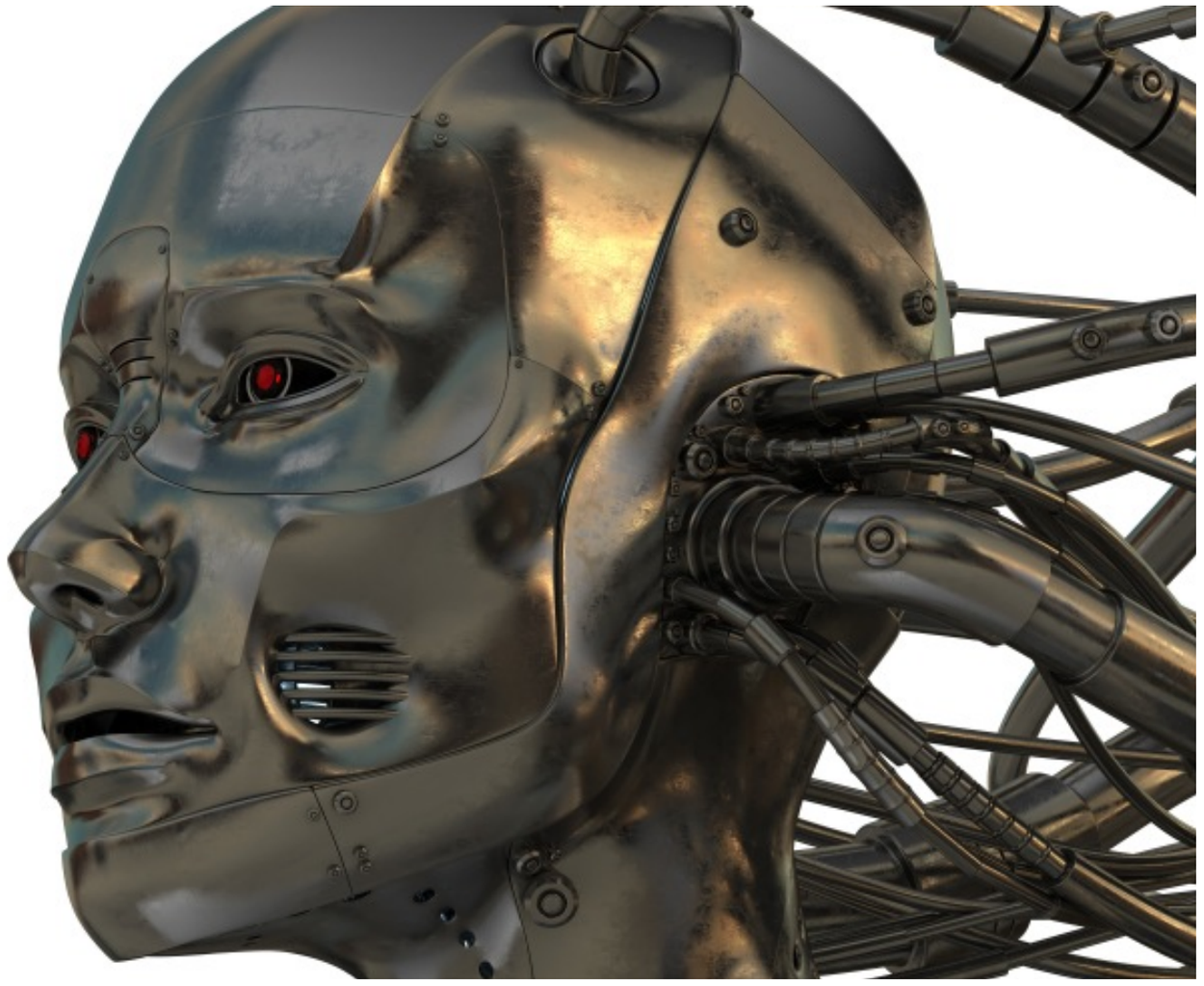


Into The Droid

Gaining Access to Android User Data

VIAFORENSICS

DEF CON 20

Introduction

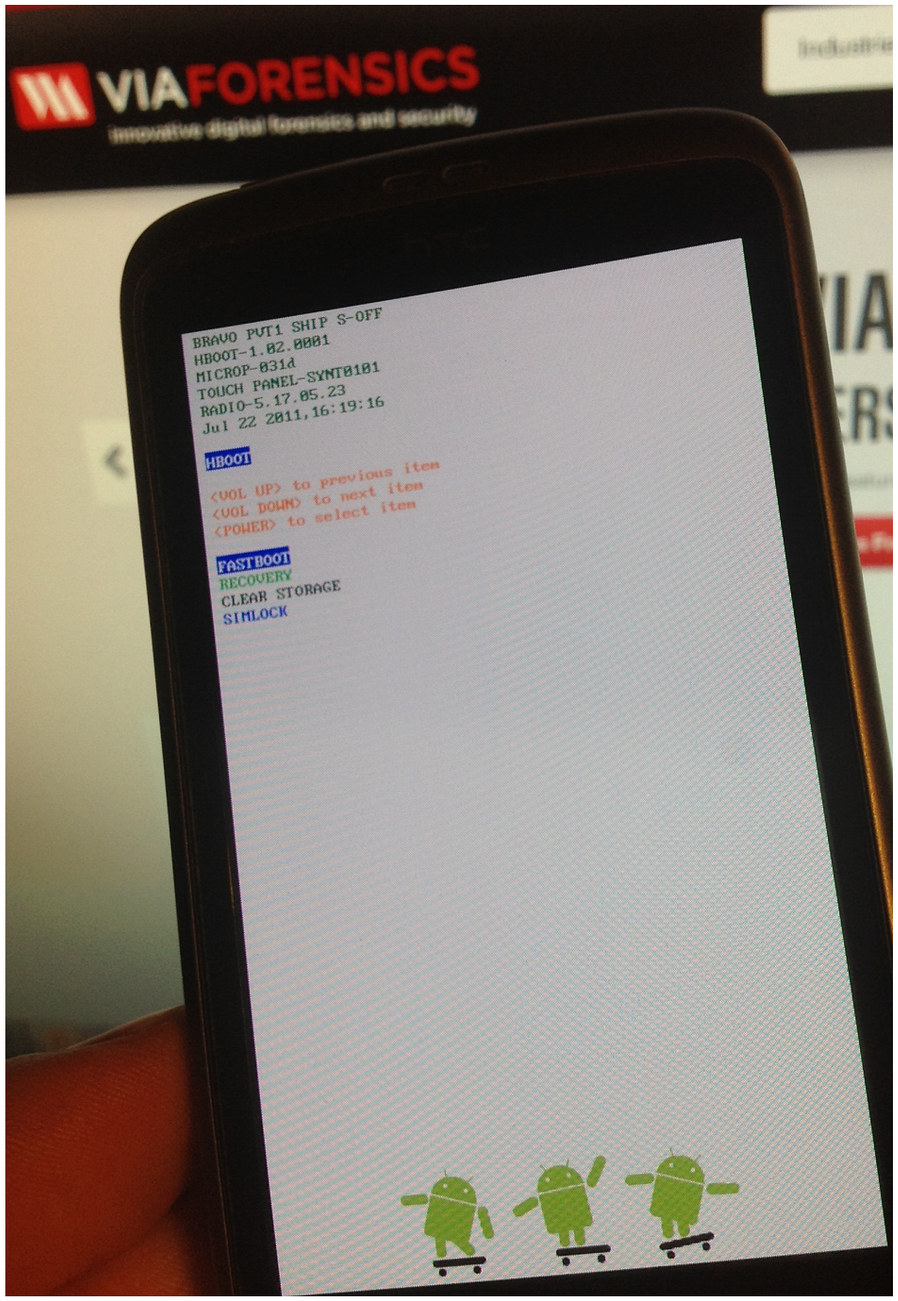
- Why this talk is useful
  - Defend access / gain access
  - Device seizure, loss, border crossing, stop and search, espionage...
- The company
  - viaForensics - Mobile security and digital forensics, strong R&D team, government agencies and corporations
- The speaker
  - Thomas Cannon - Director of Breaking Things

