Challenges in Physical Extraction of Modern Smartphones and Advance Methods to overcome

Yuval Ben-Moshe

SANS Digital Forensics Summit, Prague October-2012

Images and examples, courtesy of Cellebrite
Agenda

- Motivation
- Forensics Extraction Overview
- Physical Extraction: Challenges & Methods
- Toolbox and real life examples
Estimating 9 Billion Subscribers in 2017
Figure 3: Mobile subscriptions penetration in Q1 2012

- Western Europe: 126%
- Central & Eastern Europe: 123%
- Latin America: 100%
- North America: 97%
- Middle East: 96%
- APAC excluding China & India: 82%
- India: 70%
- China: 69%
- Africa: 55%

Source: Ericsson (June 2012)
Seized Phone Offers Clues to Bin Laden’s Pakistani Links

The United States is trying to find out how Al Qaeda’s leader, Osama bin Laden, was able to live comfortably for years in a compound in Abbottabad, Pakistan.

By CARLOTTA GALL, PIR ZUBAIR SHAH and ERIC SCHMITT
Published: June 23, 2011

ISLAMABAD, Pakistan — The cellphone of Osama bin Laden’s trusted courier, which was recovered in the raid that killed both men in Pakistan last month, contained contacts to a militant group that is a longtime asset of Pakistan’s intelligence agency, senior American officials said on Thursday. The officials said...
Extraction Methods

Logical Extraction

File System Extraction

Physical Extraction

Using phone's purposely built interface and protocol

Advantages:
- Fast
- No Decoding
- Established interface
- Complexity: Low

Disadvantages:
- Limited Data Availability
Extraction Methods

- Logical Extraction
- File System Extraction
- Physical Extraction

Copying device’s full file system

**Advantages:**
- Fast
- More data available
- Complexity: Intermediate

**Disadvantages:**
- Requires decoding
- Allocated space only
Extraction Methods

Logical Extraction

File System Extraction

Physical Extraction

Bit-for-Bit copy of internal flash memory

**Advantages:**
- Ultimate details availability
- Complexity: High

**Disadvantages:**
- Time consuming
- Requires decoding
Extraction Methods, Physical

- **App Based**
  - Device must be unlocked and on

- **JTAG**
  - Device Specific; Minimal documentation

- **Chip-Off**
  - Distractive

- **Flasher Box**
  - Device Specific; Forensically sound ??

- **Bootloader**
  - Bypasses OS
  - Full access to memory
Boot Loaders

- A bootloader is a small program/set of instructions/operations that kick off the startup process and hand off to the main controlling program, like the operating system, which supports the main or major device operations.

- In layman words:
  - The first set of operations the computer does when it is turned on.
Why Boot Loaders

- No OS, Less security
- Generic, Per Family of Devices (typically)
- Safe, Designed for Read-Only
- Accurate, Accesses Spare Areas
It’s a long journey to ‘Bootloading’
The overly simplistic map

**Push the bootloader**

- Unknown Protocol
- Security Signatures

**Gain access to memory**

- No Drivers

**Send data outside**

- No USB Drivers
Our Toolbox

- Reverse Engineering
  - Learn and understand the inner workings of the device
- Vulnerability Research
- Encryption & Digital Signatures

Make it do something it was not intended to do
Reverse Engineering

Example

```c
#include <stdio.h>

int main()
{
    printf("Hello 2012!");
    return 0x1337;
}
```
Reverse Engineering
Example (cont.)

![Hex Workshop]

```
0123456789ABCDEF0123456789ABCDEF
.
.
I.L! This program cannot be run in DOS mode...
```

```
PE L.
```

Reverse Engineering
Example (Cont.)

```
.text:00401000 ; Attributes: bp-based frame
.text:00401000 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:00401000 _main       proc near           ; CODE XREF: __tmainCRTStartup+11D↓p
.text:00401000 argc       = dword ptr  8
.text:00401000 argv       = dword ptr 0Ch
.text:00401000 envp       = dword ptr 10h
.text:00401000 push       ebp
.text:00401001 mov        ebp, esp
.text:00401003 push       offset Format ; "Hello  2012!"
.text:00401008 call       ds:printf
.text:0040100E add        esp,  4
.text:004011 mov        eax, 1337h
.text:0040116 pop         ebp
.text:0040117 retn
.text:0040117 _main       endp
```

IDA - C:sers\nadavh\Documents\visual studio 2010\Projects\test\Release\test.exe
Reverse Engineering

Example (Cont.)

```c
{ 
    printf("Hello MFC 2012!");
    ret_val = 0x1337;
}
```
Reverse Engineering
Real World Usage

- Blackberry Physical Extraction

- Debug protocols
  - Understanding supported commands
  - Understanding protocol framing
  - How to upload code to the device
  - Understanding bootloader signature
    - ... and how to bypass it.
Vulnerability = Access
Vulnerability Research

introduction

- Mostly caused by programming oversight

- Comes in all shapes and sizes:
  - E.g.: Stack/heap overflow
    - Data exceeding expected size/length overwrite other values
  - Directory traversal
    - Used in web attacks to access server files

>> what file would you like? I have “img.jpg” and “photo.png”.

