Embedded Reverse Engineering: Cracking Mobile Binaries

Seth Fogie @ Defcon 11

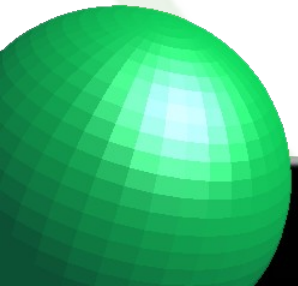
Who am I and what am I doing?

- **Airscanner.com**
  - **Mobile Security (AV, firewall, sniffer)**
- **Dissemination of Information**
  - **Reverse-engineering is a tool…not a weapon**
  - **Knowing your computer**
  - **Don’t steal…pay the programmers**

Legal Issues

- **Laws**
  - –No person shall circumvent a technological measure that effectively controls access to a work protected under this title.
  - –to "circumvent a technological measure" means to descramble a scrambled work, to decrypt an encrypted work, or otherwise to avoid, bypass, remove, deactivate, or impair a technological measure, without the authority of the copyright owner;
- **Encryption Research & Security Testing**
  - –identify and analyze flaws and vulnerabilities of encryption technologies applied to copyrighted works
  - –accessing a …computer system…solely for the purpose of …investigating… a security flaw or vulnerability…

Windows CE Architecture



- **Processors**
  - **Power&Processing = Heat**
  - **Reduced Instruction Set Computer (RISC)**
    - **ARM (1987): StrongARM, Xscale**
    - **WinCE, ARM Linux, EPOC**
    - **Intel bought DEC (StrongARM)**
      - **Larger Cache**
      - **Dynamic Voltage**
      - **StrongARM 2.5 million transistors / Xscale 5 million**
      - **Lower power usage at higher speed**

Architecture Cont. – Kernel/Process

- **Kernel**
  - **Reduced Windows 2000 (No 16 bit or MS-DOS)**
  - **Core DLL issues**
  - **DLLs are run from ROM**
    - **Can't break a program when its executing DLL code**
- **Processes (32) with dedicated 32MB/Process**
  - **512x64k memory blocks, 16 registers per thread**
  - **Kernel (OS) & User (3rd party programs) Mode**
  - **Processes are isolated but threads share data**

Architecture Cont. – Memory

- **Memory**
  - **RAM (Program & Objects)**
    - **Lose power - lose programs and objects (files)**
  - **ROM**
    - **Store OS files**
    - **Compression**
    - **eXecute In Place – Save memory (No Compression)**
  - **Object Store (Files)**
    - **Registry: Configuration settings**
    - **Programs: Compressed area for 3rd party programs**
    - **Databases: Structured storage**

Architecture Cont. – GWES/Scheduler

- **Graphics, Windowing and Event Subsystem**
  - **Handles all messaging**
  - **Your friend (Popups)**
  - **PostMessage, SendMessage, SendThreadMessage**
- **Scheduler**
  - **Multitasking**
  - **Assigns processor time at thread level**

- **Prerequisites**
  - **ASM (concept)**
  - **Hex to Binary to ASCII to Decimal**