Challenges

- ADB off by default
- Screen lock
- Code signing for updates and boot images
- Encryption
- Variety of device hardware, software and configuration

Bootloader Essentials

- How we use the bootloader
- Accessing bootloader mode
- Bootloader protocols
- Bootloader protection

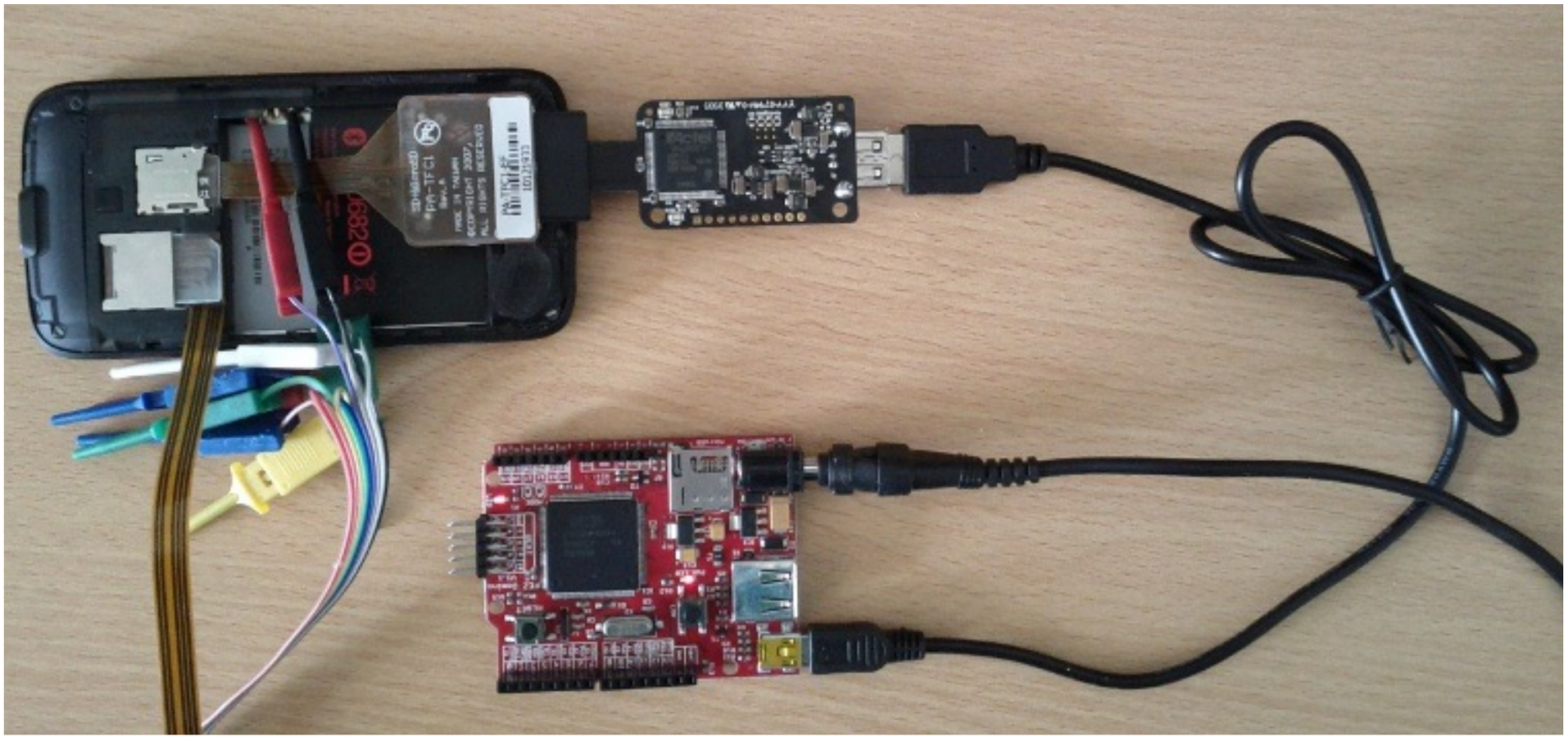


Defeat The Bootloader

- S-ON vs S-OFF
- @secuflag controlled in radio firmware
- Gold Card - specially formatted MicroSD card can bypass carrier ID check when flashing ROMs
- White Card - special SIM card used as an authentication token to control access to diagnostic mode

HTC Example

Defeat The Bootloader



- Emulate White Card with hardware, combine with Gold Card to enter diagnostics and clear S-ON

HTC Example

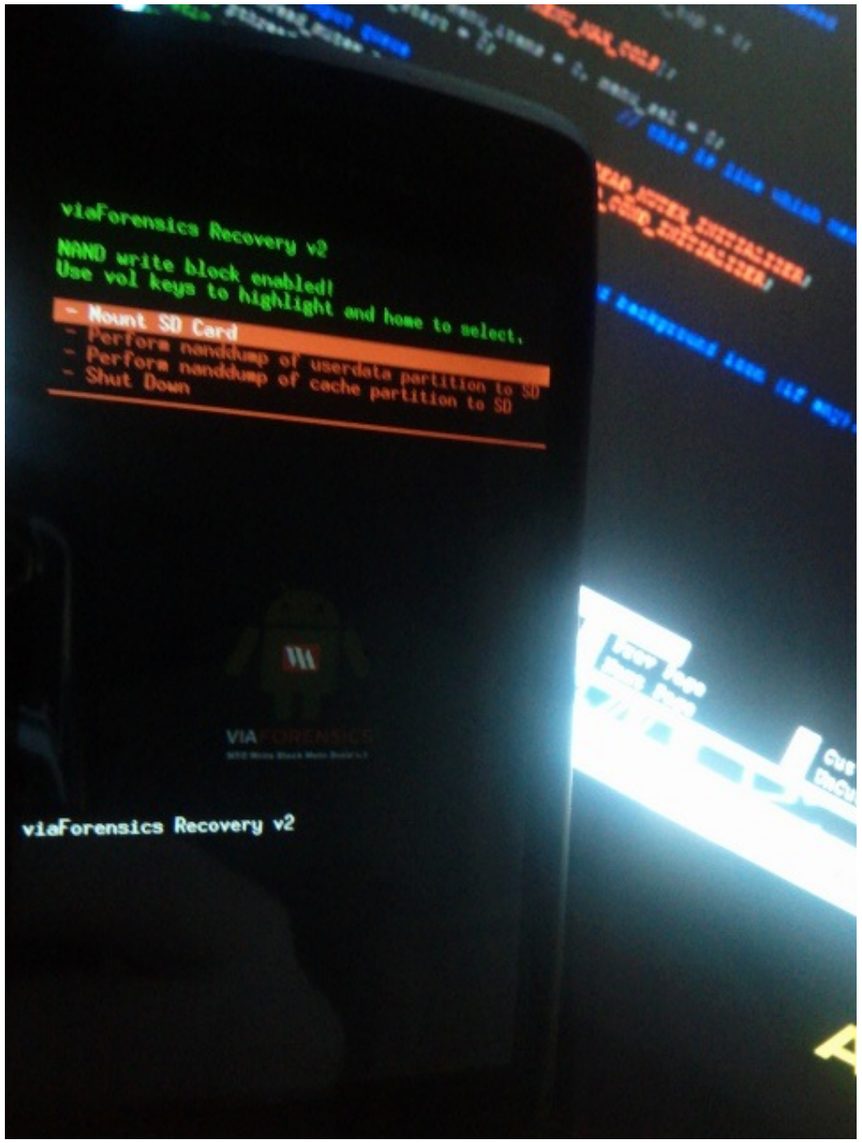
Defeat The Bootloader

- White Card not needed for CDMA phones
- Once S-OFF, can RAM load a custom boot image
- This technique wipes most devices! But not all.
- Successfully used this technique to gain access to some locked stock HTC devices such as HTC Desire
- Try it yourself with an XTC Clip