<< I would like to have “…/…/…/…/…/…/…/etc/passwd”
State Machine Mistreated

<< Sending unsigned code to device

<< Asking to verify signature

>> Error! Incorrect signature!

<< Asking to run code

>> Error! Can’t run unsigned code

<< Sending unsigned code to device

<< Asking to run code

>> OK! I have no record of the signature being invalid…

Encryption:

- Known (and unknown) mathematical exploits
Encryption & Digital Signatures
Encryption & Digital Signatures

Introduction

- Encryption is used in many places
  - Databases (Android and iPhone)
  - Records (Blackberry)
  - Protocols (iDen)
  - Log files (TomTom GPS)
  - File systems (iPhone)

- Digital signatures
  - Signed firmwares
  - Signed Apps (iPhone)
  - Signed Boot loaders (Nokia, Blackberry, Motorola)

- Asymmetric Challenge Response
  - Boot loader upload protocol (Sony Ericsson)

- Hardware or Software implemented
Tackling Encryption & Signatures

- **Reversing**
  - Sometimes all you need is to understand the algorithm and put your hands on the key to decrypt.

- **Finding exploits**
  - In some cases the programmer did not implement the encryption correctly, making it easier to break.

- **Brute Force**
  - In some cases the only option is to guess all possible keys.

**Encryption Rule #1:** if you implemented it yourself- it has holes.
Encryption & Digital Signatures

Real World Usage

- TomTom GPS devices record a detailed log wherever you go. The device just has to be turned on. What we get is encrypted triplog files.
TomTom GPS devices records detailed trip-log go
- The device just has to be turned on
- Files are encrypted

By means of reversing and a lot of trial and error revealed a vulnerability in the encryption algorithm
Real World Usage

- TomTom GPS devices record a detailed log wherever you go.
  - The device just has to be turned on.
  - What we get is encrypted triplog files.
  - A lot of reversing work, trial and error and a few white nights helped us find a vulnerability in the encryption algorithm.
  - And the result...
The Not So simplistic map

- Push the bootloader
- Gain access to memory
- Send data outside
Our Toolbox

- Reverse Engineering: Learn and understand the innards of the device
- Encryption & Digital Signatures
- Vulnerability Research: Make it do something it was not intended to do
Motorola Android Physical Extraction

Full Scale Example
The challenge:

- Physical image of flash memory from locked Android Devices
Special ‘Factory’ cable puts the device in RSD mode
- Power on the device without a battery
- Uses standard micro-USB connector with non-standard wiring
- Sketching, Cutting and Soldering skills became handy
1. The RSD Protocol

- Download mode (RSD Mode)
  - Pre-OS, i.e. agnostic to device locking

- The problem:
  - When in boot loader mode, the device accepts only signed boot loaders

- Supported Actions
  - Send and execute signed code
  - Receive device information
  - Power-off / restart device
Motorola Physical Extraction

2. The Signature

- Signature
  - Proprietary format
  - RSA 2048
  - 2-3 certificates – signed by a root key

- Reversing found a digital signature exploit
  - A way to upload unsigned code
- Reversing the firmware
- Understanding how to read the memory
- Writing a generic implementation
Forensic physical extraction for locked Motorola Android Devices

Skills employed:
- Hardware reverse Engineering
- SW Reverse Engineering
  - protocol, memory drivers,
- Encryption Reversed Engineering:
  - Digital Signature
Recap

Motivation
- Data on mobile devices is a goldmine of information

Forensics Extraction Overview
- Logical, File-System, Physical Extraction

Physical Extraction: Challenges & Methods
- App Based, JTAG, Chip-Off, Bootloaders

Toolbox and real life examples
- Reverse Engineering, Vulnerability research, Breaking Encryption and Digital Signatures

Power of Bootloaders
- Physical of Motorola Android device
Questions
Thank You

Yuval Ben-Moshe

Yuvalbm@Cellebrite.com