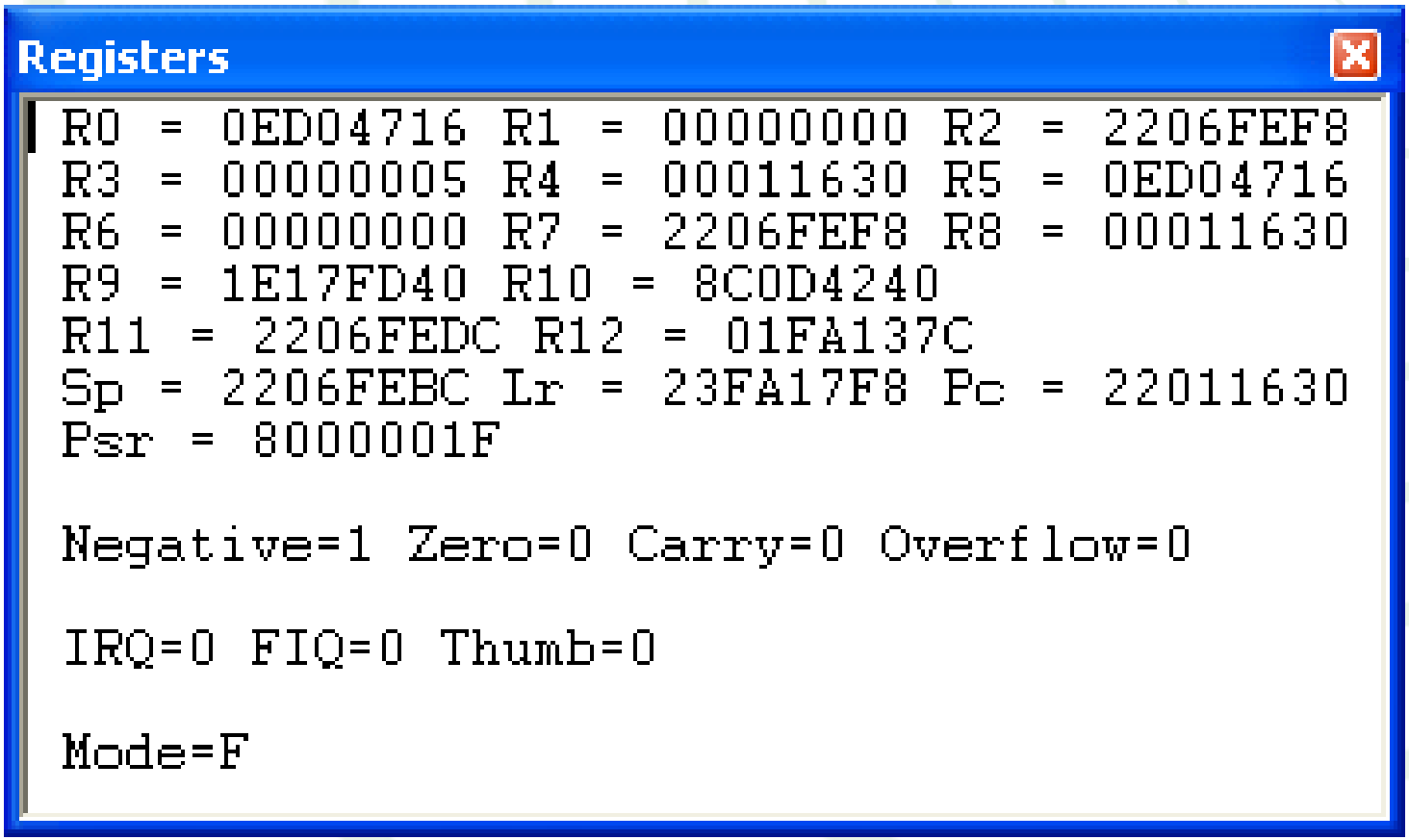
ASCII	D	E	F	C	O	N
HEX	44	45	46	43	4F	4E
Decimal	068	069	070	067	079	078
Binary	01000100	01000101	01000110	01000011	01001111	01001110

  - **ARM Processor**
    - **Registers**
    - **Opcodes**

ARM Registers

- **Registers**
  - **37 Total @ 32 bit each**
  - **Register purpose changes depending on mode**
  - **R0 – R14 + PC(R15)**
  - **R15(PC): Program Counter – Current address of execution**
  - **R14: Link Register (LR) – Hold sub routine return address.**
  - **R13: Stack Pointer (SP)**
  - **Status (NZCO)**
    - **R31: Negative / Less Than**
    - **R30: Zero (Equal)**
    - **R29: Carry / Borrow / Extend**
    - **R28: Overflow**

ARM Registers



ARM Opcodes – MOV, CMP

- **Move (MOV) – XX XX A0 EX**
  - **MOV R3, R1: 01 30 A0 E1**
  - **MOV R2, #1: 01 20 A0 E3**

- **Compare (CMP) – XX XX 5X EX**
  - **CMP R2, R3: 03 00 52 E1**
  - **CMP R4, #1: 01 00 54 E3**

ARM Status Flags

- **Status Flags**
  - **CMP R0, R1**

N	Z	C
R0 >= R1→0	R0=R1→1	R0>=R1→1

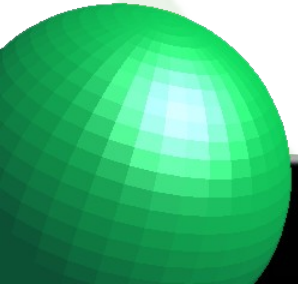
  - **MOVS R0, R1 / ANDS R0, R1, 0xFF**

N	Z	C
R1 < 0 → 1	R1 = 0 → 1	Pass through

ARM Status Flags

Airscanner

- EQ: Z set equal
- NE: Z clear not equal
- CS: C set unsigned higher or same
- CC: C clear unsigned lower
- MI: N set negative
- PL: N clear positive or zero
- VS: V set overflow
- VC: V clear no overflow
- HI: C set and Z clear unsigned higher
- LS: C clear or Z set unsigned lower or same
- GE: N equals V greater or equal
- LT: N not equal to V less than
- GT: Z clear AND (N equals V) greater than
- LE: Z set OR (N not equal to V) less than or equal
- AL: (ignored) always