HTC Example

Forensic Boot Image

- Start early in the boot chain before the system loads
- Provide ADB root shell over USB which can be used to image the device
- Do not mount anything, including cache, to prevent any writes to partitions
- Devices with raw NAND flash and wear levelling implemented in software (YAFFS2) can be prevented from overwriting deleted data



Build Boot Image

```
$ abootimg -x stock-recovery.img
$ abootimg-unpack-initrd
$ cd ramdisk
(edit ramdisk contents)
$ cd ..
$ abootimg-pack-initrd -f
$ abootimg -u stock-recovery.img -r initrd.img
```

RAM Disk Contents

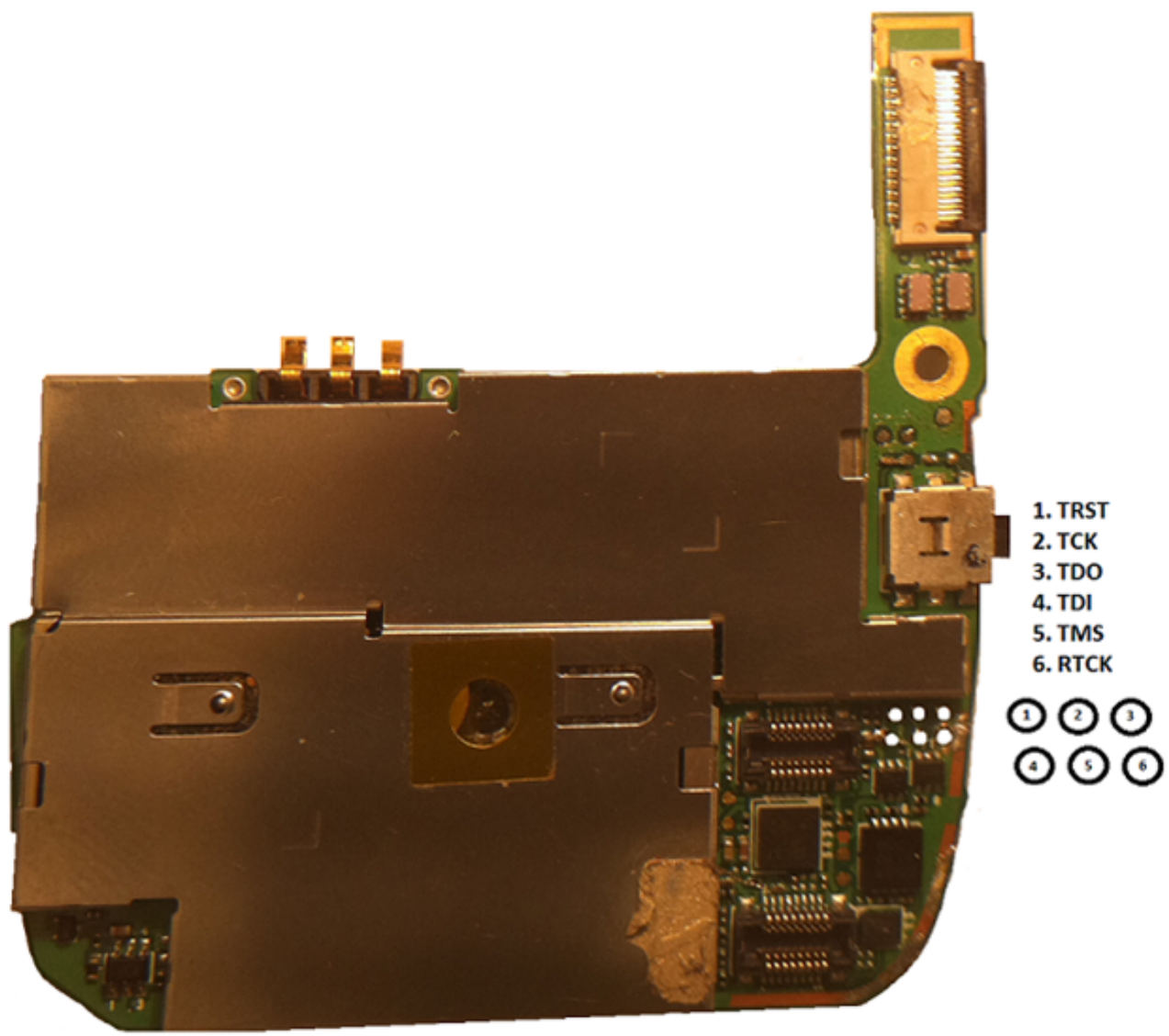
/dev
/proc
/sbin
    adbd
    busybox (+ symlinks)
    nanddump (to dump partitions)
/sys
init
default.prop (enable root shell, ro.secure=0)
init.rc (do not mount partitions, just start adbd)
ueventd.rc

Flash and RAM Load

- Samsung
  - Dump partitions with ODIN <= 1.52 or Heimdall. Maybe.
  - Flashing with ODIN or Heimdall
    - heimdall flash --recovery recovery.bin (Epic 4G)
    - heimdall flash --kernel zImage (Galaxy S)
- HTC
  - fastboot boot recovery.img (RAM Loading)
  - fastboot flash recovery recovery.img (flash partition)
- Motorola
  - sbf_flash image name.sbf (make sure it only contains recovery)

JTAG Primer

- How it works
- Flasher Box
  - ORT
  - RiffBox
  - Medusa Box



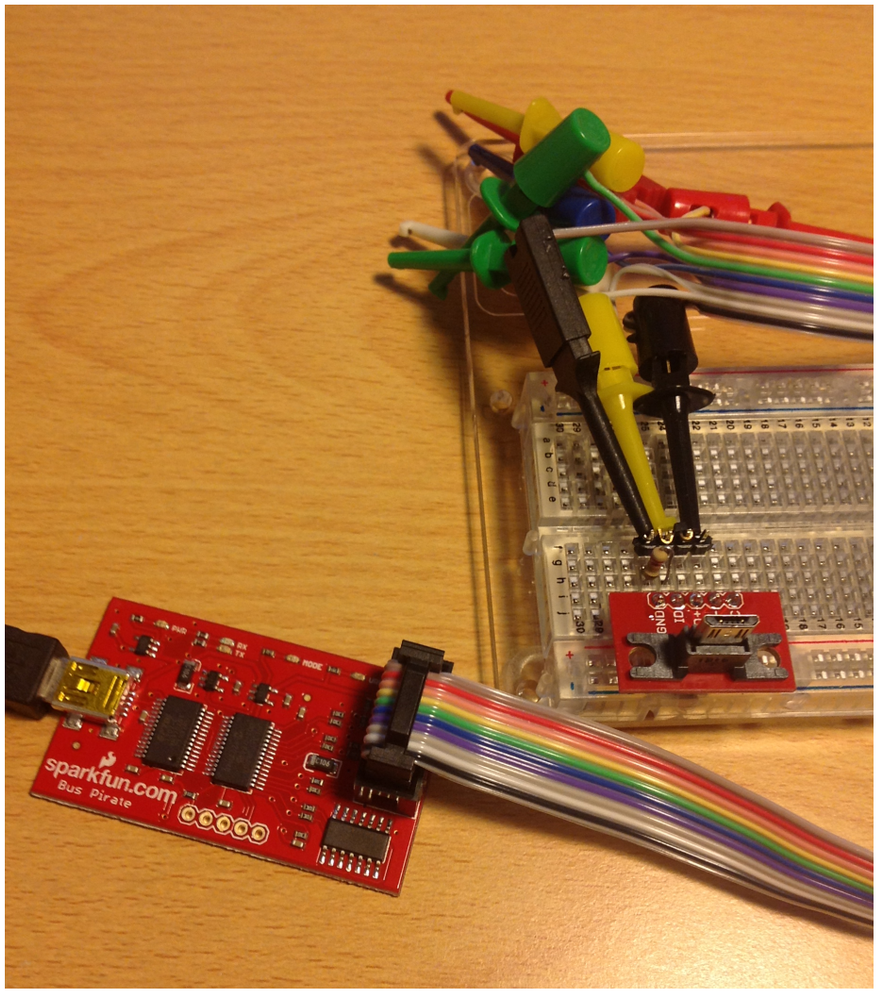
Serial Debug Cable

- Some devices have debug access via serial cables which can be used to gain access to data

