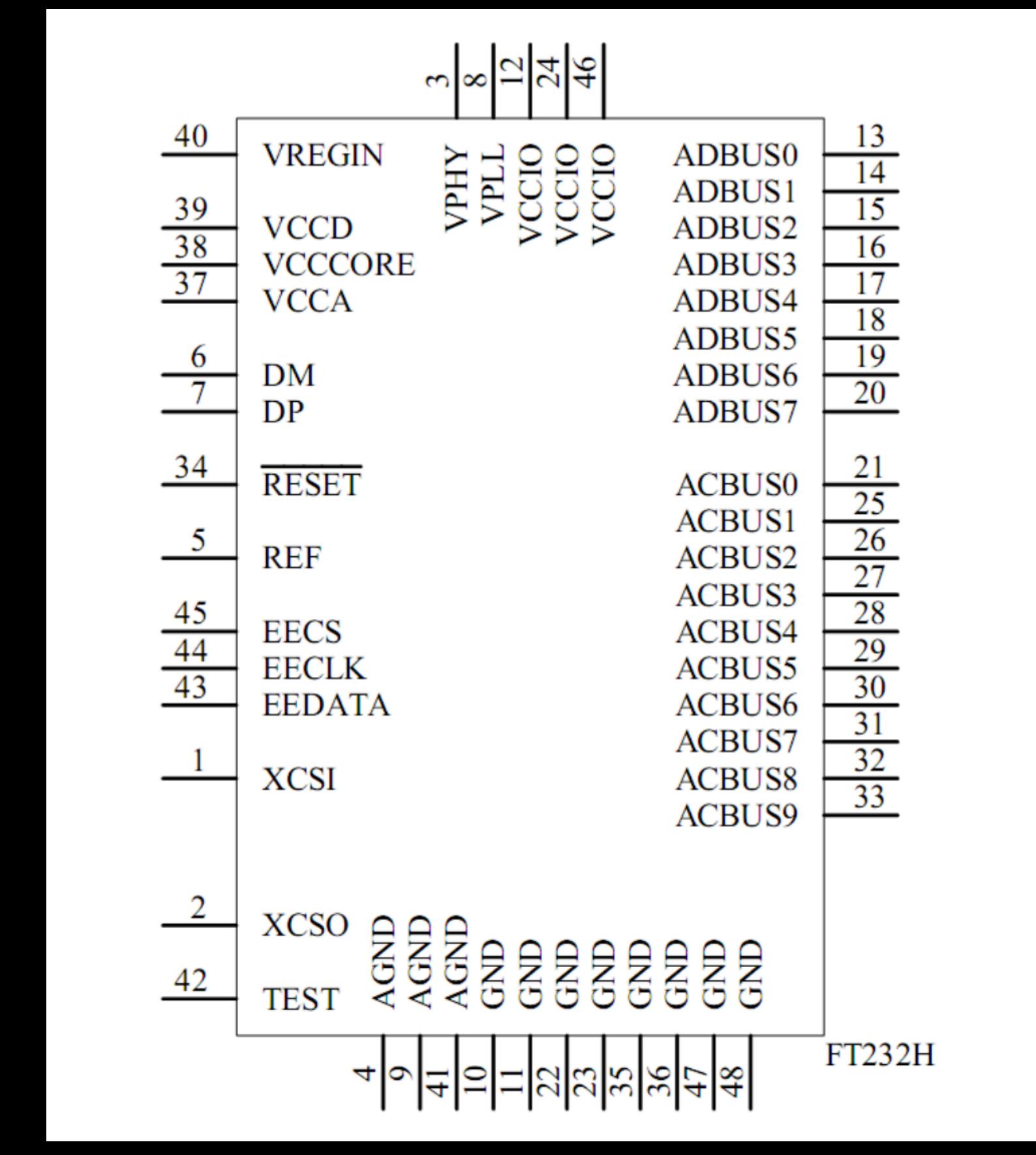
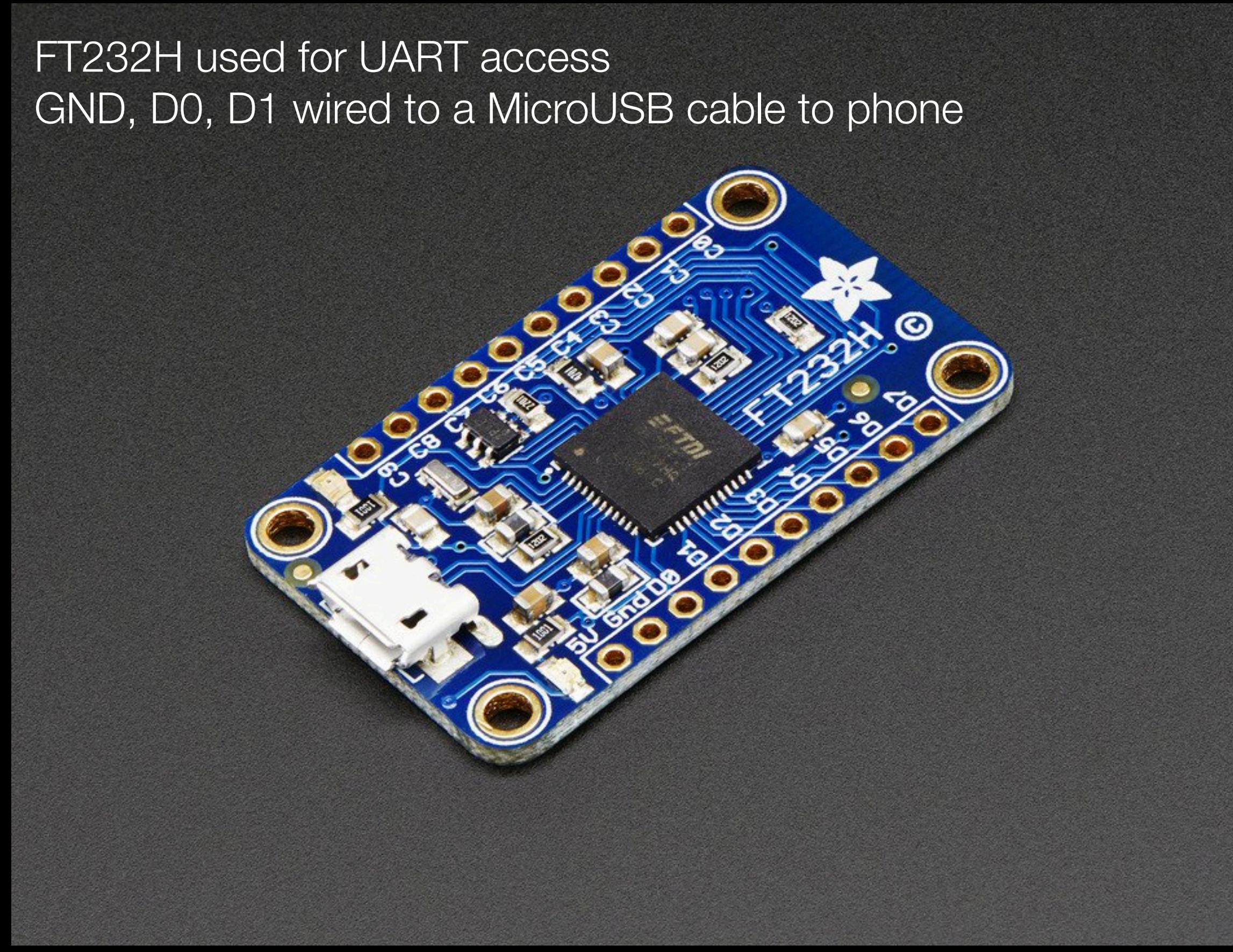


```
161     host = gethostbyname(server);
162
163     MT6582 download agent
164
165     exit(0);
166 }
```

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147 FT232H used for UART access
148 GND, D0, D1 wired to a MicroUSB cable to phone
```



```
161 host = gethostbyname(server);
162
163
164
165
166 }
```

# WBW5511 UART cable

15

```
exit(0);
```

[ @hackerfantastic ]

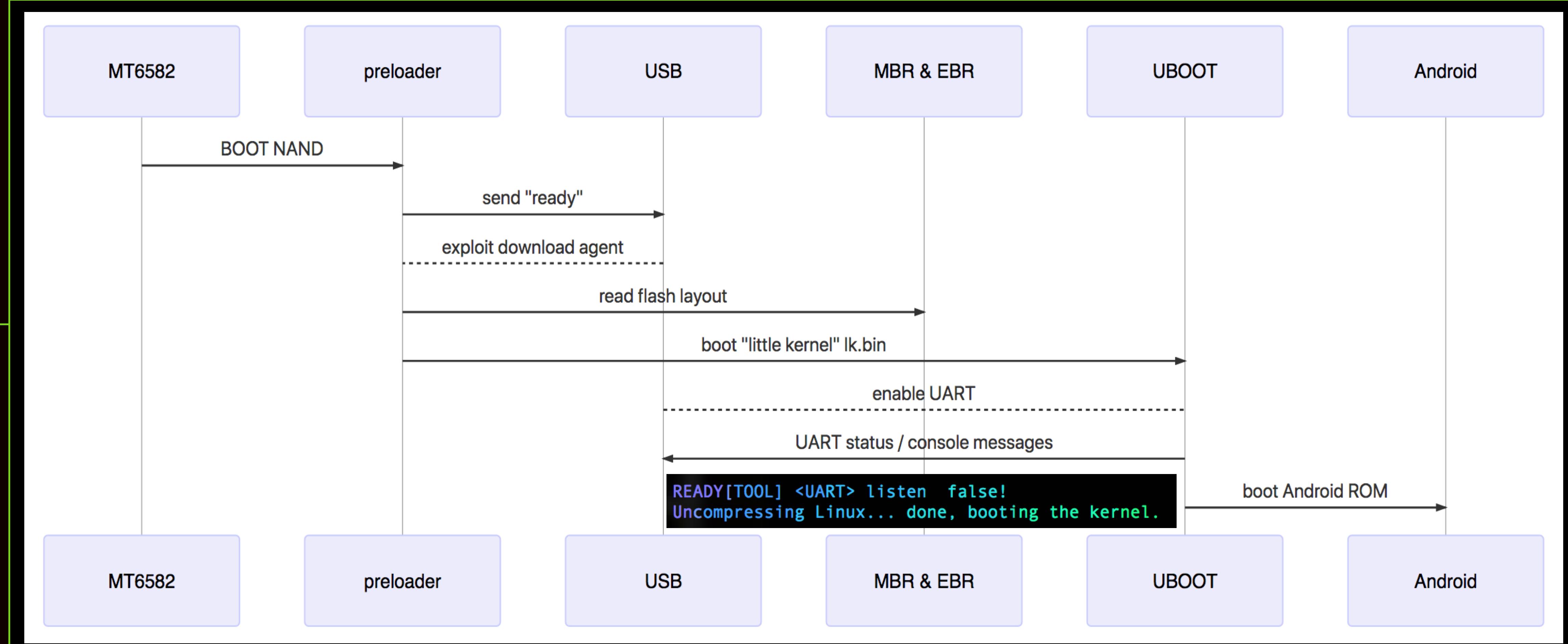
[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147 MT6582
148
149     BOOT NAND
150
151     send "ready"
152
153     exploit download agent
154
155     read flash layout
156
157     boot "little kernel" lk.bin
158
159     enable UART
160
161     UART status / console messages
162
163     READY[TOOL] <UART> listen false!
164     Uncompressing Linux... done, booting the kernel.
165
166     boot Android ROM
```



```
host = gethostbyname(server);
```

# MT6582 BOOT & USB access

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147 shell@평양:/system/bin # cat /proc/cmdline
148 console=ttyMT3,921600n1 vmalloc=496M slub_max_order=0 lcm=1-gn_tm_otm9605a fps=5911 bootprof.pl_t=3264 bootpr
of.lk_t=1621 printk.disable_uart=1 boot_reason=4 boot_mode=0
```

## patch lk.bin, changes bootarg ok. no shell :(

```
1|shell@평양:/proc # cat cmdline
2 console=ttyMT3,921600n1 vmalloc=496M slub_max_order=0 lcm=1-gn_tm_otm9605a fps=5911 bootprof.pl_t=3990 bootpr
3 of.lk_t=1604 printk.disable_uart=0 boot_reason=4 boot_mode=0
```

```
READY[TOOL] <UART> listen false!
Uncompressing Linux... done, booting the kernel.
```

```
shell@평양:/proc # ls -al /dev/ttyMT*
156 crw----- root      root      204, 209 2014-01-01 09:47 ttyMT0
157 crw----- root      root      204, 210 2014-01-01 09:47 ttyMT1
158 crw-rw---- system   system    204, 211 2014-01-01 09:47 ttyMT2
159 crw----- root      root      204, 212 2014-01-01 09:47 ttyMT3
shell@평양:/proc # busybox microcom -s 921600 /dev/ttyMT0
shell@평양:/proc # busybox microcom -s 921600 /dev/ttyMT1
shell@평양:/proc # busybox microcom -s 921600 /dev/ttyMT2
shell@평양:/proc # busybox microcom -s 921600 /dev/ttyMT3
shell@평양:/proc #
```

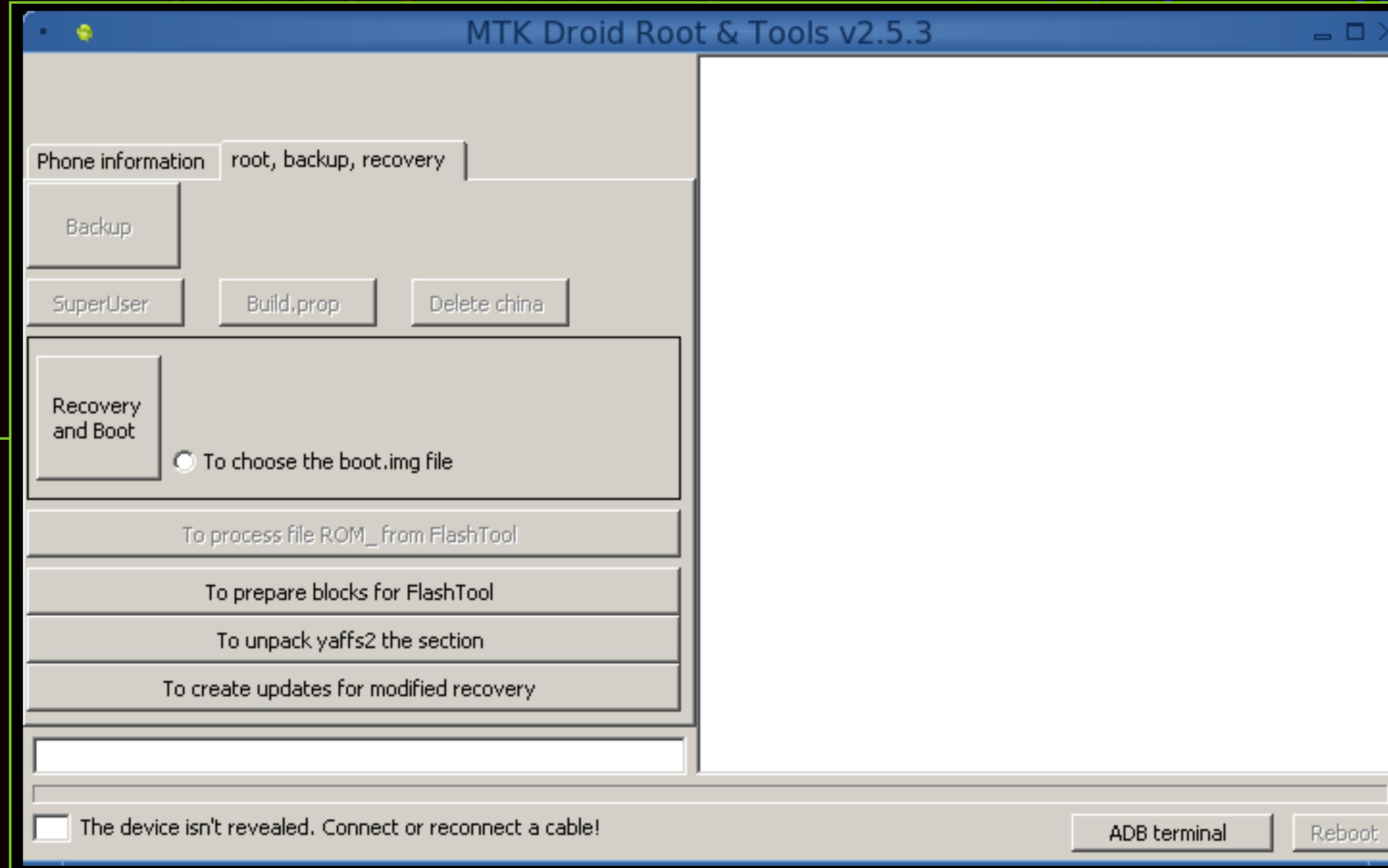