ARM Opcodes – B, BL

- **Branch (B) - XX XX XX EA**
  - BEQ: If Z = 1 (XX XX XX 0A)
  - BNE: If Z = 0 (XX XX XX 1A)
  - BMI: If N = 1 (XX XX XX 4A)
- **Branch Link (BL) - XX XX XX EB**
  - BLEQ: If Z = 1 (XX XX XX 0B)
  - BLNE: If Z = 0 (XX XX XX 1B)

ARM Opcodes – LDR / STR

- Load Register (LDR) / Store Register (STR)
  - STR R1, [R4, R6] Store R1 in R4+R6
  - STR R1, [R4,R6]! Store R1 in R4+R6 and write the address in R4
  - STR R1, [R4], R6 Store R1 at R4 and write back R4+R6 to R4
  - STR R1, [R4, R6, LSL#2] Store R1 in R4+R6*2 (LSL discussed next)
  - LDR R1, [R2, #12] Load R1 with value at R2+12.
  - LDR R1, [R2, R4, R6] Load R1 with R2+R4+R6
- LDM/STM
  - STMFD SP!, {R4,R5,LR}
  - LDMFD SP!, {R4,R5,LR}

Shifting

- **LSR: Logical Shift Right – Shift the 32 bit values *right* by x number of places, using zeros to fill in the empty spots.**

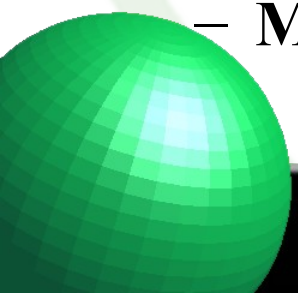
48	110000	Rsl #1 (2)	11000	27
48	110000	Rsl #2 (4)	1100	12
48	110000	Rsl #3 (9)	110	6
48	110000	Rsl #4 (16)	11	3

- **LSL: Logical Shift Left – Shift the 32 bit values *left* by x number of places, using zeros to fill in the empty spots.**

3	0011	Lsl #1 (2)	0110	6
3	0011	Lsl #2 (4)	1100	12
3	0011	Lsl #3 (9)	11000	27
3	0011	Lsl #4 (16)	110000	48

Reverse-engineering Tools

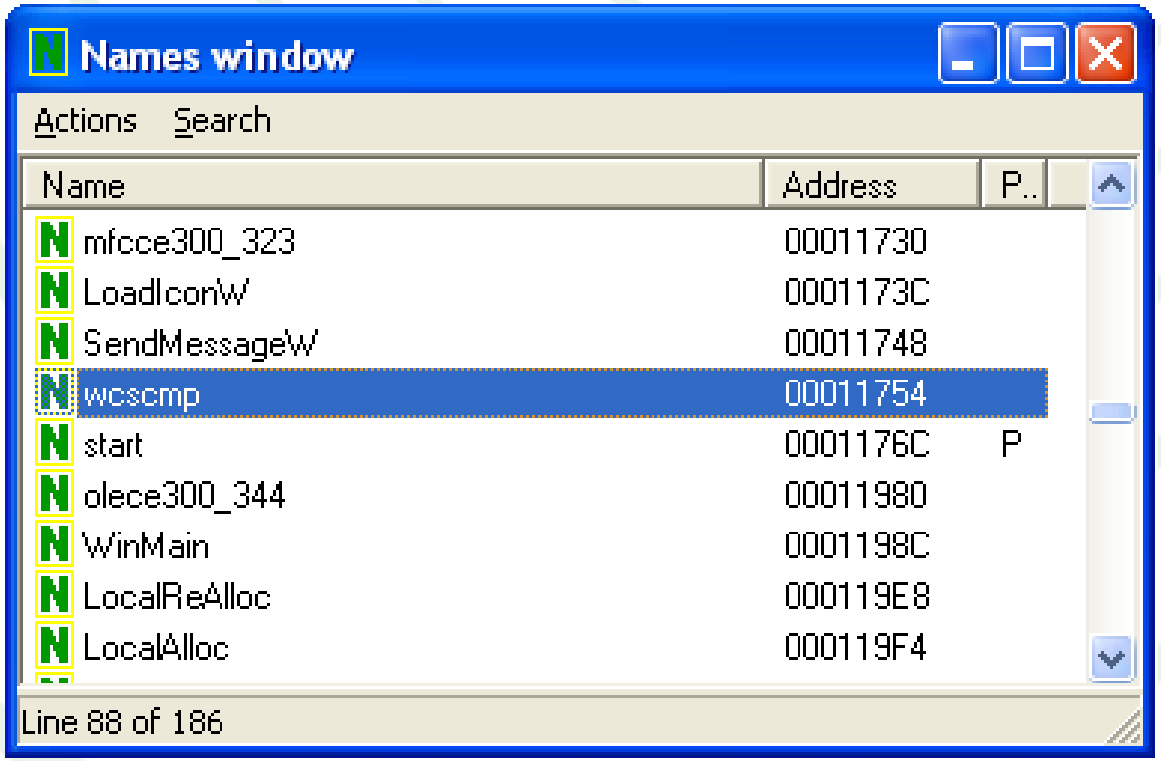
- **Hex Editor**
  - **Needed to make changes to program files**
  - **UltraEdit32**
- **Disassembler**
  - **Converts program file into ASM code**
  - **IDA Pro**
- **Debugger**
  - **USB connection SLOW! (Pocket Hosts + W/LAN)**
  - **Allows real time execution and walk through of code**
  - **Microsoft eMbedded Visual C++ 3.0**