- On Samsung Galaxy SII / Galaxy Note this is activated by grounding ID pin of USB with a 523K ohm resistor

- TTL serial access provided on D+ and D- pins of USB connector

- Use a Bus Pirate and MicroUSB breakout board to connect



Galaxy SII

Crack PIN or Password

- Salt
  - /data/data/com.android.providers.settings/databases/settings.db
    - SELECT * FROM secure WHERE name = 'lockscreen.password_salt'
- PIN / password
  - /data/system/password.key
  - Salted SHA1 of password concatenated with salted MD5

Crack PIN or Password

- Calculate the value of the salt in lowercase hex with no padding

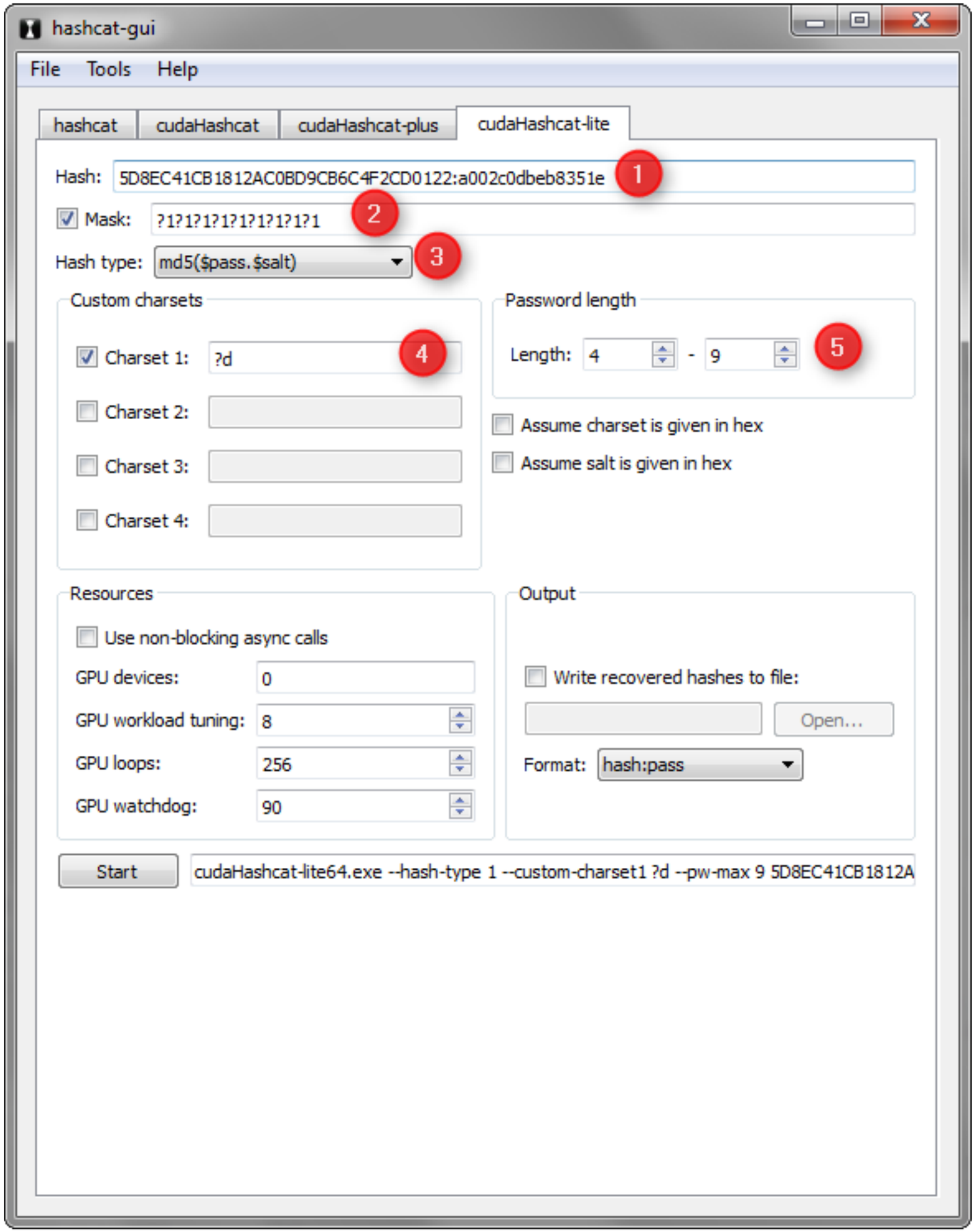
$ python -c "print '%x' % 72062437792525196l4"

a002c0dbeb8351e

- Copy the last 32 bytes of password.key (MD5 hash in hex), add a colon and then add the salt

5D8EC41CB1812AC0BD9CB6C4F2CD0122:a002c0dbeb8351e

- Crack with software such as oclHashcat-lite


hashcat-gui
File
Tools
Help
hashcat
cudaHashcat
cudaHashcat-plus
cudaHashcat-lite
Hash:
5D8EC41CB1812AC0BD9CB6C4F2CD0122:a002c0dbeb8351e
1
Mask:
?1?1?1?1?1?1?1?1?1
2
Hash type:
md5($pass.$salt)
3
Custom charsets
Charset 1:
?d
4
Charset 2:
Charset 3:
Charset 4:
Password length
Length:
4
-
9
5
Assume charset is given in hex
Assume salt is given in hex
Resources
Use non-blocking async calls
GPU devices:
0
GPU workload tuning:
8
GPU loops:
256
GPU watchdog:
90
Output
Write recovered hashes to file:
Open...
Format:
hash:pass
Start
cudaHashcat-lite64.exe --hash-type 1 --custom-charset1 ?d --pw-max 9 5D8EC41CB1812A

hashcat-gui

File Tools Help

hashcat | cudaHashcat | cudaHashcat-plus | cudaH

Hash: 5D8EC41CB1812AC0BD9CB6C4F2CD0122:a002c0d

Mask: ?1?1?1?1?1?1?1?1 (2)

Hash type: md5($pass.$salt) (3)

Custom charsets

Charset 1: ?d (4)

Charset 2:

Charset 3:

Charset 4:

Resources

Use non-blocking async calls

GPU devices: 0

GPU workload tuning: 8

GPU loops: 256

GPU watchdog: 90

Start | cudaHashcat-lite64.exe --hash-type 1 --c

C:\Windows\system32\cmd.exe

```
cudaHashcat-lite v0.8 by atom starting...

GPU-Loops: 1024
GPU-Accel: 16
Password lengths range: 4 - 9
Platform: NVidia compatible platform found
Watchdog: Temperature limit set to 90c
Device #1: GeForce GTX 560M, 1536MB, 1550Mhz, 4MCU

5d8ec41cb1812ac0bd9cb6c4f2cd0122:a002c0dbeb8351e:123456789

Status.......: Cracked
Hash.Target..: 5d8ec41cb1812ac0bd9cb6c4f2cd0122:a002c0dbeb8351e
Hash.Type....: md5($pass.$salt)
Time.Running.: 2 seconds
Time.Left....: 0 secs
Plain.Mask...: ?1?1?1?1?1?1?1?1
Plain.Text...: ****40389
Plain.Length.: 9
Progress.....: 991723520/1000000000 (99.17%)
Speed.GPU.#1.:   536.3M/s
HWMon.GPU.#1.: 83% GPU, 46c Temp

Started: Wed Jan 11 06:24:04 2012
Stopped: Wed Jan 11 06:24:08 2012
```