```
161 host = gethostbyname(server);
162
163 lk.bin printk.disable_uart=0
164
165 exit(0);
166 }
```

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148     Phone information  root, backup, recovery
149     Backup
150
151     SuperUser  Build.prop  Delete china
152
153     Recovery and Boot
154     To choose the boot.img file
155
156     To process file ROM_From FlashTool
157
158     To prepare blocks for FlashTool
159
160     To unpack yaffs2 the section
161
162     To create updates for modified recovery
163
164
165     host = gethostbyname(server);
166
167     exit(0);
168 }
```



Windows binary executable

Can create “scatter file” backups,  
uses adb and custom recovery.img

Backups compatible with  
“spflashtool”

Back up and protect the preloader (no  
preloader, no RAM recovery)

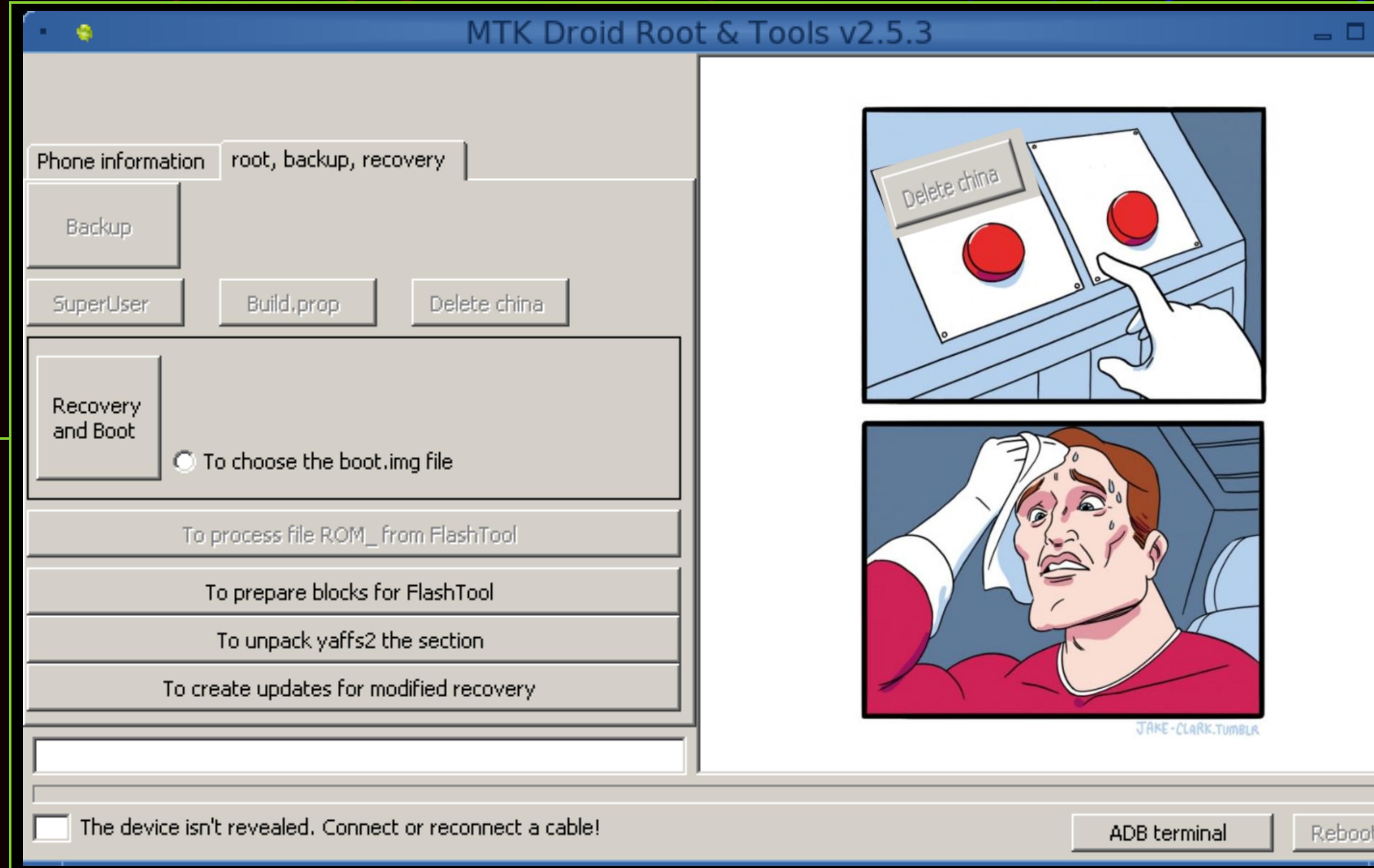
# MTK Droid Root & Tools

18

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161     host = gethostbyname(server);
162
163
164
```



***"Delete not used and Chinese apps. Reads from files\_for\_delete.txt"***

A CWM backup & recovery from recovery.img mode.

# Deleting China?

19

```
exit(0);
```

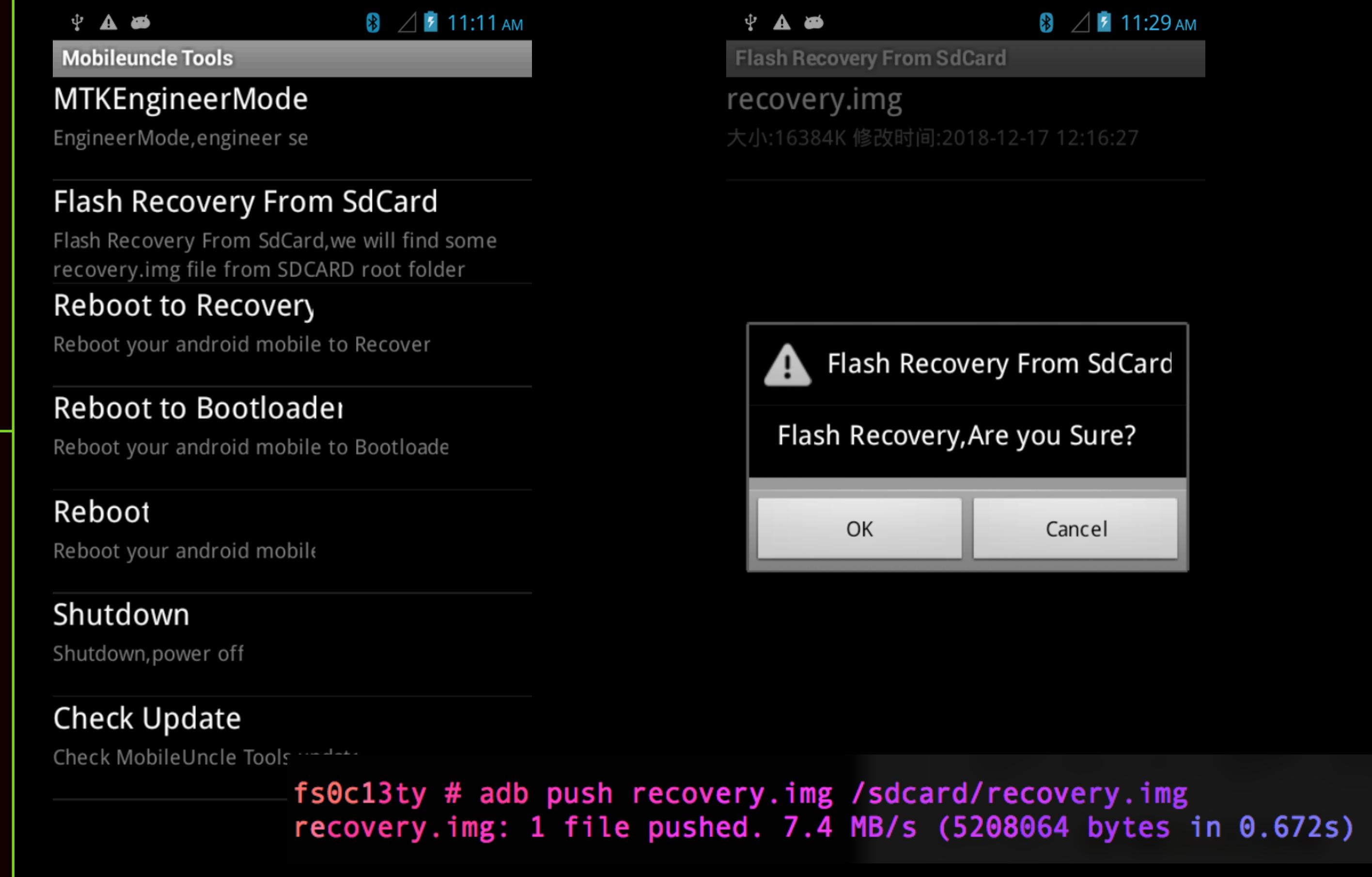
[ @hackerfantastic ] [ @myhackerhouse ] [ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



165
166

```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147     MTKEngineerMode
148     EngineerMode,engineer se
149
150     Flash Recovery From SdCard
151     Flash Recovery From SdCard,we will find some
152     recovery.img file from SD CARD root folder
153     Reboot to Recovery
154     Reboot your android mobile to Recover
155
156     Reboot to Bootloader
157     Reboot your android mobile to Bootloader
158
159     Reboot
160     Reboot your android mobile
161
162     Shutdown
163     Shutdown,power off
164
165     Check Update
166     Check MobileUncle Tools
```



Elliot Alderson  
@fs0c131y

Follow

1. On the top it's the system/app folder of the [@BLU\\_Products](#) Vivo 8. At the bottom it's the system/app folder found in the Pyongyang 2407 Cellphone disk image released at 34c3 by [@willscott](#) [koreacomputercenter.org](#)

7:24 am - 5 Jan 2018

8 Retweets 25 Likes

4 8 25

```
host = gethostbyname(server):
```

Mobileuncle Tools: hi friend.

20

```
exit(0);
```

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161     host = gethostbyname(server);
162
163
164
165     exit(0);
166 }
```

# 평양 2407 Android

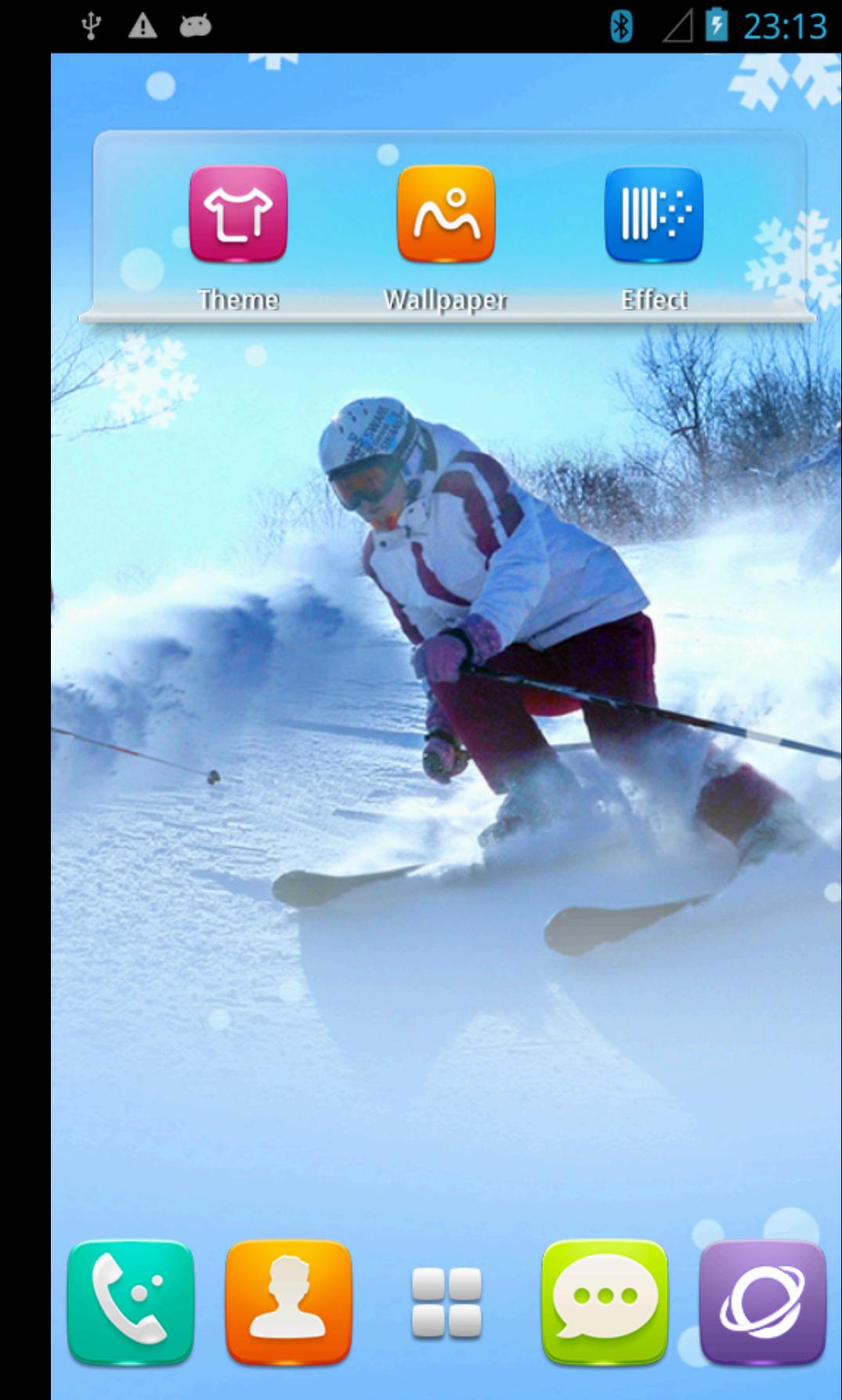
## Based on Jellybean 4.2.2\*

The Pyongyang 2407 Cellphone disk image contains the “system” and “data” ext4 partitions as tar files. It also contains a kernel zImage and ramdisk.gz.

Indicators from the device were removed and cleaned off before taken out of North Korea. Patching needed to work on aftermarket devices, added to github.

**Patched ROM files, archive & tools [https://github.com/hackerhouse-opensource/pyongyang\\_2407](https://github.com/hackerhouse-opensource/pyongyang_2407)**

More APK's to install and play with on KCC <http://www.koreacomputercenter.org/>



평양 2407 Android ROM

21

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
```



```
161     host = gethostbyname(server);
162
163
164
```

# DPRK Android Applications

22

```
exit(0);
```

[ @hackerfantastic ]

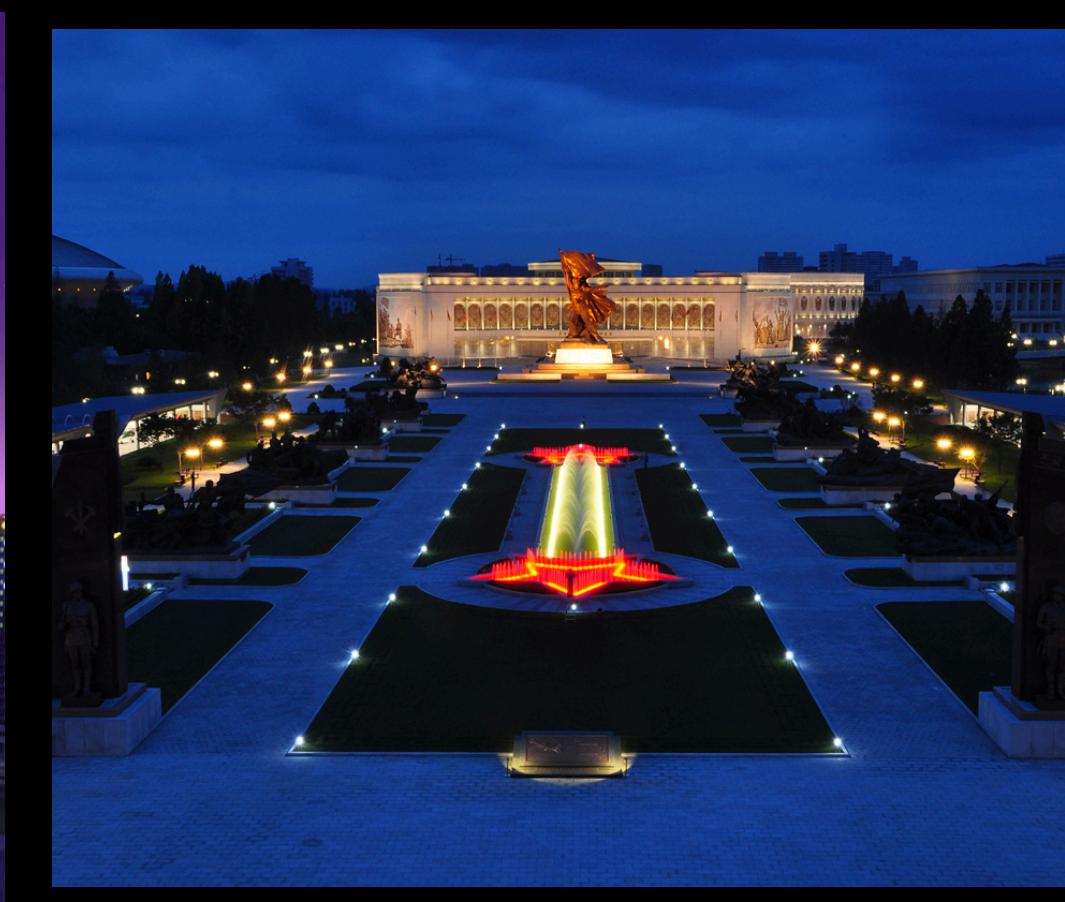
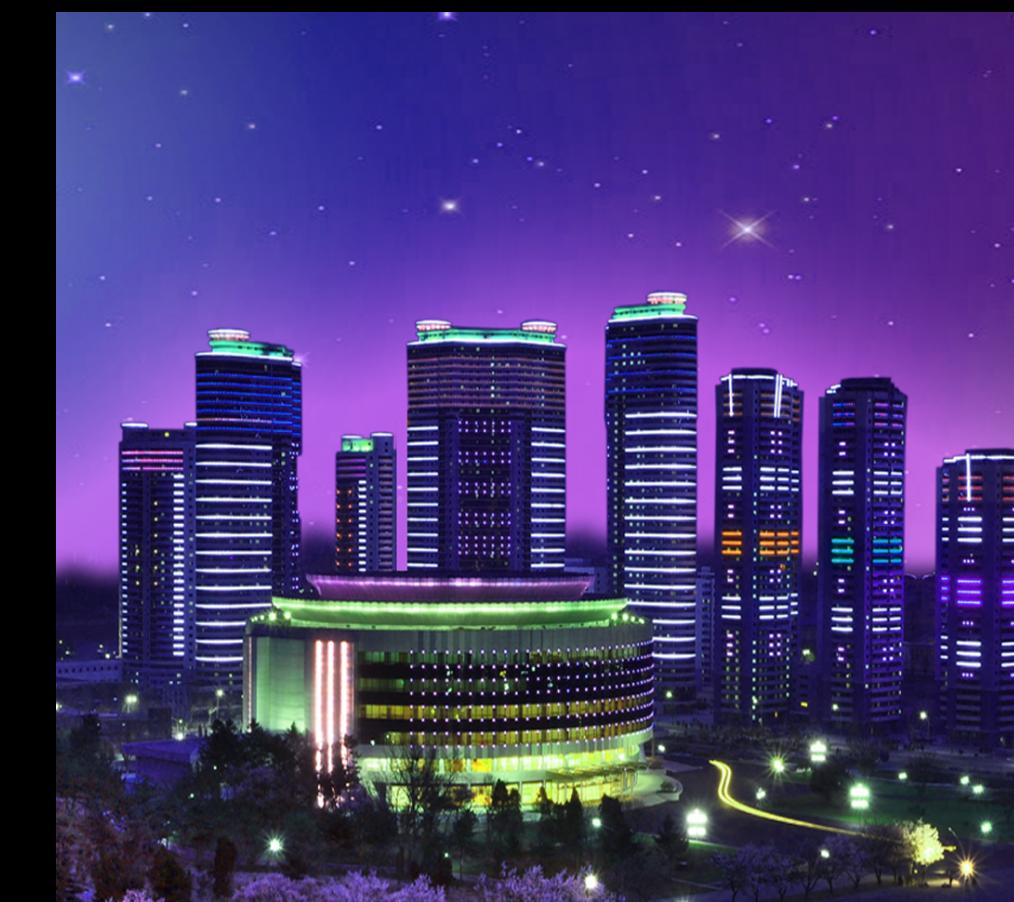
[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
bzero(&(sa.sin_zero),8);
printf("[ connecting to %s %d/tcp\n",server.port);
```



[host -> https://github.com/hackerhouse-opensource/pyongyang\\_2407/tree/master/wallpapers](https://github.com/hackerhouse-opensource/pyongyang_2407/tree/master/wallpapers)



# North Korea Wallpapers

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147 GET / HTTP/1.1
148 Host: 127.0.0.1
149 Connection: keep-alive
150 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
151 x-wap-profile: http://218.249.47.94/Xianghe/MTK_Phone_JB_UAprofile.xml
152 X-Requested-With: com.android.browser
153 User-Agent: Mozilla/5.0 (Linux; U; Android 4.2.2; en-us; 2407 Build/JDQ39) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mobile Safari/534.30
154 Accept-Encoding: gzip,deflate
155 Accept-Language: en-US
156 Accept-Charset: utf-8, utf-16, *;q=0.7
```



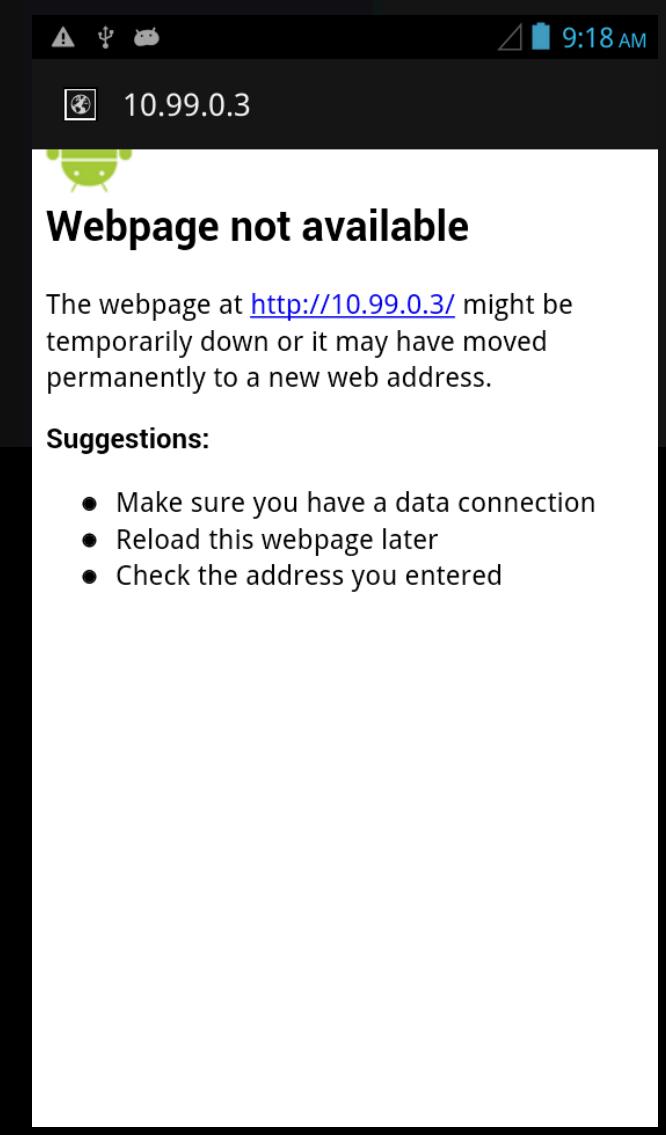
열람기

Android Webview Browser locked to the KwangMyong intranet

Wifi disabled in ROM (attempted to enable, missing code)

3G/CDMA Koroyolink network support

rtsp:// handler opens the com.gionee.video player activity

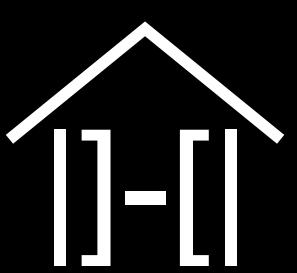


```
161 host = gethostbyname(server);
162
163
164 exit(0);
```

# Web Browser: KwangMyong

24

Pyongyang 2407  
Hacking North Korea



```

144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146

```

X Certificate and Key management

Internal name	commonName	CA	Serial	Expiry date
Autoridad de Certificacion Firmaprofesional CIF A62634068	Autoridad de Certificacion Firmaprofesional CIF A62634068	✓ Yes	01	24/10/2013
KISA RootCA 3	KISA RootCA 3	✓ Yes	02	19/11/2014
TÜRKTRUST Elektronik Sertifika Hizmet Sağlayicisi	TÜRKTRUST Elektronik Sertifika Hizmet Sağlayicisi	✓ Yes	01	22/03/2015
Buypass Class 3 CA 1	Buypass Class 3 CA 1	✓ Yes	02	09/05/2015
A-Trust-nQual-03	A-Trust-nQual-03	✓ Yes	016C1E	17/08/2015
TÜRKTRUST Elektronik Sertifika Hizmet Sağlayicisi	TÜRKTRUST Elektronik Sertifika Hizmet Sağlayicisi	✓ Yes	01	16/09/2015
Staat der Nederlanden Root CA	Staat der Nederlanden Root CA	✓ Yes	98968A	16/12/2015
CA Disig	CA Disig	✓ Yes	01	22/03/2016
EBG Elektronik Sertifika Hizmet Sağlayicisi	EBG Elektronik Sertifika Hizmet Sağlayicisi	✓ Yes	4CAF73421C8E7402	14/08/2016
Juur-SK	Juur-SK	✓ Yes	3B8E4BFC	26/08/2016
Buypass Class 2 CA 1	Buypass Class 2 CA 1	✓ Yes	01	13/10/2016
e-Guvan Kok Elektronik Sertifika Hizmet Saglayicisi	e-Guvan Kok Elektronik Sertifika Hizmet Saglayicisi	✓ Yes	44998D3CC00327BD9C7695B9EADBCB5	04/01/2017
Microsec e-Szigno Root CA	Microsec e-Szigno Root CA	✓ Yes	CCB8E7BF4E291AFDA2DC66A51C2C0F11	06/04/2017
TÜBITAK UEKAE Kök Sertifika Hizmet Sağlayicisi - Sürüm 3	TÜBITAK UEKAE Kök Sertifika Hizmet Sağlayicisi - Sürüm 3	✓ Yes	11	21/08/2017
DST ACES CA X6	DST ACES CA X6	✓ Yes	0D5E990AD69DB778ECD807563B8615D9	20/11/2017
ApplicationCA	GTE CyberTrust Global Root	✓ Yes	31	12/12/2017
GTE CyberTrust Global Root		✓ Yes	01A5	13/08/2018
Equifax Secure Certificate Authority		✓ Yes	35DEF4CF	22/08/2018
DSTCA E2		✓ Yes	366ED3CE	09/12/2018
DSTCA E1		✓ Yes	36701596	10/12/2018
NetLock Kozjegyzo (Class A) Tanusitvanykiado	NetLock Kozjegyzo (Class A) Tanusitvanykiado	✓ 4	0103	19/02/2019
NetLock Expressz (Class C) Tanusitvanykiado	NetLock Expressz (Class C) Tanusitvanykiado	✓ 4	68	20/02/2019
NetLock Uzleti (Class B) Tanusitvanykiado	NetLock Uzleti (Class B) Tanusitvanykiado	✓ 4	69	20/02/2019
GeoTrust Global CA 2	GeoTrust Global CA 2	✓ Yes	01	04/03/2019
FNMT Clase 2 CA		✓ Yes	36F11B19	18/03/2019
Entrust.net Secure Server Certification Authority	Entrust.net Secure Server Certification Authority	✓ Yes	374AD243	25/05/2019
Equifax Secure eBusiness CA-2		✓ Yes	3770CFB5	23/06/2019
UTN - DATAcorp SGC	UTN - DATAcorp SGC	✓ Yes	44BE0C8B500021B411D32A6806A9AD69	24/06/2019
http://www.valicert.com/	http://www.valicert.com/		01	25/06/2019
http://www.valicert.com/	http://www.valicert.com/		01	26/06/2019
http://www.valicert.com/	http://www.valicert.com/		01	26/06/2019
Class 2 Primary CA	Class 2 Primary CA	✓ 10	85BD4BF3D8DAE369F694D75FC3A54423	06/07/2019
UTN-USERFirst-Hardware	UTN-USERFirst-Hardware	✓ Yes	44BE0C8B500024B411D3362AFE650AFD	09/07/2019
UTN-USERFirst-Network Applications	UTN-USERFirst-Network Applications	✓ Yes	44BE0C8B500024B411D336304BC03377	09/07/2019
Deutsche Telekom Root CA 2	Deutsche Telekom Root CA 2	✓ 5	26	09/07/2019
Staat der Nederlanden Root CA - G2	Staat der Nederlanden Root CA - G2	✓ Yes	98968C	25/03/2020
AddTrust External CA Root	AddTrust External CA Root	✓ Yes	01	30/05/2020
Equifax Secure Global eBusiness CA-1	Equifax Secure Global eBusiness CA-1	✓ Yes	01	21/06/2020
Equifax Secure eBusiness CA-1	Equifax Secure eBusiness CA-1	✓ Yes	04	21/06/2020
SecureSign RootCA1	SecureSign RootCA1		5F60585F00000000	15/09/2020

Database: dprk\_ssl\_CA\_certs.xdb

Search

No DPRK root CA?

No Pyongyang root CA?

Test Certificate for Gionee.

Common global CA chain  
for Android WebView based  
browsers.

# SSL Certificates

25

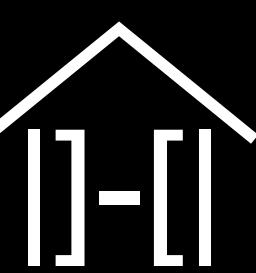
```
exit(0);
```

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147 package                                     code
148 com.android.providers.calendar             225
149 com.android.settings                      4636
150 com.mediatek.bluetooth                   87410
151
152                                         2872714
153                                         2872713
154                                         2872710
155
156                                         3646633
157
158 com.mediatek.engineermode
159
160
161 host = gethostbyname(server);
```

package	code
com.android.providers.calendar	225
com.android.settings	4636
com.mediatek.bluetooth	87410
	2872714
	2872713
	2872710
com.mediatek.engineermode	3646633