C:\Windows\system32\cmd.exe

```
cudaHashcat-lite v0.8 by atom starting...

GPU-Loops: 1024
GPU-Accel: 16
Password lengths range: 4 - 9
Platform: NVidia compatible platform found
Watchdog: Temperature limit set to 90c
Device #1: GeForce GTX 560M, 1536MB, 1550Mhz, 4MCU

46ec7f12b91cdd818e4bcfa933947cf2:a002c0dbeb8351e:P4sswd

Status.......: Cracked
Hash.Target..: 46ec7f12b91cdd818e4bcfa933947cf2:a002c0dbeb8351e
Hash.Type....: md5($pass.$salt)
Time.Running.: 47 seconds
Time.Left....: 54 seconds
Plain.Mask...: ?1?1?1?1?1?1
Plain.Text...: ***ijZ
Plain.Length.: 6
Progress.....: 26263945216/56800235584 (46.24%)
Speed.GPU.#1.:   561.5M/s
HWMon.GPU.#1.: 97% GPU, 61c Temp

Started: Wed Jan 11 07:30:57 2012
Stopped: Wed Jan 11 07:31:48 2012
```

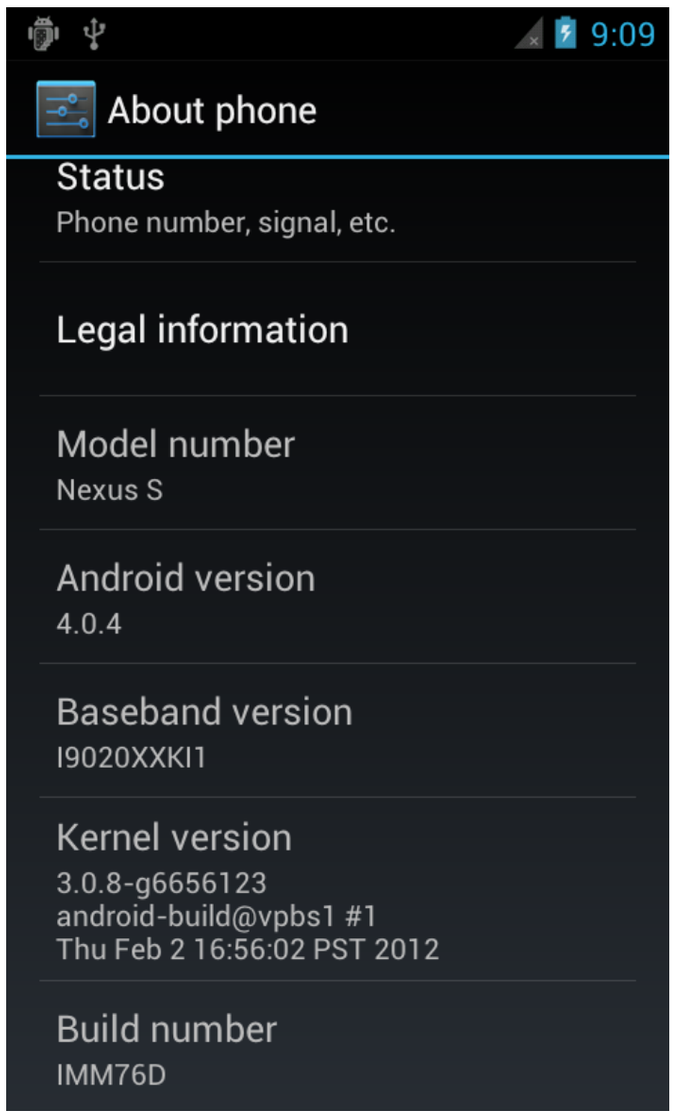
HID Brute Force?

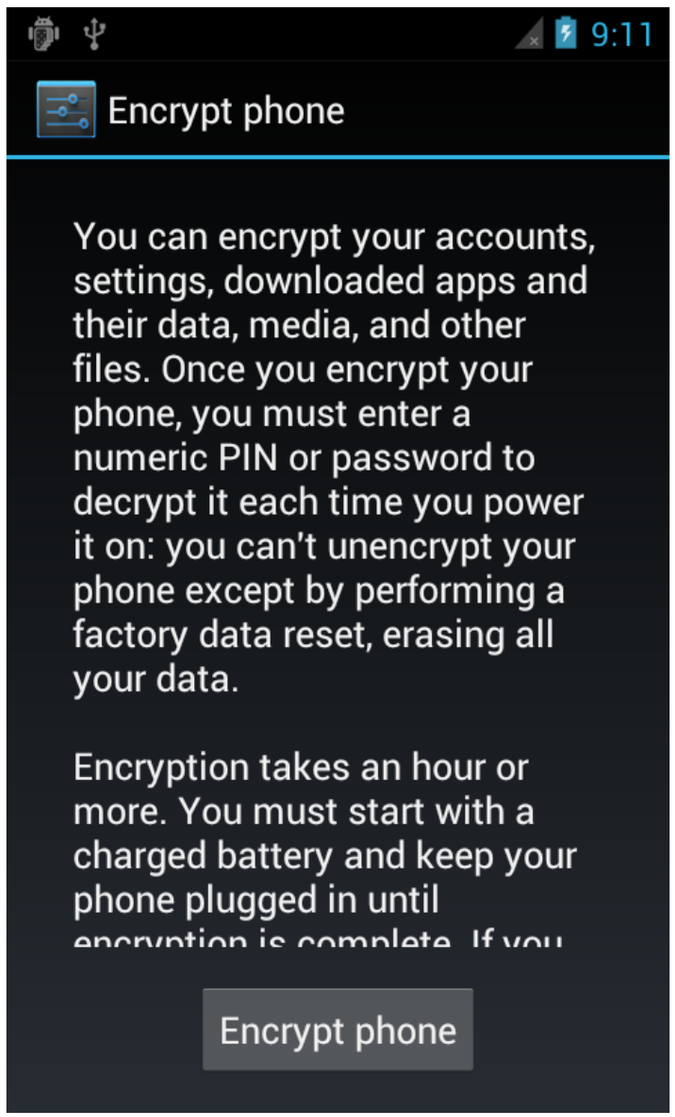
Video

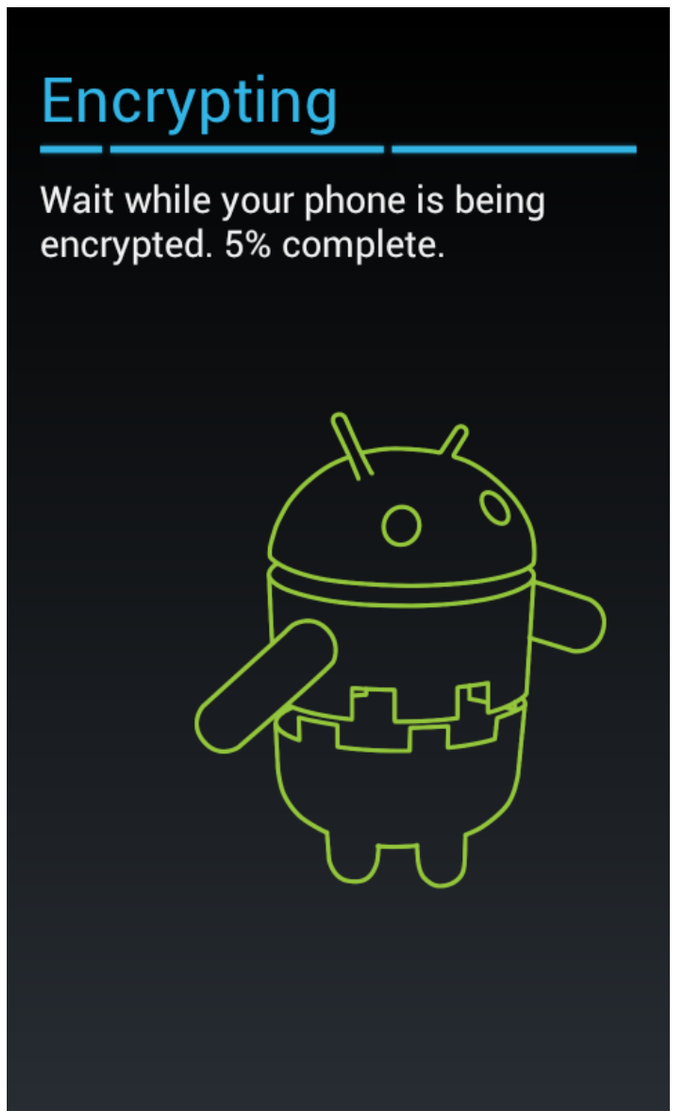
HID Brute Force

- AVR ATMEGA32U4 emulates USB keyboard typing PINs
- USB OTG cable for USB host
- Devices usually rate limit attempts and wipe after too many incorrect passcodes

Android Encryption



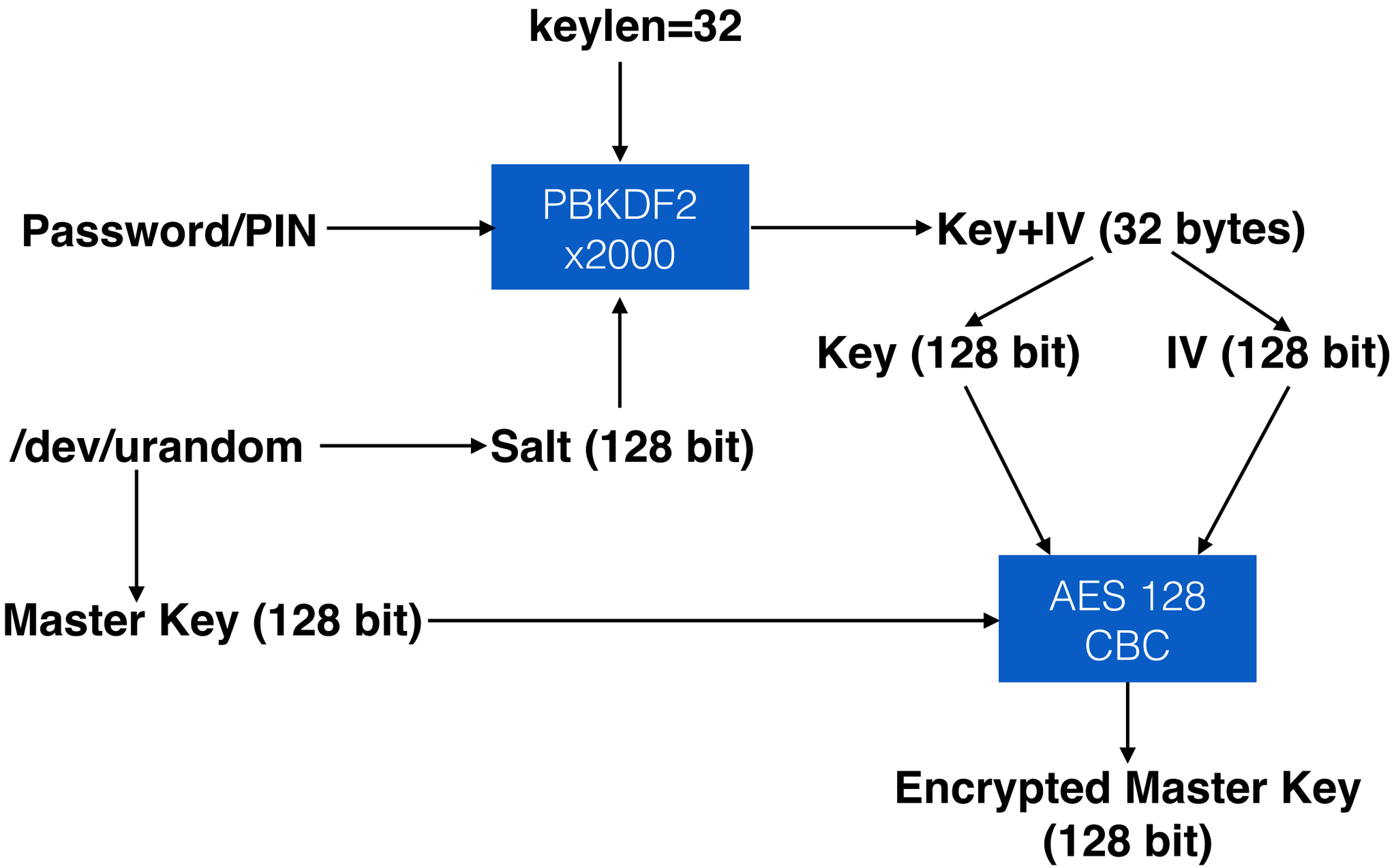




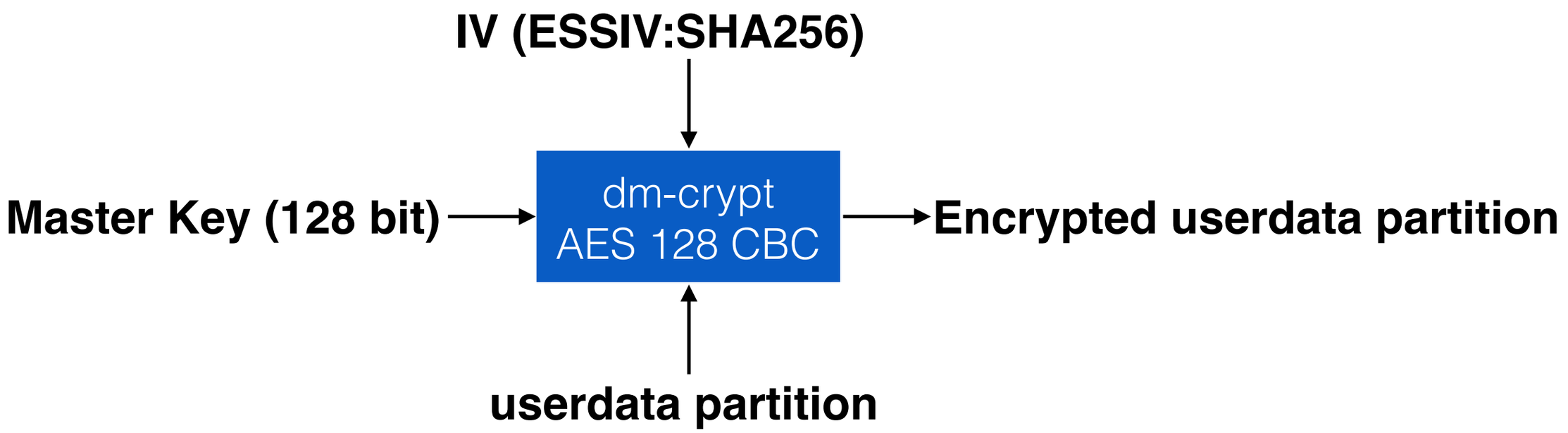
Android Encryption

- Supported since Android 3.0
- Based on dm-crypt
- AES 128 CBC
- Implementations may vary, e.g. Samsung has their own key management module

Android Encryption



Android Encryption



Cracking Encryption

- Encrypted Master Key + Salt stored in footer

```
struct crypt_mnt_ftr {
    __le32 magic;
    __le16 major_version;
    __le16 minor_version;
    __le32 ftr_size;
    __le32 flags;
    __le32 keysize;
    __le32 spare1;
    __le64 fs_size;
    __le32 failed_decrypt_count;
    unsigned char crypto_type_name[MAX_CRYPTO_TYPE_NAME_LEN];
};
```