# Phone Secret Codes

26

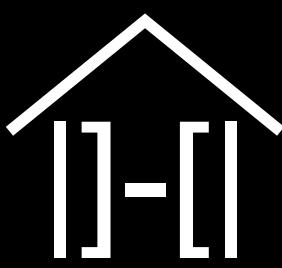
```
exit(0);
```

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161     host = gethostbyname(server);
162
163
164
165
166 }
```



North Korea has been described as a "*massive police state*", and its people "*under constant surveillance*".

booted mobile device, use it for a few days as a phone.

GOAL: watch "*The Interview*" movie without detection or alerting in the handset.



# North Korean Surveillance

27

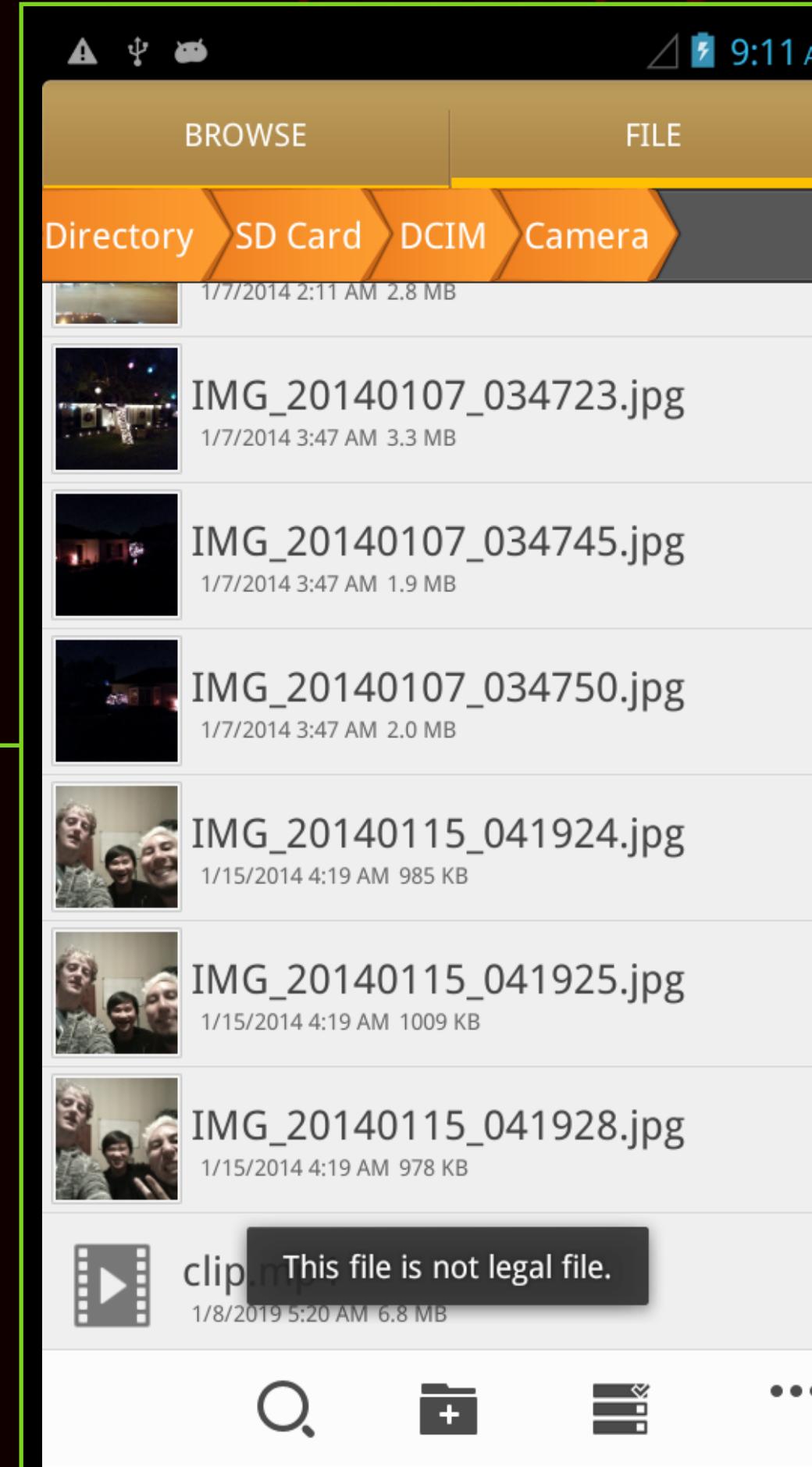
```
exit(0);
```

[ @hackerfantastic ] [ @myhackerhouse ] [ https://hacker.house ]

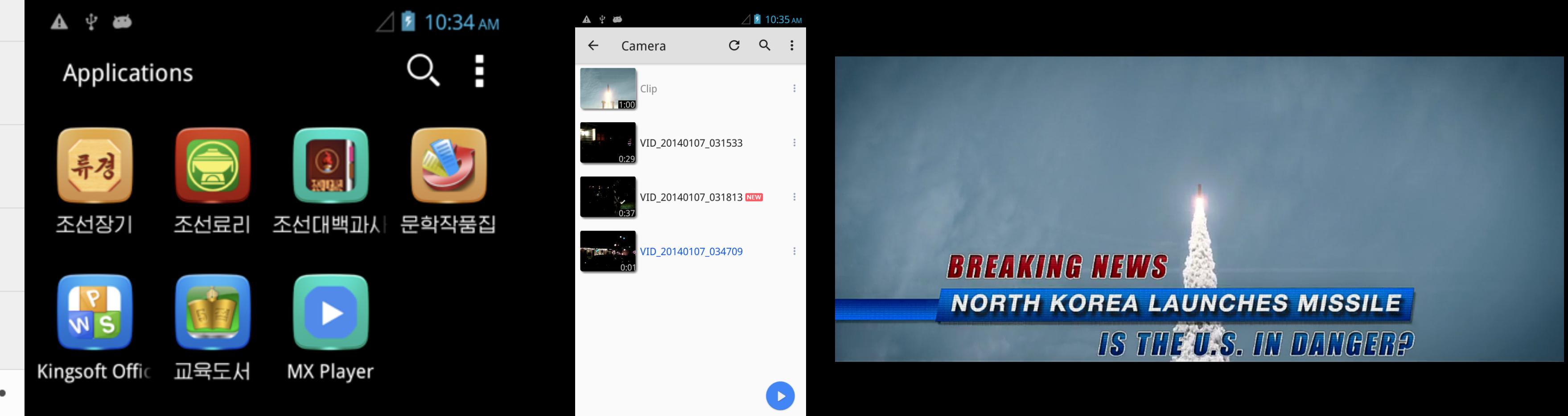
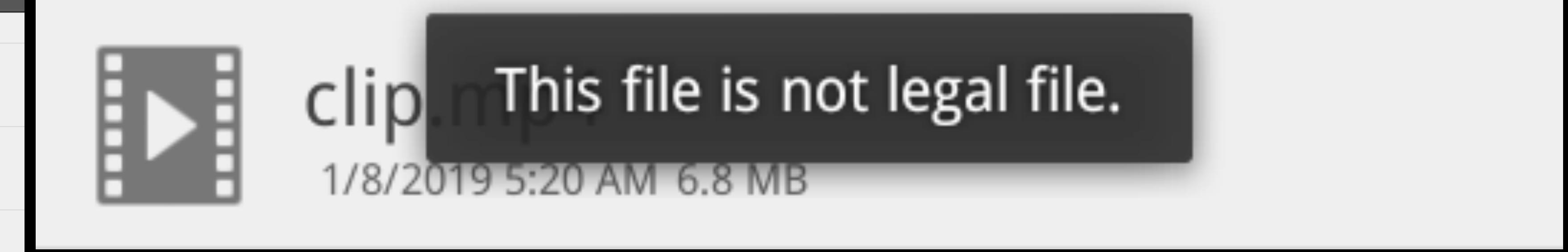
Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
```



```
$ adb push The.Interview.2014.mp4 /sdcard/Video
The.Interview.2014.mp4: 1 file pushed. 3.0 MB/s (851606052 bytes in 270.372s)
```



```
161     host = gethostbyname(server);
162
163
164
```

Legal Files (adb sideload apps) Pyongyang 2407  
Hacking North Korea [ ]-[ ]

```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147 Package: kuts.ktr.RFService
148     Application Label: 전자서명조작체계 2.0
149     Process Name: kuts.ktr.RFService
150     Version: 2.0
151     Data Directory: /data/data/
152     kuts.ktr.RFService
153     APK Path: /system/app/RedFlag.apk
154     UID: 10038
155     GID: [1015, 1023, 1028]
156     Shared Libraries: null
157     Shared User ID: null
158
159     Package: kuts.ktr.RFService
160     kuts.ktr.RFService.RFReceiver
161     Permission: null
```

#### Uses Permissions:

- android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS
- android.permission.WRITE\_EXTERNAL\_STORAGE
- android.permission.RECEIVE\_BOOT\_COMPLETED
- android.permission.WRITE\_MEDIA\_STORAGE
- android.permission.REBOOT
- android.permission.DEVICE\_POWER
- android.permission.READ\_FRAME\_BUFFER
- android.permission.WAKE\_LOCK
- android.permission.VIBRATE
- android.permission.GET\_TASKS
- android.permission.READ\_PHONE\_STATE
- android.permission.WRITE\_SETTINGS
- android.permission.WRITE\_SECURE\_SETTINGS
- android.permission.WRITE\_SYNC\_SETTINGS
- android.permission.READ\_SYNC\_SETTINGS
- android.permission.READ\_EXTERNAL\_STORAGE

#### Defines Permissions:

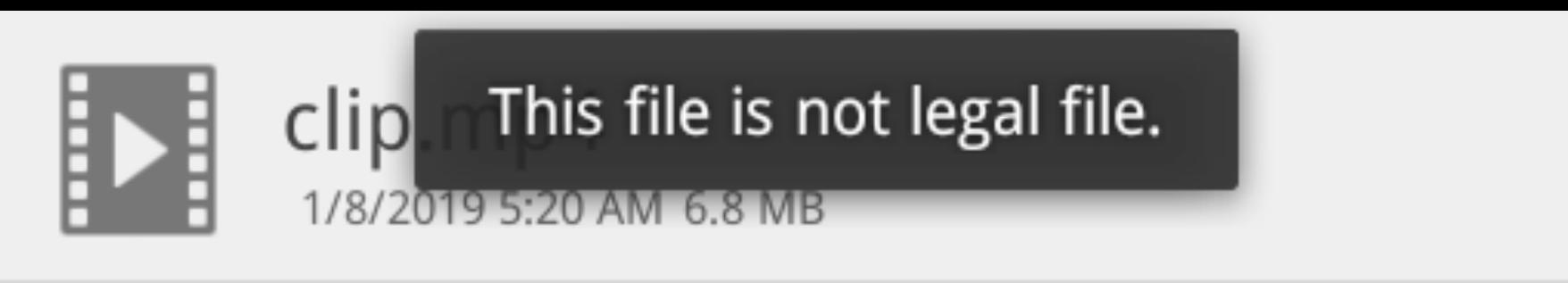
- None

```
161     host = gethostbyname(server);
162
163     전자서명조작체계 2.0 (RedFlag)
```



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
146 D/gov_sign( 1866): -----.3gp-----4---
147 D/gov_sign( 1866): get_flength : file name = /storage/sdcard0/DCIM/Camera/VID_20140107_031533.3gp, fSize = 35604822, POSTFIX_LEN = 0
148 D/gov_sign( 1866): MnsNative isNatSignFile : file name = /storage/sdcard0/DCIM/Camera/VID_20140107_031533.3gp, result = 264
149 E/MediaSelfSign( 1866): Call isSelfSignFile()----- succeeded!
150 E/MediaSelfSign( 1866): MSSMediaVerifyFileByPath()-----! return 0
151 E/MediaSelfSign( 1866): MSSMediaVerifyFileByPath()-----! return 0
152 E/MediaSelfSign( 1866): isSelfSignFile()-----successed! Err = 0, return 1
153 E/MediaSelfSign( 1866): Call CheckSelfSignFile()----- succeeded!
154 E/MediaSelfSign( 1866): MSSMediaVerifyFileByPath()-----! return 0
155 E/MediaSelfSign( 1866): CheckSelfSignFile()-----successed! Err = 0, return 2
```

```
153 D/gov_sign( 1866): -----.pdf-----4---
154 D/gov_sign( 1866): get_flength : file name = /storage/sdcard0/프로그래밍자료/.KwangMyong/도움말.pdf, fSize = 1224818, POSTFIX_LEN = 0
155 D/gov_sign( 1866): MnsNative isNatSignFile : file name = /storage/sdcard0/프로그래밍자료/.KwangMyong/도움말.pdf, result = 1
156 D/SignalClusterView( 719): apply() setImageResource(mMobileTypeId) mShowSimIndicator = false
```



**평양 2407 has two kinds of signed content**

- **Media Self-Signed (Camera, Video, Docs)**
- **Nat Signed (gov\_sign) approved gov media**

전자서명조작체계 logcat API trace info leak

## NATISIGN & SELFSIGN

30

```
exit(0);
```

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144         bzero(&(sa.sin_zero),8);
145         printf("[ connecting to %s %d/tcp\n",server,port);
146
147 1|shell@평양:/system/lib # ls -al *Legal*
148 -rw-r--r-- root      root      18020 2014-12-29 12:39 libLegalInterface.so
149 shell@평양:/system/lib # ls -al *sign*
150 -rw-r--r-- root      root      26244 2014-03-22 01:36 libmedianatsign.so
151 -rw-r--r-- root      root      61564 2015-04-23 05:45 libmediaselfsign.so
152
153 int MSSIsSelfSigned(int arg0, int arg1) {
154     r1 = arg1;
155     sp = sp - 0x30;
156     r7 = arg0 + 0x0;
157     r6 = r1 + 0x0;
158     var_1C = **0xaa7c;
159     *r1 = 0x0;
160     r3 = lseek64(sign_extend_32(*(int16_t *)(arg0 + 0xe)), r1, 0x0) + 0x0;
161     if (r1 == 0x0) {
162         r0 = 0x0;
163         if (r3 > 0x7) {
164             r0 = sign_extend_32(*(int16_t *)(r7 + 0xe));
165             r0 = lseek64(r0, r1, 0xffffffff8);
166             r0 = r0 + 0x1;
167             if (r0 != 0x0) {
168                 r0 = sign_extend_32(*(int16_t *)(r7 + 0xe));
169                 r7 = sp + 0x8;
170                 if (read(r0, r7 + 0x0, 0x8) != 0x8) {
171                     r0 = 0x0;
172                     *r6 = 0x2a1;
173                 }
174             } else {
175                 r3 = memcmp(r7 + 0x0, "SELFSIGN", 0x8) + 0x0;
176                 r0 = 0x0 - r3;
177                 r0 = r0 + r3 + CARRY(FLAGS);
178             }
179         }
180     }
181
182     host = gethostbyname(server);
183
184     loc_3b9e:
185         var_C = lseek64(sign_extend_32(*(int16_t *)(r5 + 0xe)), 0xe, 0x0);
186         var_8 = 0xe;
187         android_log_print();
188         if (var_8 != 0x0) {
189             r2 = var_C + 0xffffffff8;
190             lseek64(sign_extend_32(*(int16_t *)(r5 + 0xe)) + 0x0, 0x0, r2);
191             read(sign_extend_32(*(int16_t *)(r5 + 0xe)), r6 + 0x0, 0x8);
192             r0 = strncmp(r6 + 0x0, "NATISIGN" + 0x0, 0x8);
193             if (r0 != 0x0) {
194                 r2 = 0xfffffffffc + var_C;
195                 lseek64(sign_extend_32(*(int16_t *)(r5 + 0xe)), 0x0, r2);
196                 if (read(sign_extend_32(*(int16_t *)(r5 + 0xe)), sp + 0x334, 0x4) == 0x4) {
197                     var_4 = arg_310;
198                     r0 = var_C - var_4;
199                     r1 = var_8 - 0x0 + !CARRY(FLAGS);
200                     var_8 = r1;
201                     if ((r1 == 0x0) && (r0 < 0xdc << 0x1)) {
202                         r2 = var_4 + 0xffffffff8;
203                         lseek64(sign_extend_32(*(int16_t *)(r5 + 0xe)), 0x0, r2);
204                         read(sign_extend_32(*(int16_t *)(r5 + 0xe)), r6 + 0x0, 0x8);
205                         if (strncmp(r6 + 0x0, "NATISIGN" + 0x0, 0x8) != 0x0) {
206                             fclose(r5 + 0x0);
207                             r0 = 0x83 << 0x1;
208                         }
209                     }
210                 }
211             }
212         }
213     }
214 }
```

# 평양 2407 Legal Interface



# Pyongyang 2407

## Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147 package gov.no.media.natsign;
148
149 public class MnsNative
150 {
151     private static final boolean DBG = true;
152     public static final String TAG = "MnsNative";
153     static
154     {
155         System.loadLibrary("medianatsign");
156     }
157     public static native void getIMEIandIMSI(String paramString1, String paramString2);
158     public static native int getNatSignInfoLen(String paramString, int[] paramArrayOfInt);
159     public static native int isMagicCorrect(String paramString, int[] paramArrayOfInt);
160     public static native int isNatSignFile(String paramString, int[] paramArrayOfInt);
161     public static native void saveKeyToFile(byte[] paramArrayOfByte, int paramInt);
162     public static native void savePatternToFile(byte[] paramArrayOfByte, int paramInt);
163     public static native void saveSelfKeyToFile(byte[] paramArrayOfByte, int paramInt);
164 }
```

```
161     host = gethostbyname(server);
```

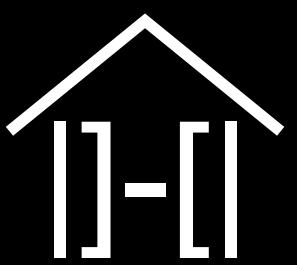
# gov.no.media.natsign API

32

```
165     exit(0);
166 }
```

[ @hackerfantastic ] [ @myhackerhouse ] [ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147     package gov.no.media.selfsign;
148
149     public class MSSNative
150     {
151         private static final boolean D = true;
152         public static final String TAG = "MSSNative";
153         static
154         {
155             System.loadLibrary("mediaselfsign");
156         }
157         public static native int CheckSelfSignFile(String paramString, int[] paramArrayOfInt);
158         public static native void SetMMSInfoToFile(String paramString, int[] paramArrayOfInt);
159         public static native int getSelfSignInfoLen(String paramString, int[] paramArrayOfInt);
160         public static native int isSelfResignedByPhone(String paramString, int[] paramArrayOfInt);
161         public static native int isSelfSignFile(String paramString, int[] paramArrayOfInt);
162         public static native int isSendedByMMS(String paramString, int[] paramArrayOfInt);
163         public static native int rsa2048_Sign(byte[] paramArrayOfByte1, byte[] paramArrayOfByte2,
164             int paramInt, byte[] paramArrayOfByte3);
165         public static native int rsa2048_Verify(byte[] paramArrayOfByte1, byte[] paramArrayOfByte2
166             , byte[] paramArrayOfByte3, int[] paramArrayOfInt);
167         public static native int saveSelfSignFile(String paramString, int paramInt);
168     }
169
170
171     host = gethostbyname(server);
172
173 }
```

# gov.no.media.selfsign API

33

```
exit(0);
```

[ @hackerfantastic ] [ @myhackerhouse ] [ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
```

# /system/app/RedFlag.apk

Every time a file is accessed, it is checked using Android native code for a signature to see if legal content. Self-signed files use RSA2048 to store a signature when files are created on the device, e.g. Camera photo.

The following files are used to to create SELFSIGN and to validate NATISIGN files.

- **/data/local/tmp/0.dat** - key file material
- **/data/local/tmp/selfsignkey.dat** - RSA2048 (self-signing key material)
- **/data/local/tmp/legalref.dat** - (IMEI & IMSI of the subscriber, created by saveImeiImsiToFile());
- **/data/local/tmp/pattern.dat** - (files and patterns to match, try to identify .apk or patterns)

kuts.ktr.RFService when “digital signature” checks fail, and a file is not self-signed or state approved, alert.

Every file / folder / removable media access goes through the signature / file checking system.

Any signature alerts are stored in “legals.db”. It stores record of self-signing file history such as photos taken. It captures details of signature infringements such as attempts to load movies.