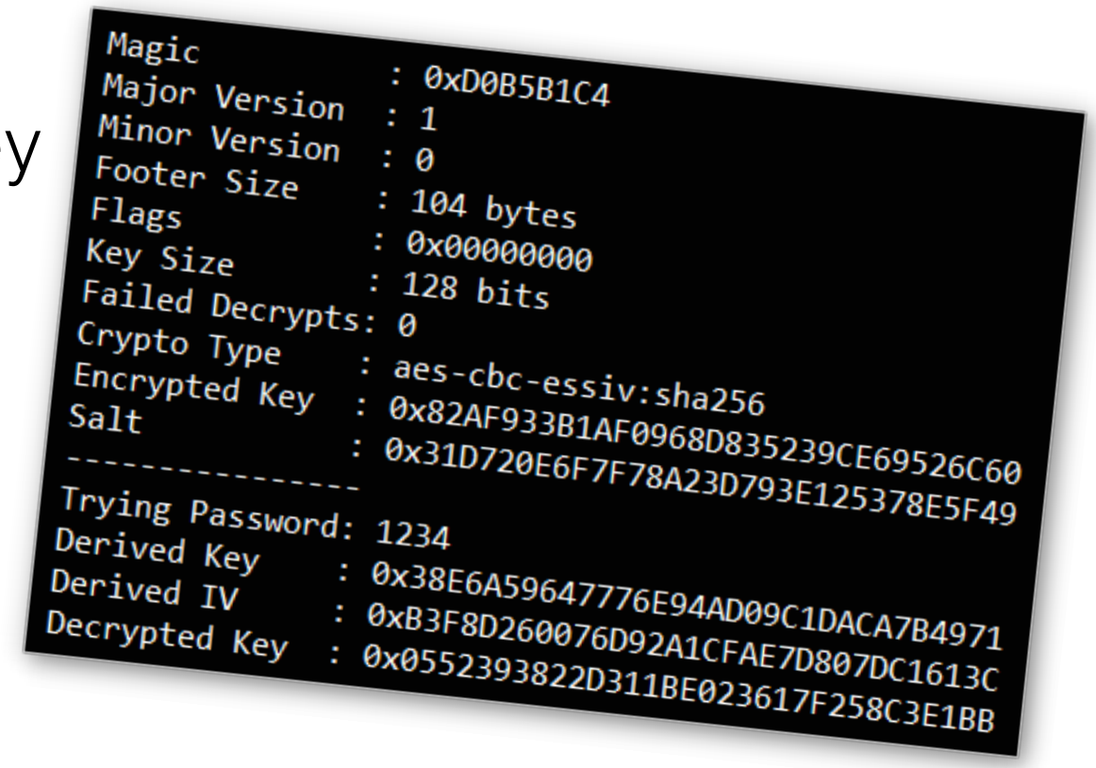
- Footer stored at end of partition or in a footer file on another partition or as a partition itself

- Image device and locate footer + encrypted userdata partition

```
~ # mkdir /efs
mkdir /efs
~ # mount -t yaffs2 /dev/block/mtdblock6 /efs
mount -t yaffs2 /dev/block/mtdblock6 /efs
~ # ls /efs
ls /efs
bluetooth       lost+found      nv_data.bin     userdata_footer
imei            nv.log          nv_data.bin.md5
```

Cracking Encryption

- Parse footer
- Locate Salt and Encrypted Master Key
- Run a password guess through PBKDF2 with salt, use resulting key and IV to decrypt master key, use resulting master key to decrypt first sector of encrypted image.
- If password is correct, plain text will be revealed



```
Magic          : 0xD0B5B1C4
Major Version  : 1
Minor Version  : 0
Footer Size    : 104 bytes
Flags          : 0x00000000
Key Size       : 128 bits
Failed Decrypts: 0
Crypto Type    : aes-cbc-essiv:sha256
Encrypted Key  : 0x82AF933B1AF0968D835239CE69526C60
Salt           : 0x31D720E6F7F78A23D793E125378E5F49
----------------
Trying Password: 1234
Derived Key    : 0x38E6A59647776E94AD09C1DACA7B4971
Derived IV     : 0xB3F8D260076D92A1CFAE7D807DC1613C
Decrypted Key  : 0x0552393822D311BE023617F258C3E1BB
----------------
ESSIV IV       : 0xB31C2837995393102ECC539D460D77C1
Decrypted Data : 0xEE961EE40CC036D88D2D29206D888FC500000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
000
```

Correct PIN

```
Magic          : 0xD0B5B1C4
Major Version  : 1
Minor Version  : 0
Footer Size    : 104 bytes
Flags          : 0x00000000
Key Size       : 128 bits
Failed Decrypts: 0
Crypto Type    : aes-cbc-essiv:sha256
Encrypted Key  : 0x82AF933B1AF0968D835239CE69526C60
Salt           : 0x31D720E6F7F78A23D793E125378E5F49
----------------
Trying Password: 5555
Derived Key    : 0x3AC2D38F705281EBB45430D5770B2BFD
Derived IV     : 0xAF4CB6F2C3481C20B8430DE869608A4A
Decrypted Key  : 0x85BE68592503F89CB0F9BBD82972AE07
----------------
ESSIV IV       : 0x52989B5B082368326FB4014D06A0A67C
Decrypted Data : 0xB0D17B11AB13D1C432A4816ABF39319976187A54D58961B7808191E6AA4F4
06E8A4DCC2D4986F377981D79AD7C625390545B3033472EB069AEC33451BF8DDB8E6B4A502C41A61
1309BDFC3E65A88027B7ECCB1B7D04573EAD6194C87C0A8D661E62D942354EF218378E76EB30BFB2
5E97391AC9A268BCD49F274680313ED1D1FFD4663B249A282CCC49D5C54EB528427FCC43B3CDF2FC
86892271F89736CE53BBD117BFBB8025DA75C74EF46025F6AC1547DA8D3F1FCC6542DA0EBA9F5D87
0D3743E5440D61D4DDAE26048018002C597335FDDB84AF89922E0C165128A3C51082877CE0F51840
12FFE73CA346922B1F931E3C6EE30AC781479D2DB6FB168965A92809B54523DC228AC473AE8164A8
F20A2BB74617CFB536AF3B61BBF0C273BE30F006E646817E42FD4579B8F3B34934C12F5D280CE764
F1381120D746683F2AC5AF825CC25AE089B355044E86CC207484A0F365E465C7B2E980456D4C0BA8
462A19EF4227E1E3AF92A9386BF585894327740647B537DB96302B51899DED97B37FC49D1B35F618
FC4F775E3FB159DD4800FB8577C38404AAD366E18E1A62C2E944CDE45732A9B5E20F80D6DE03FCAF
855CE41D1D8CE3D033FE8B05F861F781DCA100A18DF910EA519A6E52C15DADCCC8C10A42AC3D7BD0
DE37C3B28932131D37198BD7AB1D48CB895C8B1159715F49424887D245034C6510F43FED085CBE23
8CA
```

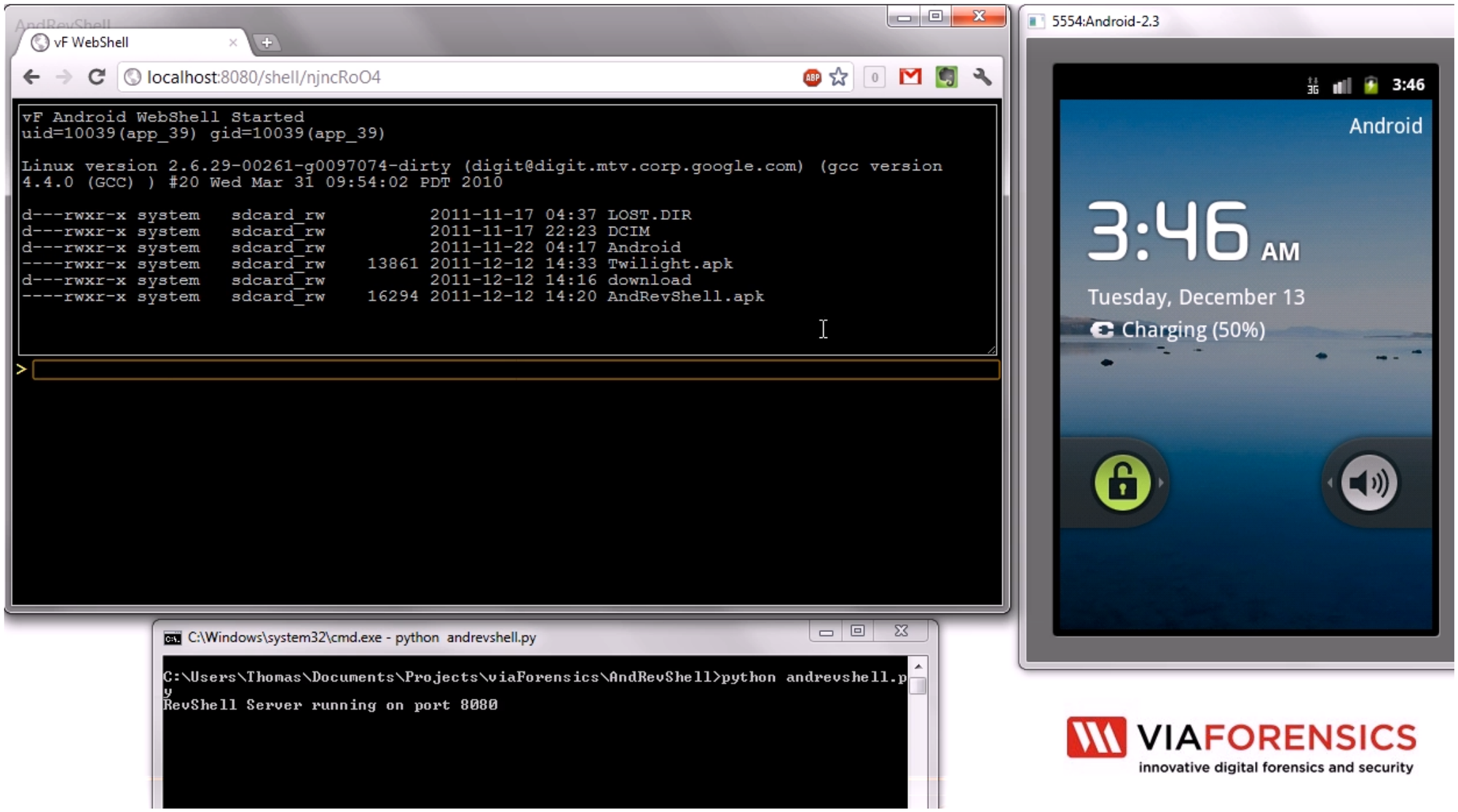
Incorrect PIN

- Cracking PINs takes seconds. Passwords are usually short or follow patterns due to being the same as the lock screen password

Evil Maid Attack

- Load app onto system partition, wait for user to boot phone, get remote access to decrypted user data
- Rootkits - easy to compile for Android
- Evil USB charger

Reverse Shell



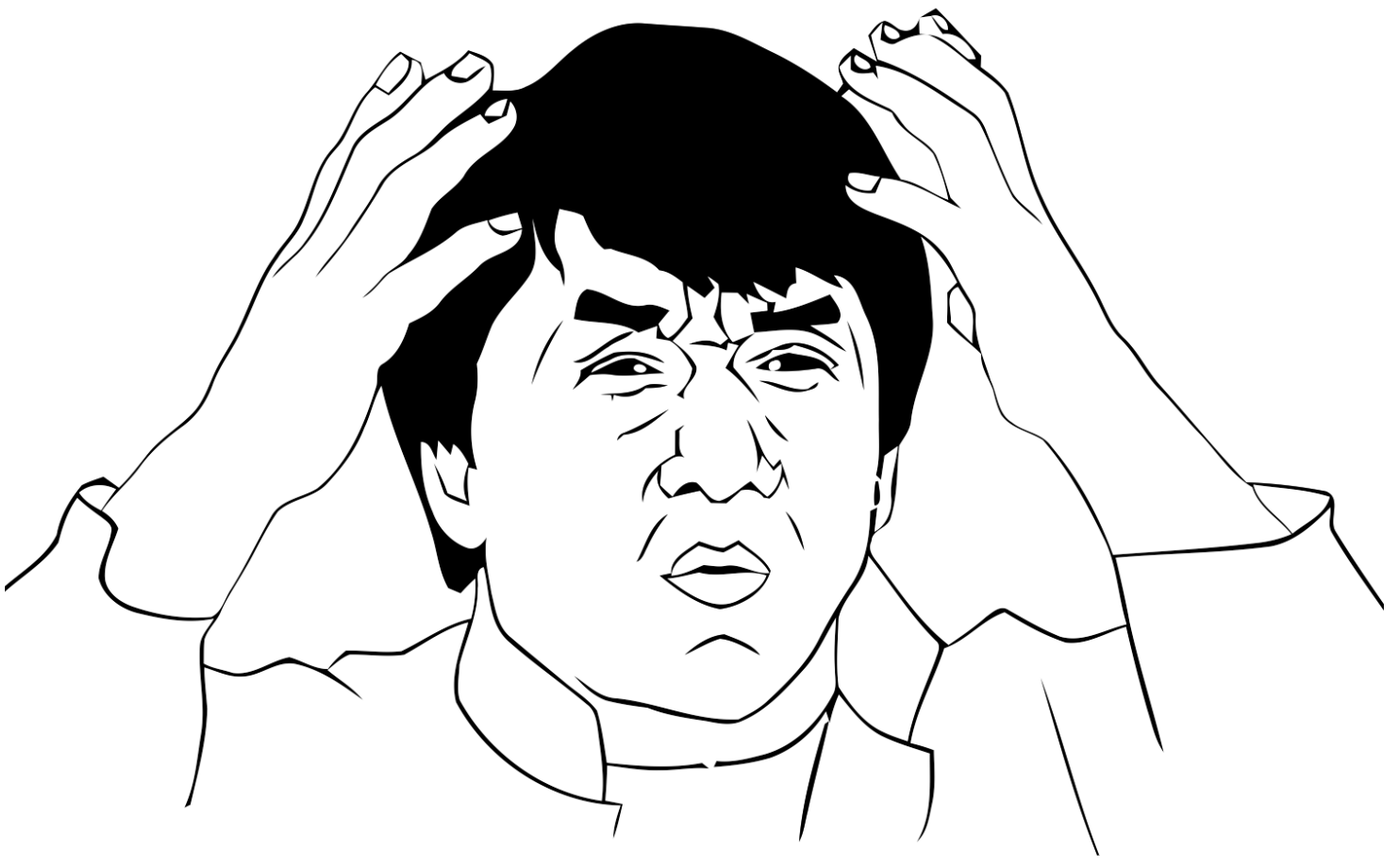
- App with no permissions can create a reverse shell, giving remote access to attacker

Desperate Techniques

- Hard reset - some devices prior to 3.0 did not wipe data properly. Wipe, boot, root and recover

- Chip-off - de-solder NAND chips

- Screen smudges



More Techniques!

- Custom update.zip - can you get one signed?
- Race condition on updates via SD cards - fixed
- Own a CA? Who doesn't these days? MITM connection, push app, update or exploit
- Entry via Google Play, if credentials cached on desktop

Santoku Linux

- Free and open bootable Linux distribution full of tools
- Project is a collaboration with other mobile security pros
- Mobile Forensics
- Mobile App Security Testing
- Mobile Malware Analysis

Check out the Alpha release at **https://santoku-linux.com**

VIAFORENSICS

Thank you!

Thomas Cannon

@thomas_cannon
github.com/thomascannon
tcannon@viaforensics.com

For the latest versions of our presentations visit:
https://viaforensics.com/resources/presentations