```
161     host = gethostbyname(server);
162
163
164
```

Digital signature manipulation system 2.0

34

Pyongyang 2407  
Hacking North Korea



```
165     exit(0);
166 }
```

[ @hackerfantastic ] [ @myhackerhouse ] [ https://hacker.house ]

```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
```

## 전자서명조작체계 (Digital signature manipulation system) 2.0 (RedFlag)

```
./com/kptc/nv/CertKeyRW.smali
./kuts/ktr/RFService/RFReceiver.smali
./kuts/ktr/RFService/CopyUtils.smali
./kuts/ktr/RFService/RFService.smali
./kuts/ktr/RFService/LegalProvider.smali
./kuts/ktr/RFService/RFScanThread.smali
./kuts/ktr/RFService/BuildConfig.smali
./kuts/ktr/RFService/R.smali
./kuts/ktr/RFService/RFScreenThread.smali
./kuts/ktr/RFService/LegalContract.smali
./gov/no/media/natsign/MnsNative.smali
./gov/no/media/Sign.smali
./gov/no/media/selfsign/MssNative.smali

content://kuts.jsc.providers.legal
```

전자서명조작체계 *native Android components using cryptographic functions (RSA2048) to watermark all created documents and check/scan files for signatures.*

*Android content provider to SQLite (legals.db). Traceviewer can read/export contents.*



열람리력

*Broadcast Reciever MEDIA\_MOUNTED, BOOTUP & SHUTDOWN for 전자서명조작체계 RFService*

```
host = gethostbyname(server);
```

# Android kuts.ktr.RFService

35

```
exit(0);
```

[ @hackerfantastic ] [ @myhackerhouse ] [ https://hacker.house ]



Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server,port);
```

# Debugging enabled in some 전자서명조작체계 components

```
public static String[] Video_extensions = { "3gp", "amv", "avi", "avc", "f4v",
"flv", "mov", "mp4", "mpeg", "mpg", "mpeg4", "m1v", "wmv", "wma", "m2v", "mpv2",
"rm", "rmvb", "vob" };

public static String[] extensions = { "3g2", "3gp2", "3gpp", "aac", "ac3", "amr",
"ape", "ASF", "awb", "asx", "bmp", "bik", "cda", "csf", "cda", "cue", "dat",
"divx", "doc", "docx", "dts", "evo", "flac", "gif", "h1v", "htm", "html", "ifo",
"ivm", "jpeg", "jpg", "kpl", "m1a", "m2a", "m2p", "m2t", "m2ts", "m3u", "m4a",
"m4b", "m4p", "m4r", "m4v", "mid", "midi", "mka", "mkv", "mmf", "mp2", "mp2v",
"mp3", "mpa", "mpc", "mpe", "mod", "mts", "ofr", "ogg", "ogm", "pcx", "pdf",
"png", "ppt", "pmp", "pptx", "pss", "pva", "pls", "qpl", "qt", "ra", "ram", "rp",
"rpm", "rt", "rtf", "smf", "swf", "smi", "smil", "scm", "smpl", "tga", "tif",
"tpr", "tiff", "tp", "ts", "tta", "txt", "tod", "vp6", "wav", "wv", "wm", "wmp",
"webm", "xls", "xlsx", "xml" };

host = agethostbyname(server);
```

# RFScan Signature'd File Exts.

# Pyongyang 2407

## Hacking North Korea



36

exit(0)

[ @hackerfantastic

└ @myhackerhouse

[ <https://hacker.house> ]

```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
```

TraceViewer

FILES HISTORY CAPTURE HISTORY

Total : 12files

_id	path	size	date_modified	isvalid
1	/storage/sdcard0/DCIM/Camera/IMG_20140106_200922.jpg	2774753	1389038275104	1
2	/storage/sdcard0/DCIM/Camera/IMG_20140106_201131.jpg	3203553	1389038275321	1
3	/storage/sdcard0/DCIM/Camera/IMG_20140107_021105.jpg	2965729	1389038275406	1
4	/storage/sdcard0/DCIM/Camera/VID_20140107_031533.3gp	35604822	1389038275571	1
5	/storage/sdcard0/DCIM/Camera/VID_20140107_031813.3gp	44422147	1389038275702	1
6	/storage/sdcard0/DCIM/Camera/VID_20140107_034709.3gp	1205374	1389038275906	1
7	/storage/sdcard0/DCIM/Camera/IMG_20140107_034723.jpg	3439201	1389038276005	1
8	/storage/sdcard0/DCIM/Camera/IMG_20140107_034745.jpg	1945313	1389038276173	1
9	/storage/sdcard0/DCIM/Camera/IMG_20140107_034750.jpg	2046177	1389038276298	1
10	/storage/sdcard0/DCIM/Camera/IMG_20140115_041924.jpg	1008941	1390152767996	1
11	/storage/sdcard0/DCIM/Camera/IMG_20140115_041925.jpg	1033645	1390152768110	1
12	/storage/sdcard0/DCIM/Camera/IMG_20140115_041928.jpg	1001901	1390152768202	1

```
host = gethostbyname(server);
```

# TraceViewer (browser\_history) Pyongyang 2407 Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
```

Table: capture_list					
	_id	apk_name	captured_time	_data	isValid
1	1	KPubReader	138879847...	/data/data/kuts.ktr.RFService/files/capture_list/1.jpg	1
2	2	drozer Agent	139031109...	/data/data/kuts.ktr.RFService/files/capture_list/2.jpg	1
3	3	MX Player	139033393...	/data/data/kuts.ktr.RFService/files/capture_list/3.jpg	1
4	4	MX Player	138853880...	/data/data/kuts.ktr.RFService/files/capture_list/4.jpg	1
5	5	MX Player	138854384...	/data/data/kuts.ktr.RFService/files/capture_list/5.jpg	1
6	6	MX Player	138871374...	/data/data/kuts.ktr.RFService/files/capture_list/6.jpg	1
7	7	Video	138871674...	/data/data/kuts.ktr.RFService/files/capture_list/7.jpg	1
8	8	Video	138871704...	/data/data/kuts.ktr.RFService/files/capture_list/8.jpg	1
9	9	Video	138871734...	/data/data/kuts.ktr.RFService/files/capture_list/9.jpg	1
10	10	Video	138871754...	feh [1 of 11 - 5.ipa	files/capture
11	11	Video	138871754...	feh [1 of 11 - 6.ipa	files/capture
12	12	Video	138871754...	feh [1 of 11 - 7.ipa	files/capture
13	13	Video	138871754...	feh [1 of 11 - 8.ipa	files/capture
14	14	Video	138871754...	feh [1 of 11 - 9.ipa	files/capture
15	15	Video	138871754...	feh [1 of 11 - 10.ipa	files/capture
16	16	Video	138871754...	feh [1 of 11 - 11.ipa	files/capture
17	17	Video	138871754...	feh [1 of 11 - 12.ipa	files/capture

```
161     host = gethostbyname(server);
162
163
164
```

# TraceViewer (capture\_list)

38

```
exit(0);
```

[ @hackerfantastic ] [ @myhackerhouse ] [ https://hacker.house ]

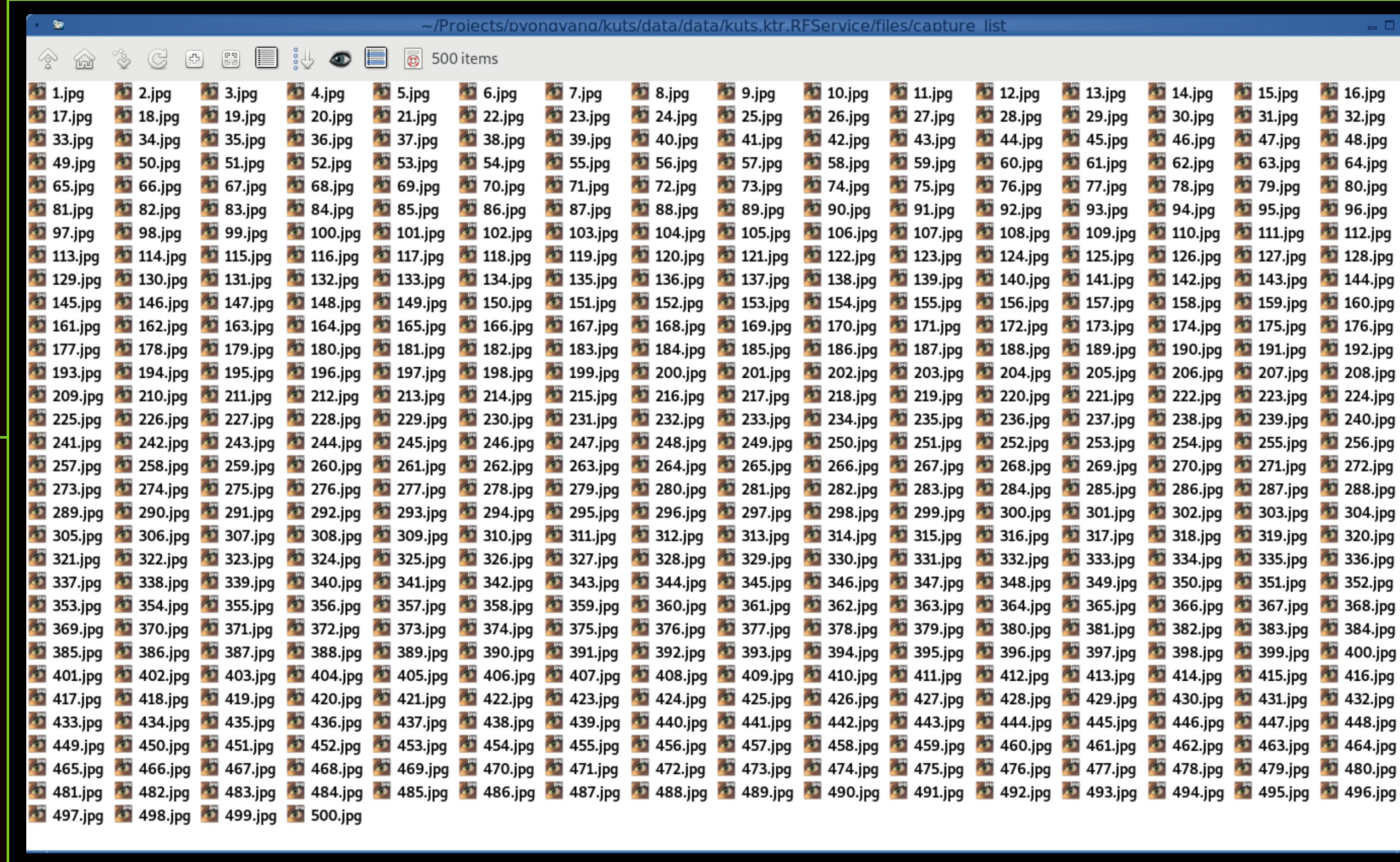


열람리액

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
```



*Files not self-signed or signed by the state, when accessed or viewed, are recorded in a database.*

**Screen shots are taken of the offending access.**

```
161     host = gethostbyname(server);
162
163
164
```

## RedFlag History (capture\_list)

39

```
exit(0);
```

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147 /프로그램자료/.KwangMyong/도움말.pdf
148
149
150
151
152
153
154
155
156
157
158 00000114E8 43 44 D8 25 84 A4 61 5D 42 CF E7 CD 40 01 00 00 4E 41 54 49 53 49 .CD.%..a]B...@...NATISI
159 0000012B47 4E
160
161 host = gethostbyname(server);
162
163 Example of NATISIGN file
164 40
165 exit(0);
166 }
```

Pyongyang 2407  
Hacking North Korea



# /DCIM/Camera/IMG\_2014\_0103\_085922.jpg

**Watermarking contains IMEI/IMSI in the process. SELFSIGN uses a RSA2048 key process along with legalref.dat. Its appended to any file created on the device.**

```
D/gov_sign( 2070): -----.jpg-----4----  
D/gov_sign( 2070): get_flength : file name = /storage/sdcard0/DCIM/Camera/  
IMG_20140103_085922.jpg, fSize = 820065, POSTFIX_LEN = 0  
D/gov_sign( 2070): MnsNative isNatSignFile : file name = /storage/sdcard0/DCIM/Camera/  
IMG_20140103_085922.jpg, result = 264  
E/MediaSelfSign( 2070): Call isSelfSignFile()----- successed!  
E/MediaSelfSign( 2070): MSSMediaVerifyFileByPath()-----! return 0  
E/MediaSelfSign( 2070): MSSMediaVerifyFileByPath()-----! return 0  
E/MediaSelfSign( 2070): isSelfSignFile()-----successed! Err = 0, return 1
```

```
host = gethostbyname(server):
```

## Example of SELFSIGN file

41

```
exit(0);
```

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```

144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147     ; ===== BEGINNING OF PROCEDURE =====
148
149     ; Variables:
150     ;   var_18: int32_t, -24
151
152     isNatSignFile:
153
154     push    {r4, r5, lr}
155     ldr     r1, [r0]
156     movs   r3, #0xa9
157     lsls   r3, r3, #0x2
158     sub    sp, #0xc
159     ldr     r3, [r1, r3]
160     adds   r1, r2, #0x0
161     movs   r2, #0x0
162     blx    r3
163     adds   r5, r0, #0x0
164
165     bl     verifyFileSign | ; verifyFileSign
166     ldr    r1, =0x3e52 ; dword_19ac,0x3e52
167     ldr    r2, =0x3e9c ; dword_19b0,0x3e9c
168
169     adds   r4, r0, #0x0
170     str    r0, [sp, #0x18 + var_18]
171
172     adds   r3, r5, #0x0
173
174     add    r1, pc ; "gov_sign"
175     add    r2, pc ; "MnsNative isNatSignFile : file name = %s, result = %d"
176
177     movs   r0, #0x3
178
179     blx    __android_log_print@PLT ; __android_log_print
180
181     add    sp, #0xc
182     adds   r0, r4, #0x0
183
184     pop    {r4, r5, pc}
185
186     ; endp
187
188
189     host = gethostbyname(server);
190
191     exit(0);
192 }

```

평양 2407 libmedianatsign.so

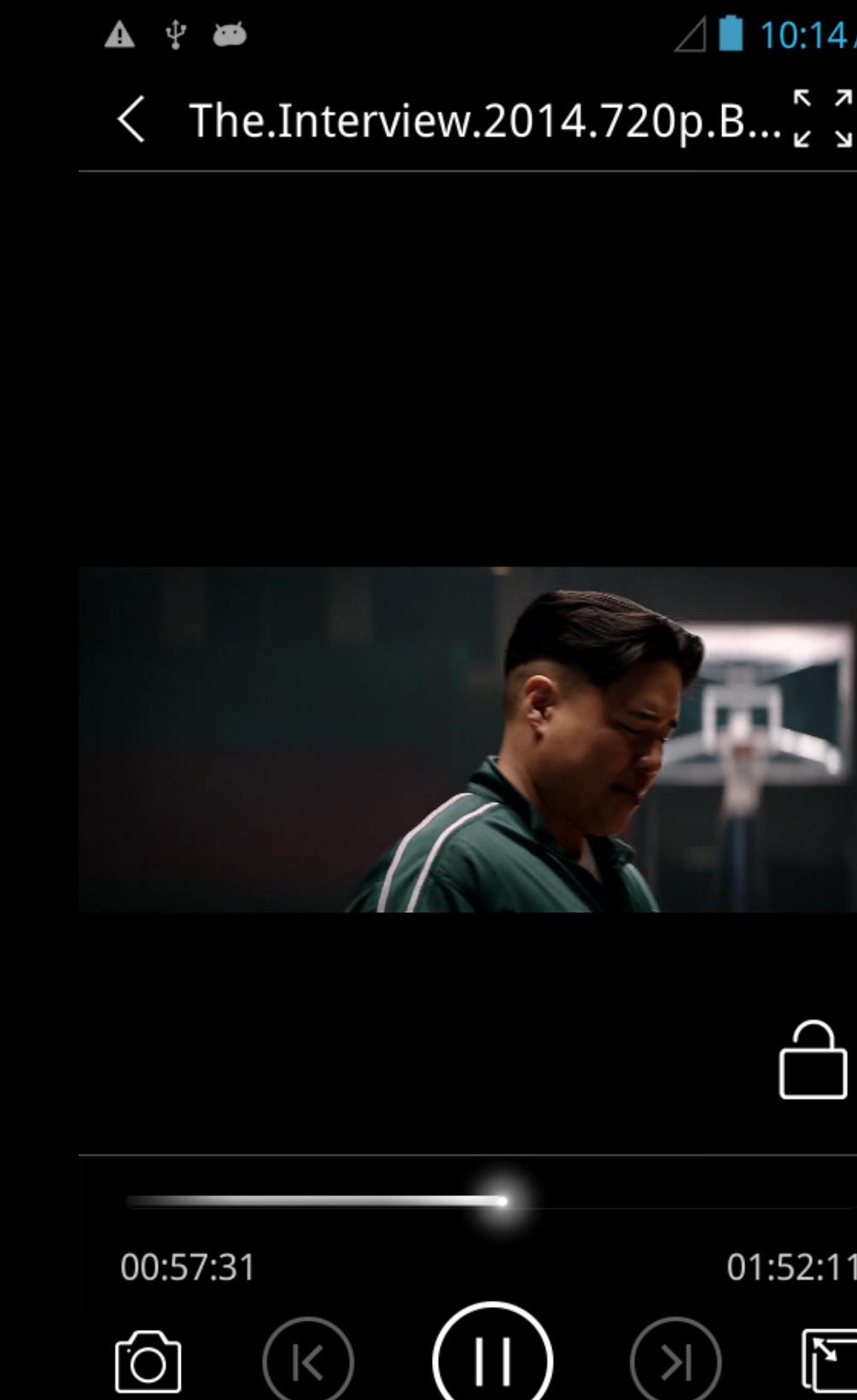
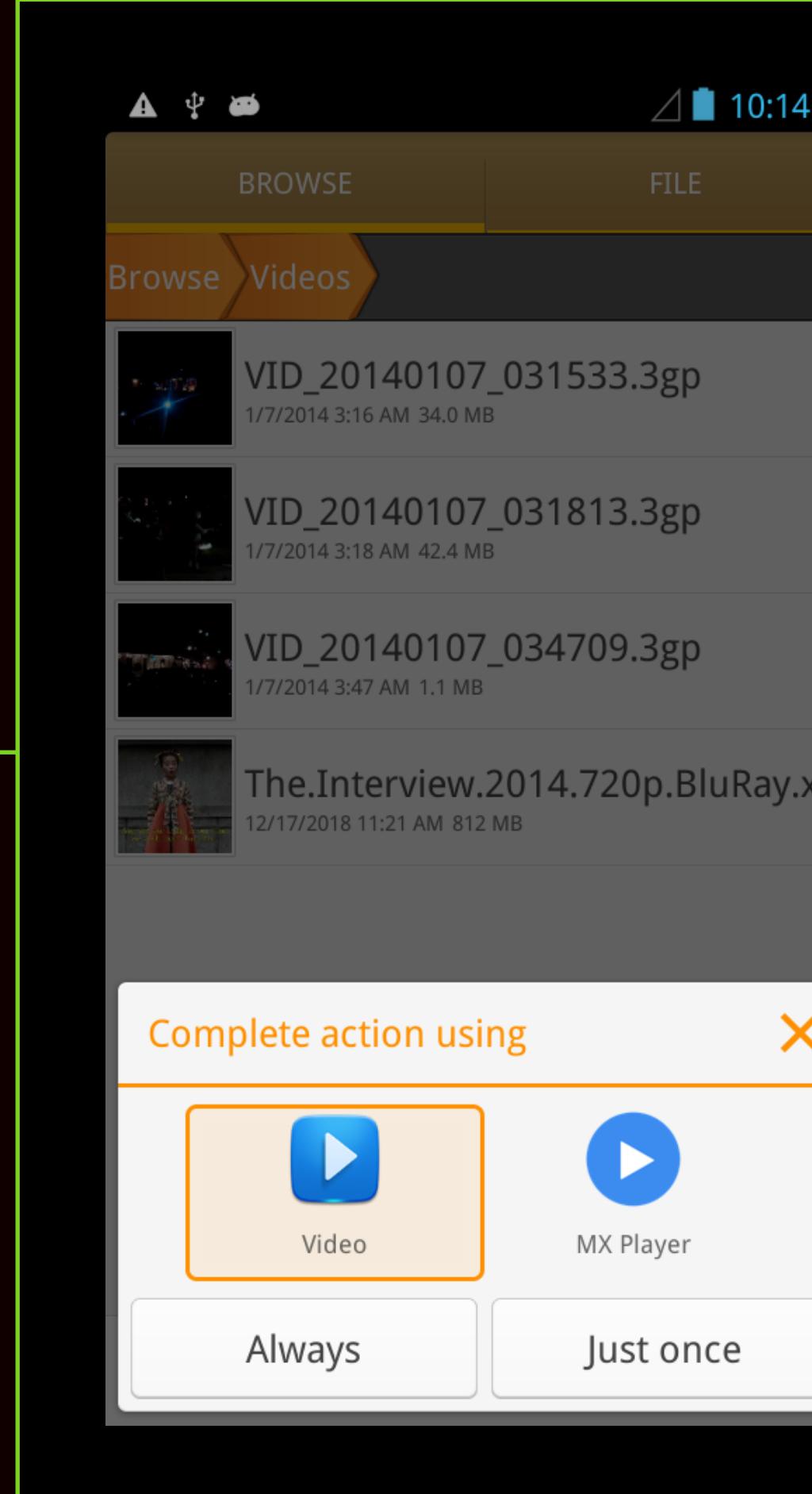
42

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147     isNatSignFile:
148
149     00001978    push   {r4, r5, lr}
150     0000197a    ldr    r1, [r0]
151     0000197c    movs   r3, #0xa9
152     0000197e    lsls   r3, r3, #0x2
153     00001980    sub    sp, #0xc
154     00001982    ldr    r3, [r1, r3]
155     00001984    adds   r1, r2, #0x0
156     00001986    movs   r2, #0x0
157     00001988    blx   r3
158     0000198a    adds   r5, r0, #0x0
159     0000198c    movs   r0, #0x1
160
161     0000198e    nop
162     00001990    ldr    r1, =0x3e52
163     00001992    ldr    r2, =0x3e9c
164     00001994    movs   r4, #0x1
165     00001996    str    r0, [sp]
166     00001998    adds   r3, r5, #0x0
167     0000199a    add    r1, pc
168     0000199c    add    r2, pc
169     0000199e    movs   r0, #0x3
170     000019a0    blx   __android_log_print@PLT
171     000019a4    add    sp, #0xc
172     000019a6    movs   r0, #0x1
173     000019a8    pop    {r4, r5, pc}
174
175     host = gethostbyname(server);
176
177     if(host == NULL)
178     {
179         perror("gethostbyname error");
180         exit(1);
181     }
182
183     if((host->h_length != sizeof(struct hostent)) ||
184        (host->h_name != server) ||
185        (host->h_aliases[0] != NULL))
186     {
187         perror("gethostbyname error");
188         exit(1);
189     }
190
191     if((host->h_addrtype != AF_INET) ||
192        (host->h_length != sizeof(struct in_addr)))
193     {
194         perror("gethostbyname error");
195         exit(1);
196     }
197
198     if((host->h_addr_list[0] == NULL) ||
199        (host->h_addr_list[1] == NULL))
200     {
201         perror("gethostbyname error");
202         exit(1);
203     }
204
205     if((host->h_addr_list[0] != NULL) ||
206        (host->h_addr_list[1] != NULL))
207     {
208         struct in_addr *addr;
209
210         if((addr = (struct in_addr *)host->h_addr_list[0]) == NULL)
211             exit(1);
212
213         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
214             break;
215     }
216
217     if((host->h_addr_list[1] != NULL) ||
218        (host->h_addr_list[2] != NULL))
219     {
220         struct in_addr *addr;
221
222         if((addr = (struct in_addr *)host->h_addr_list[1]) == NULL)
223             exit(1);
224
225         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
226             break;
227     }
228
229     if((host->h_addr_list[2] != NULL) ||
230        (host->h_addr_list[3] != NULL))
231     {
232         struct in_addr *addr;
233
234         if((addr = (struct in_addr *)host->h_addr_list[2]) == NULL)
235             exit(1);
236
237         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
238             break;
239     }
240
241     if((host->h_addr_list[3] != NULL) ||
242        (host->h_addr_list[4] != NULL))
243     {
244         struct in_addr *addr;
245
246         if((addr = (struct in_addr *)host->h_addr_list[3]) == NULL)
247             exit(1);
248
249         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
250             break;
251     }
252
253     if((host->h_addr_list[4] != NULL) ||
254        (host->h_addr_list[5] != NULL))
255     {
256         struct in_addr *addr;
257
258         if((addr = (struct in_addr *)host->h_addr_list[4]) == NULL)
259             exit(1);
260
261         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
262             break;
263     }
264
265     if((host->h_addr_list[5] != NULL) ||
266        (host->h_addr_list[6] != NULL))
267     {
268         struct in_addr *addr;
269
270         if((addr = (struct in_addr *)host->h_addr_list[5]) == NULL)
271             exit(1);
272
273         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
274             break;
275     }
276
277     if((host->h_addr_list[6] != NULL) ||
278        (host->h_addr_list[7] != NULL))
279     {
280         struct in_addr *addr;
281
282         if((addr = (struct in_addr *)host->h_addr_list[6]) == NULL)
283             exit(1);
284
285         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
286             break;
287     }
288
289     if((host->h_addr_list[7] != NULL) ||
290        (host->h_addr_list[8] != NULL))
291     {
292         struct in_addr *addr;
293
294         if((addr = (struct in_addr *)host->h_addr_list[7]) == NULL)
295             exit(1);
296
297         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
298             break;
299     }
300
301     if((host->h_addr_list[8] != NULL) ||
302        (host->h_addr_list[9] != NULL))
303     {
304         struct in_addr *addr;
305
306         if((addr = (struct in_addr *)host->h_addr_list[8]) == NULL)
307             exit(1);
308
309         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
310             break;
311     }
312
313     if((host->h_addr_list[9] != NULL) ||
314        (host->h_addr_list[10] != NULL))
315     {
316         struct in_addr *addr;
317
318         if((addr = (struct in_addr *)host->h_addr_list[9]) == NULL)
319             exit(1);
320
321         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
322             break;
323     }
324
325     if((host->h_addr_list[10] != NULL) ||
326        (host->h_addr_list[11] != NULL))
327     {
328         struct in_addr *addr;
329
330         if((addr = (struct in_addr *)host->h_addr_list[10]) == NULL)
331             exit(1);
332
333         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
334             break;
335     }
336
337     if((host->h_addr_list[11] != NULL) ||
338        (host->h_addr_list[12] != NULL))
339     {
340         struct in_addr *addr;
341
342         if((addr = (struct in_addr *)host->h_addr_list[11]) == NULL)
343             exit(1);
344
345         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
346             break;
347     }
348
349     if((host->h_addr_list[12] != NULL) ||
350        (host->h_addr_list[13] != NULL))
351     {
352         struct in_addr *addr;
353
354         if((addr = (struct in_addr *)host->h_addr_list[12]) == NULL)
355             exit(1);
356
357         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
358             break;
359     }
360
361     if((host->h_addr_list[13] != NULL) ||
362        (host->h_addr_list[14] != NULL))
363     {
364         struct in_addr *addr;
365
366         if((addr = (struct in_addr *)host->h_addr_list[13]) == NULL)
367             exit(1);
368
369         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
370             break;
371     }
372
373     if((host->h_addr_list[14] != NULL) ||
374        (host->h_addr_list[15] != NULL))
375     {
376         struct in_addr *addr;
377
378         if((addr = (struct in_addr *)host->h_addr_list[14]) == NULL)
379             exit(1);
380
381         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
382             break;
383     }
384
385     if((host->h_addr_list[15] != NULL) ||
386        (host->h_addr_list[16] != NULL))
387     {
388         struct in_addr *addr;
389
390         if((addr = (struct in_addr *)host->h_addr_list[15]) == NULL)
391             exit(1);
392
393         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
394             break;
395     }
396
397     if((host->h_addr_list[16] != NULL) ||
398        (host->h_addr_list[17] != NULL))
399     {
400         struct in_addr *addr;
401
402         if((addr = (struct in_addr *)host->h_addr_list[16]) == NULL)
403             exit(1);
404
405         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
406             break;
407     }
408
409     if((host->h_addr_list[17] != NULL) ||
410        (host->h_addr_list[18] != NULL))
411     {
412         struct in_addr *addr;
413
414         if((addr = (struct in_addr *)host->h_addr_list[17]) == NULL)
415             exit(1);
416
417         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
418             break;
419     }
420
421     if((host->h_addr_list[18] != NULL) ||
422        (host->h_addr_list[19] != NULL))
423     {
424         struct in_addr *addr;
425
426         if((addr = (struct in_addr *)host->h_addr_list[18]) == NULL)
427             exit(1);
428
429         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
430             break;
431     }
432
433     if((host->h_addr_list[19] != NULL) ||
434        (host->h_addr_list[20] != NULL))
435     {
436         struct in_addr *addr;
437
438         if((addr = (struct in_addr *)host->h_addr_list[19]) == NULL)
439             exit(1);
440
441         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
442             break;
443     }
444
445     if((host->h_addr_list[20] != NULL) ||
446        (host->h_addr_list[21] != NULL))
447     {
448         struct in_addr *addr;
449
450         if((addr = (struct in_addr *)host->h_addr_list[20]) == NULL)
451             exit(1);
452
453         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
454             break;
455     }
456
457     if((host->h_addr_list[21] != NULL) ||
458        (host->h_addr_list[22] != NULL))
459     {
460         struct in_addr *addr;
461
462         if((addr = (struct in_addr *)host->h_addr_list[21]) == NULL)
463             exit(1);
464
465         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
466             break;
467     }
468
469     if((host->h_addr_list[22] != NULL) ||
470        (host->h_addr_list[23] != NULL))
471     {
472         struct in_addr *addr;
473
474         if((addr = (struct in_addr *)host->h_addr_list[22]) == NULL)
475             exit(1);
476
477         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
478             break;
479     }
480
481     if((host->h_addr_list[23] != NULL) ||
482        (host->h_addr_list[24] != NULL))
483     {
484         struct in_addr *addr;
485
486         if((addr = (struct in_addr *)host->h_addr_list[23]) == NULL)
487             exit(1);
488
489         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
490             break;
491     }
492
493     if((host->h_addr_list[24] != NULL) ||
494        (host->h_addr_list[25] != NULL))
495     {
496         struct in_addr *addr;
497
498         if((addr = (struct in_addr *)host->h_addr_list[24]) == NULL)
499             exit(1);
500
501         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
502             break;
503     }
504
505     if((host->h_addr_list[25] != NULL) ||
506        (host->h_addr_list[26] != NULL))
507     {
508         struct in_addr *addr;
509
510         if((addr = (struct in_addr *)host->h_addr_list[25]) == NULL)
511             exit(1);
512
513         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
514             break;
515     }
516
517     if((host->h_addr_list[26] != NULL) ||
518        (host->h_addr_list[27] != NULL))
519     {
520         struct in_addr *addr;
521
522         if((addr = (struct in_addr *)host->h_addr_list[26]) == NULL)
523             exit(1);
524
525         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
526             break;
527     }
528
529     if((host->h_addr_list[27] != NULL) ||
530        (host->h_addr_list[28] != NULL))
531     {
532         struct in_addr *addr;
533
534         if((addr = (struct in_addr *)host->h_addr_list[27]) == NULL)
535             exit(1);
536
537         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
538             break;
539     }
540
541     if((host->h_addr_list[28] != NULL) ||
542        (host->h_addr_list[29] != NULL))
543     {
544         struct in_addr *addr;
545
546         if((addr = (struct in_addr *)host->h_addr_list[28]) == NULL)
547             exit(1);
548
549         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
550             break;
551     }
552
553     if((host->h_addr_list[29] != NULL) ||
554        (host->h_addr_list[30] != NULL))
555     {
556         struct in_addr *addr;
557
558         if((addr = (struct in_addr *)host->h_addr_list[29]) == NULL)
559             exit(1);
560
561         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
562             break;
563     }
564
565     if((host->h_addr_list[30] != NULL) ||
566        (host->h_addr_list[31] != NULL))
567     {
568         struct in_addr *addr;
569
570         if((addr = (struct in_addr *)host->h_addr_list[30]) == NULL)
571             exit(1);
572
573         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
574             break;
575     }
576
577     if((host->h_addr_list[31] != NULL) ||
578        (host->h_addr_list[32] != NULL))
579     {
580         struct in_addr *addr;
581
582         if((addr = (struct in_addr *)host->h_addr_list[31]) == NULL)
583             exit(1);
584
585         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
586             break;
587     }
588
589     if((host->h_addr_list[32] != NULL) ||
590        (host->h_addr_list[33] != NULL))
591     {
592         struct in_addr *addr;
593
594         if((addr = (struct in_addr *)host->h_addr_list[32]) == NULL)
595             exit(1);
596
597         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
598             break;
599     }
510
511     if((host->h_addr_list[33] != NULL) ||
512        (host->h_addr_list[34] != NULL))
513     {
514         struct in_addr *addr;
515
516         if((addr = (struct in_addr *)host->h_addr_list[33]) == NULL)
517             exit(1);
518
519         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
520             break;
521     }
522
523     if((host->h_addr_list[34] != NULL) ||
524        (host->h_addr_list[35] != NULL))
525     {
526         struct in_addr *addr;
527
528         if((addr = (struct in_addr *)host->h_addr_list[34]) == NULL)
529             exit(1);
530
531         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
532             break;
533     }
534
535     if((host->h_addr_list[35] != NULL) ||
536        (host->h_addr_list[36] != NULL))
537     {
538         struct in_addr *addr;
539
540         if((addr = (struct in_addr *)host->h_addr_list[35]) == NULL)
541             exit(1);
542
543         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
544             break;
545     }
546
547     if((host->h_addr_list[36] != NULL) ||
548        (host->h_addr_list[37] != NULL))
549     {
550         struct in_addr *addr;
551
552         if((addr = (struct in_addr *)host->h_addr_list[36]) == NULL)
553             exit(1);
554
555         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
556             break;
557     }
558
559     if((host->h_addr_list[37] != NULL) ||
560        (host->h_addr_list[38] != NULL))
561     {
562         struct in_addr *addr;
563
564         if((addr = (struct in_addr *)host->h_addr_list[37]) == NULL)
565             exit(1);
566
567         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
568             break;
569     }
570
571     if((host->h_addr_list[38] != NULL) ||
572        (host->h_addr_list[39] != NULL))
573     {
574         struct in_addr *addr;
575
576         if((addr = (struct in_addr *)host->h_addr_list[38]) == NULL)
577             exit(1);
578
579         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
580             break;
581     }
582
583     if((host->h_addr_list[39] != NULL) ||
584        (host->h_addr_list[40] != NULL))
585     {
586         struct in_addr *addr;
587
588         if((addr = (struct in_addr *)host->h_addr_list[39]) == NULL)
589             exit(1);
590
591         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
592             break;
593     }
594
595     if((host->h_addr_list[40] != NULL) ||
596        (host->h_addr_list[41] != NULL))
597     {
598         struct in_addr *addr;
599
600         if((addr = (struct in_addr *)host->h_addr_list[40]) == NULL)
601             exit(1);
602
603         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
604             break;
605     }
606
607     if((host->h_addr_list[41] != NULL) ||
608        (host->h_addr_list[42] != NULL))
609     {
610         struct in_addr *addr;
611
612         if((addr = (struct in_addr *)host->h_addr_list[41]) == NULL)
613             exit(1);
614
615         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
616             break;
617     }
618
619     if((host->h_addr_list[42] != NULL) ||
620        (host->h_addr_list[43] != NULL))
621     {
622         struct in_addr *addr;
623
624         if((addr = (struct in_addr *)host->h_addr_list[42]) == NULL)
625             exit(1);
626
627         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
628             break;
629     }
630
631     if((host->h_addr_list[43] != NULL) ||
632        (host->h_addr_list[44] != NULL))
633     {
634         struct in_addr *addr;
635
636         if((addr = (struct in_addr *)host->h_addr_list[43]) == NULL)
637             exit(1);
638
639         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
640             break;
641     }
642
643     if((host->h_addr_list[44] != NULL) ||
644        (host->h_addr_list[45] != NULL))
645     {
646         struct in_addr *addr;
647
648         if((addr = (struct in_addr *)host->h_addr_list[44]) == NULL)
649             exit(1);
650
651         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
652             break;
653     }
654
655     if((host->h_addr_list[45] != NULL) ||
656        (host->h_addr_list[46] != NULL))
657     {
658         struct in_addr *addr;
659
660         if((addr = (struct in_addr *)host->h_addr_list[45]) == NULL)
661             exit(1);
662
663         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
664             break;
665     }
666
667     if((host->h_addr_list[46] != NULL) ||
668        (host->h_addr_list[47] != NULL))
669     {
670         struct in_addr *addr;
671
672         if((addr = (struct in_addr *)host->h_addr_list[46]) == NULL)
673             exit(1);
674
675         if((addr->s_addr != 0) && (addr->s_addr != 0xffffffff))
676             break;
677     }
678
679     if((host->h_addr_list[47] != NULL) ||
680        (host->h_addr_list[48] != NULL))
681     {
682         struct in_addr *addr;
683
684         if((addr = (struct in_addr *)host->h_addr_list[47])
```

```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161     host = gethostbyname(server);
162
163
164
```



Exploited gov\_sign logic, DPRK government approved media

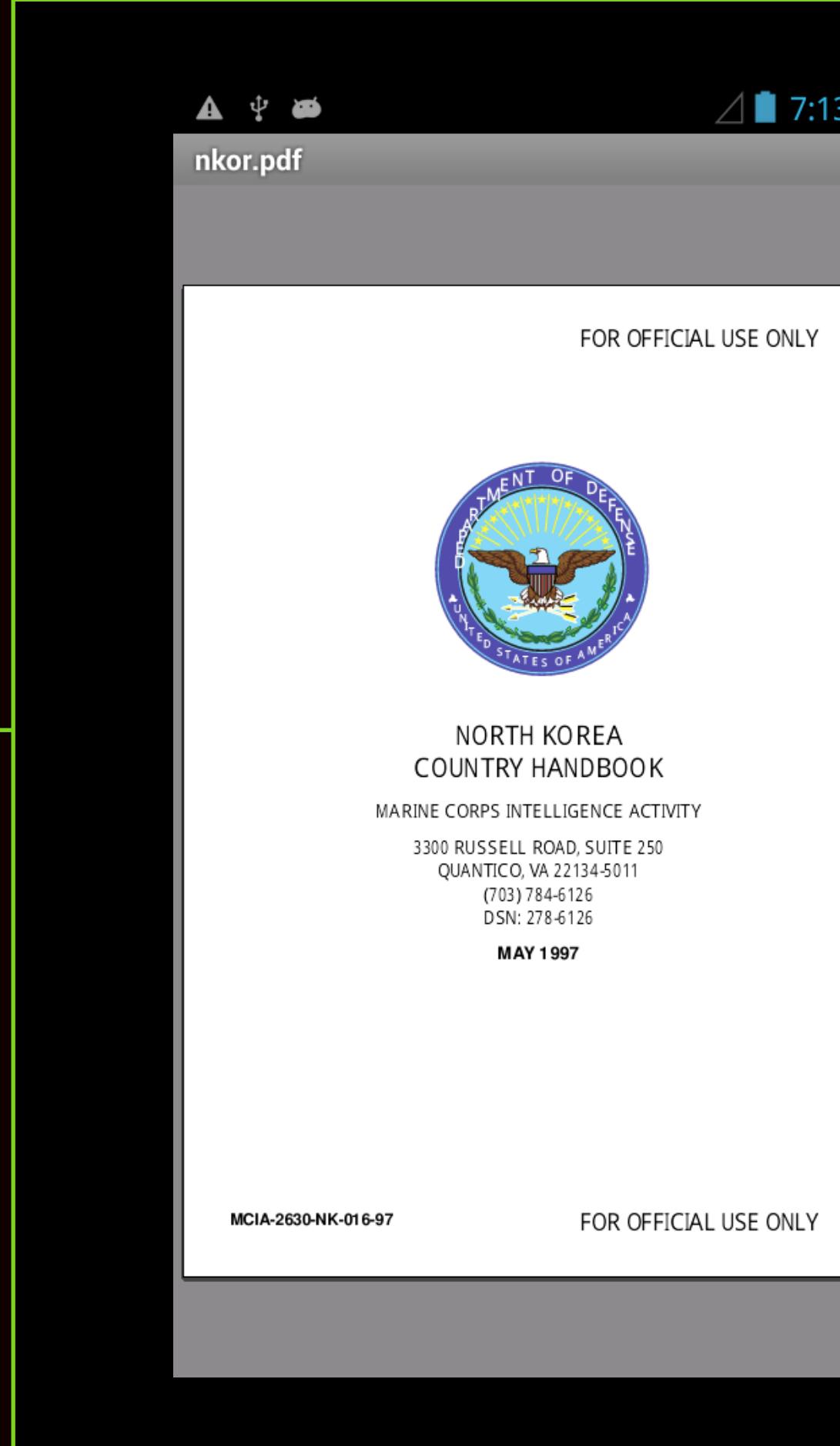
Phone now treats all media as Nat\_Sign\_File bypassing 전자서명조작체계 validation

Can load any PDF, MP4, JPG etc - phone treats it as DPRK approved state media, no longer logs or alerts on use.

전자서명조작체계 gov\_sign exploit Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
```



I watched The.Interview.2014 on 평양 2407 device.

I loaded documents of interesting facts about North Korea onto the device.

“adb logcat” shows all files are treated as Nat\_Sign\_File.

Patch files & jailbreak tool - [https://github.com/hackerhouse-opensource/pyongyang\\_2407](https://github.com/hackerhouse-opensource/pyongyang_2407)

```
host = gethostbyname(server):
```

전자서명조작체계 Nat\_Sign\_File.

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161     host = gethostbyname(server);
162
163
164
165     exit(0);
166 }
```

# MT6582 & MT6627

## com.mediatek.bluetooth

MT6627 is the mediatek companion RF chipset.

Bluetooth, GPS, FM-Radio & 802.11 **ONLY**

<https://android.googlesource.com/kernel MEDIATEK/+refs>  
drivers/misc/mediatek/conn\_soc/common/core/wmt\_\*

- **Bluetooth**
  - Bluetooth specification v2.1+EDR
  - Bluetooth specification 3.0+HS compliance
  - Bluetooth v4.0 Low Energy (LE)
  - Integrated PA with 10dBm (class 1) transmit power and Balun
  - Rx sensitivity: GFSK -95dBm, DQPSK -94dBm, 8-DPSK -88dBm
  - Best-in-class BT/Wi-Fi coexistence performance
  - Up to 4 piconets simultaneously with background inquiry/page scan
  - Supports Scatternet
  - Packet loss concealment (PLC) function for better voice quality
  - Low-power scan function to reduce the power consumption in scan modes

# MEDIATEK MT6627 RF CHIP

Pyongyang 2407  
Hacking North Korea



```

144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
[bluetooth]# info E6:68:46:EE:41:F2
Device E6:68:46:EE:41:F2
    Name: 평양 2407
    Alias: 평양 2407
    Class: 0x5a020c
    Icon: phone
    Paired: yes
    Trusted: yes
    Blocked: no
    Connected: no
    LegacyPairing: no
    UUID: Dialup Networking          (00001103-0000-1000-8000-00805f9b34fb)
    UUID: OBEX Object Push          (00001105-0000-1000-8000-00805f9b34fb)
    UUID: Audio Source              (0000110a-0000-1000-8000-00805f9b34fb)
    UUID: A/V Remote Control Target (0000110c-0000-1000-8000-00805f9b34fb)
    UUID: Advanced Audio Distribu.. (0000110d-0000-1000-8000-00805f9b34fb)
    UUID: Headset AG                (00001112-0000-1000-8000-00805f9b34fb)
    UUID: NAP                      (00001116-0000-1000-8000-00805f9b34fb)
    UUID: GN                       (00001117-0000-1000-8000-00805f9b34fb)
    UUID: Handsfree Audio Gateway   (0000111f-0000-1000-8000-00805f9b34fb)
    UUID: Phonebook Access Server   (0000112f-0000-1000-8000-00805f9b34fb)
    UUID: PnP Information           (00001200-0000-1000-8000-00805f9b34fb)
    Modalias: bluetooth:v0046p0802d0903
160
161     host = gethostbyname(server);
162
163     평양 2407 Bluetooth Profile
164

```

```

[NEW] Device E6:68:46:EE:41:F2 평양 2407
[bluetooth]# devices
Device E6:68:46:EE:41:F2 평양 2407
[bluetooth]# connect E6:68:46:EE:41:F2
Attempting to connect to E6:68:46:EE:41:F2
[CHG] Device E6:68:46:EE:41:F2 Connected: yes
[평양 2407]# connect E6:68:46:EE:41:F2
Attempting to connect to E6:68:46:EE:41:F2
Connection successful
[평양 2407]# devices
[CHG] Device E6:68:46:EE:41:F2 Connected: no

```

# 평양 2407 Bluetooth Profile

47

exit(0);

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161     host = gethostbyname(server);
162
163
164
```

# Bluetooth Attacks

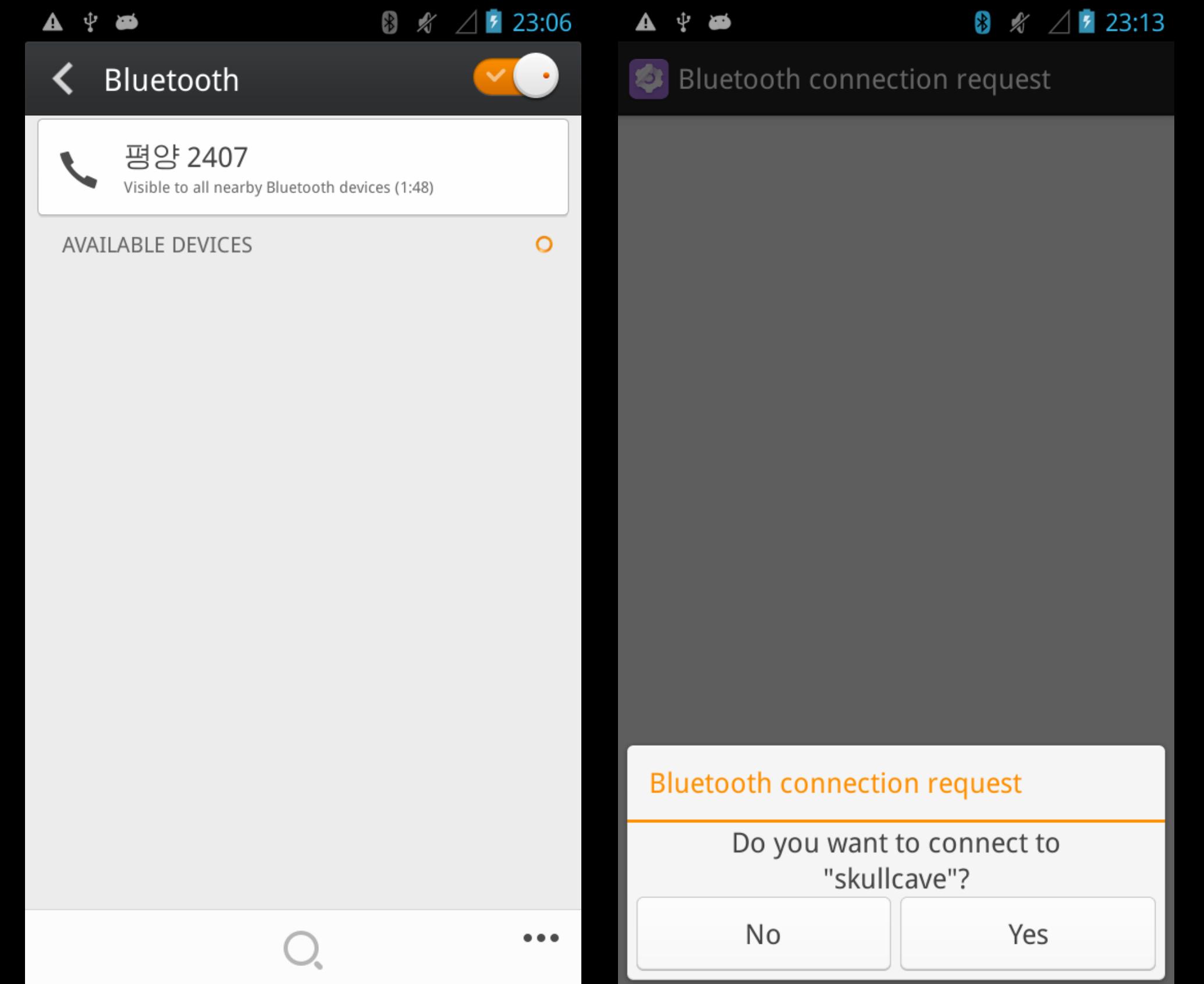
## 평양 2407 file transfer

CVE-2017-0781 - CVE-2017-785 - Android Blueborne(?)

MediaTek wrote own Bluetooth HAL/interface for MT6627.

Once a bluetooth device is paired, connections are always accepted from that device. Entering “no” to “Bluetooth connection request” still allows connection regardless until user forcefully selects “Unpair”

Default Bluetooth name is “평양 2407” discoverable for 2 minutes on activation and user forced scanning only.



# Bluetooth Attacks

48

```
exit(0);
```

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



```
bzero(&(sa.sin_zero),8);
printf("[ connecting to %s %d/tcp\n",server.port);
```

The following North Korean providers are known to have been active.

- Koryolink (고려링크) 467/05
- Kang Song NET 467/06
- Byol (별) Star



Koryolink SIM cards permitted to buy for foreigners. +850 0 191 xxxx

North Korea - KP [ edit ]						
MCC	MNC	Brand	Operator	Status	Bands (MHz)	References and notes
467	05	Koryolink	Cheo Technology Jv Company	Operational	UMTS 2100	for foreigners
467	06	Koryolink	Cheo Technology Jv Company	Operational	UMTS 2100	for DPRK citizens
467	193	SunNet	Korea Posts and Telecommunications Corporation	Not operational	GSM 900	

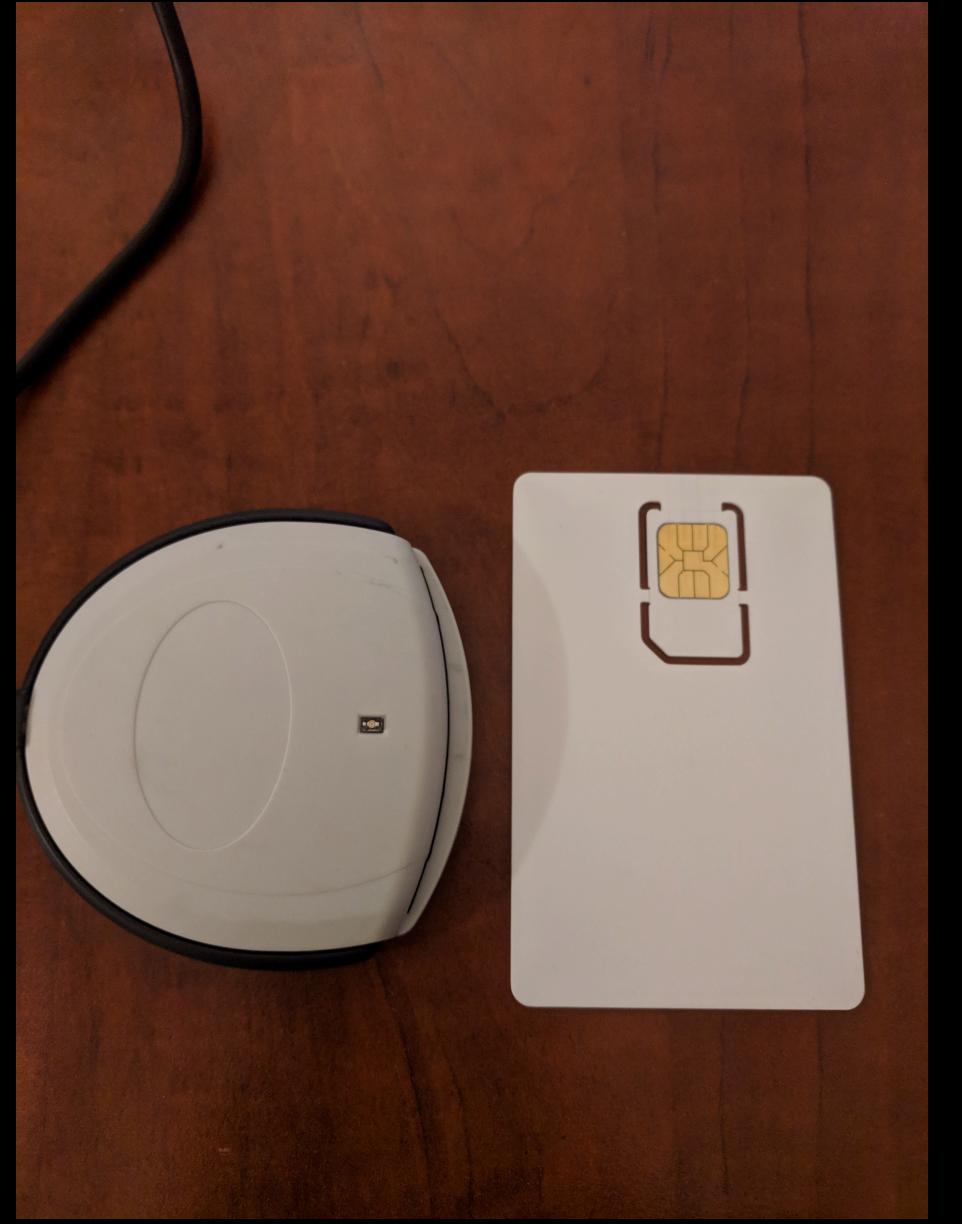
```
host = gethostbyname(server);
```

## Network Infrastructure (MCC)

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147     ./pySim-prog.py -p1 -t grcardsim -n Koryolink -i 4670600000000000 -s 000000000000000000000000
148     Insert card now (or CTRL-C to cancel)
149     Generated card parameters :
150     > Name      : Koryolink
151     > SMSP      : e1fffffffffffff0581005155f5fffffff000000
152     > ICCID     : 00000000000000000000
153     > MCC/MNC   : 467/6
154     > IMSI      : 4670600000000000
155     > Ki        : 12db6bfdd5cb8c8af844a3b6b65a9f49
156     > OPC       : ef11dc6f2fdc9aed6bd16c2fd28d4f3d
157     > ACC       : None
158
159     Programming ...
160     Done !
```



Phone can send MMS.... no SIM error.

Clone Kang Song NET SIM to test with...

Works to test MMS and other network functions...

Kang Song NET is for DPRK citizens only..

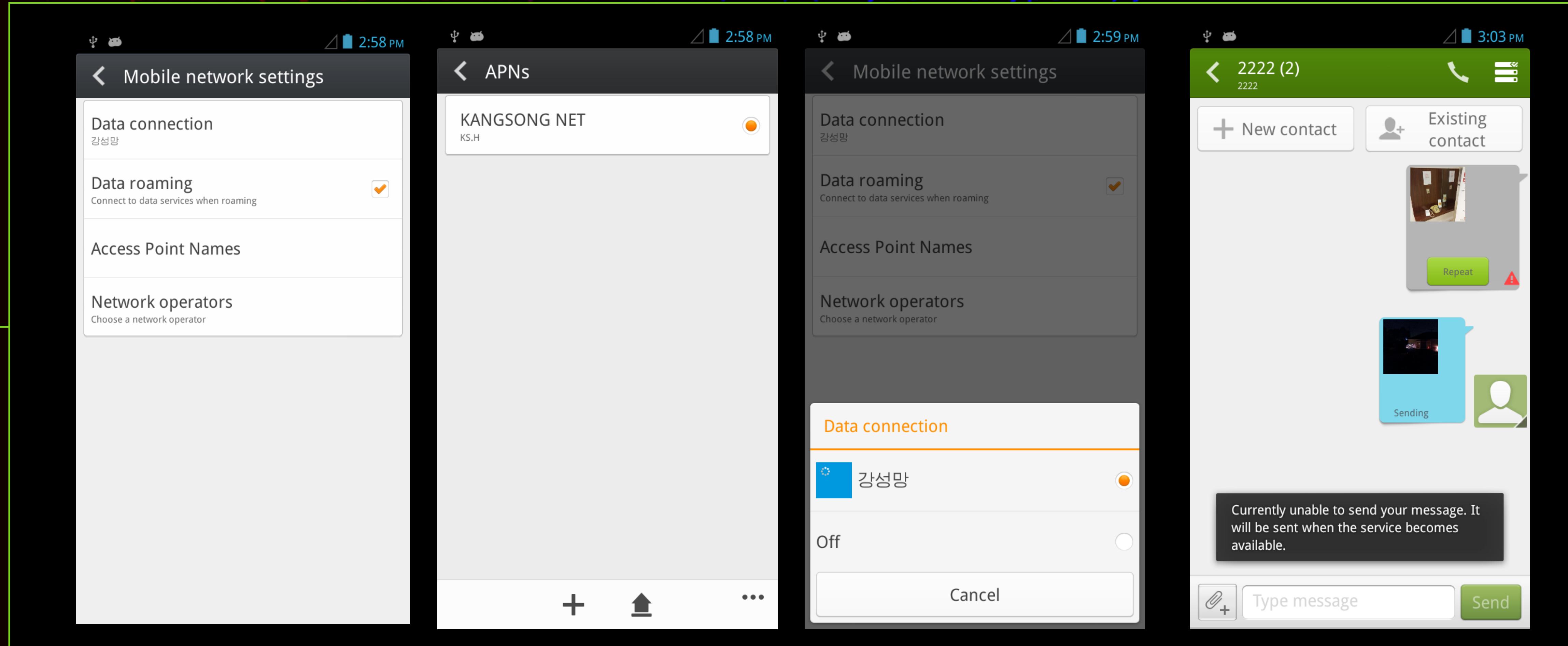
```
161     host = gethostbyname(server):
```

## Over-The-Air Network Attacks

Pyongyang 2407  
Hacking North Korea



```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
```



```
host = gethostbyname(server);
```

# Kang Song Net APN details

51

```
exit(0);
```

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

Pyongyang 2407  
Hacking North Korea



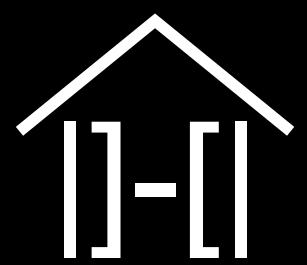
- **MediaTek bugdoors** - features for hacking/access
- **Red Flag** Digital Watermarking & Signature System
- Limited **USB/Bluetooth/MMS/WiFi** connectivity
- Digital media is state controlled inside **North Korea**
- **Exploits** required for unauthorised digital media use
- Mobile technology is supplied by **China**
- No platform diversity, attack techniques are portable

## Summary

52



Pyongyang 2407  
Hacking North Korea



exit(0);

[ @hackerfantastic ]

[ @myhackerhouse ]

[ https://hacker.house ]

```
144     bzero(&(sa.sin_zero),8);
145     printf("[ connecting to %s %d/tcp\n",server.port);
146
147
148
149
150
151
152
153
```

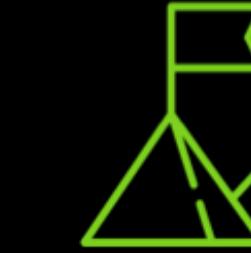
# Hands-on Hacking™

Learn practical ethical hacking skills online with Hacker House

<https://hacker.house/training>



158 Exclusive forum access



159 Learn at your pace



160 Download the labs

```
161 host = gethostbyname(server);
162
163
164
```

## Hacker House Online Training

Pyongyang 2407  
Hacking North Korea



53

```
165 exit(0);
166 }
```

@hackerfantastic [ @myhackerhouse ] [ https://hacker.house ]

```
144     bzero(&(sa.sin_zero),8);  
145     printf("[ connecting to %s %d/tcp\n",server.port);  
146  
147  
148  
149  
150
```

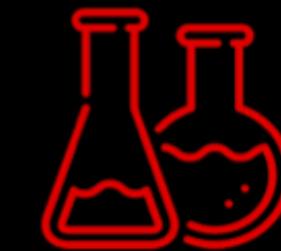


Exclusive forum access

## 12 Module Course

PRE-SALE

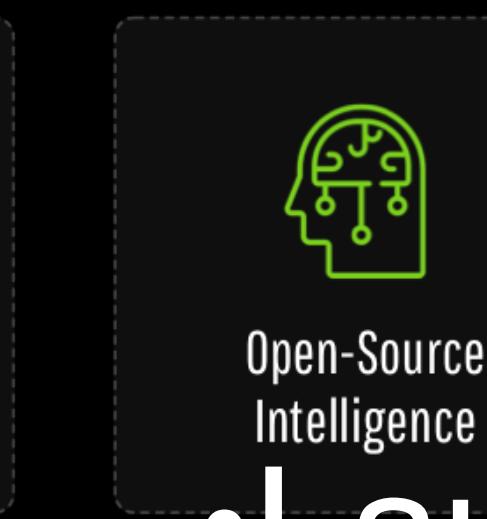
\$ 695 ~~\$1549~~



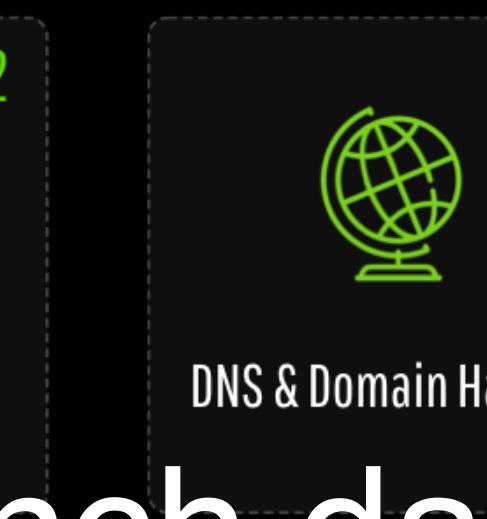
Download the labs



Ethics & Legalities



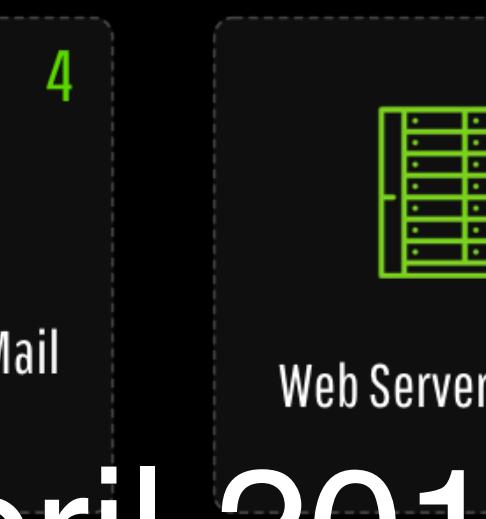
Open-Source Intelligence



DNS & Domain Hacking



E-Mail attacks & Mail infrastructure



Web Server Hacking



VPN Attacks



File Servers & Internal Attacks



UNIX Server Infrastructure



9

Databases



10  
Web Application Assessments



11  
Windows Enterprise Environments



12  
Art of Password Cracking

•Launch date: 18 April 2019

```
161     host = gethostbyname(server);  
162  
163  
164
```

# Hands-on Hacking™ Launch

54

```
165     exit(0);  
166 }
```

@hackerfantastic

@myhackerhouse

Pyongyang 2407  
Hacking North Korea



<https://hacker.house>



# Questions & Thanks!

Pyongyang 2407 ROM, tools, documents & exploits

[https://github.com/hackerhouse-opensource/pyongyang\\_2407](https://github.com/hackerhouse-opensource/pyongyang_2407)

[https://github.com/hackerhouse-opensource/pyongyang\\_2407#acknowledgements](https://github.com/hackerhouse-opensource/pyongyang_2407#acknowledgements)

<https://hacker.house>