Practical RVE - Entry

Airscanner

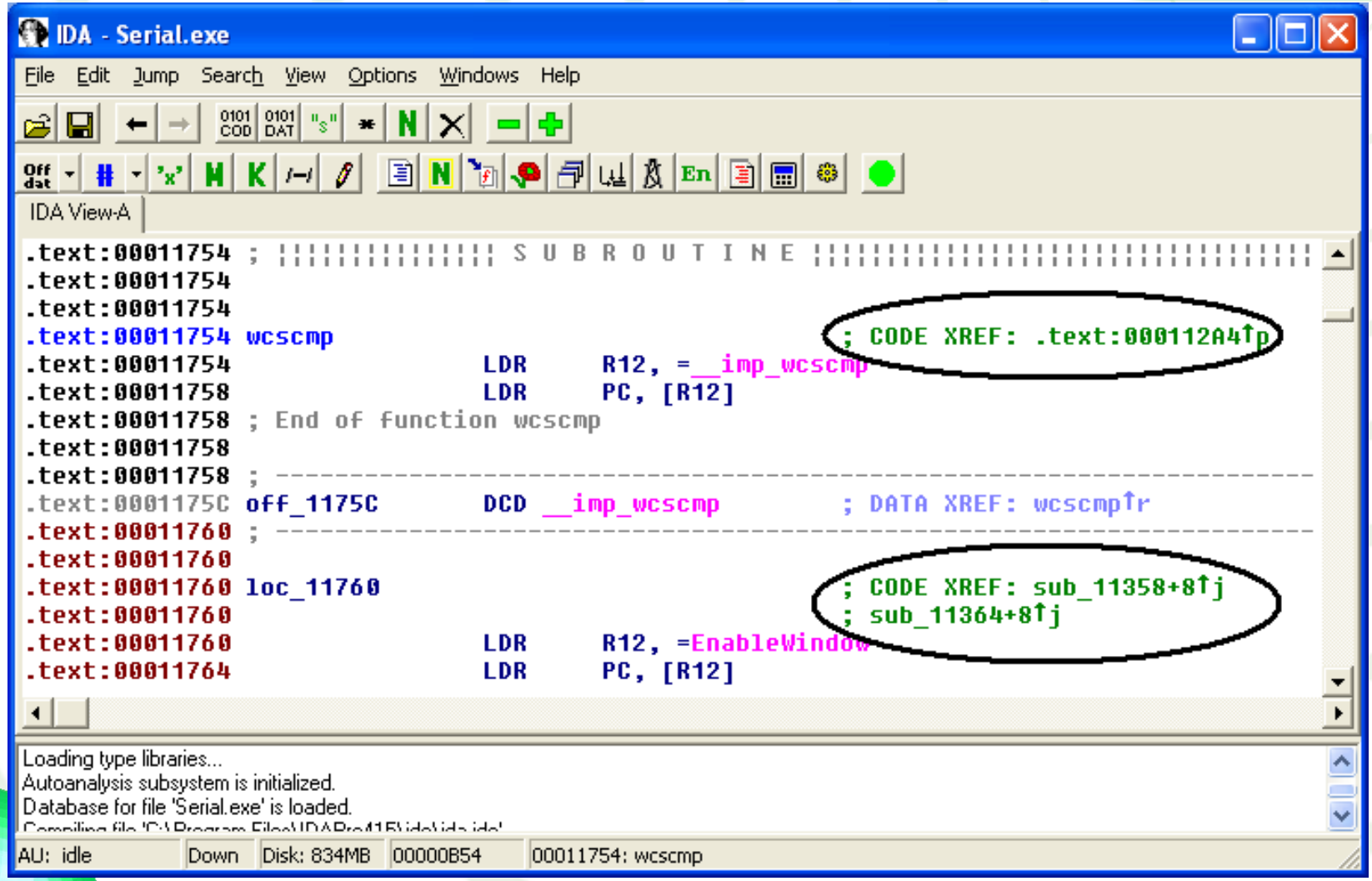
- **Load it in Disassembler**
- **Locate Needed Files!**
- **Note Names of Functions**
  - **LoadStringW**
  - **MessageBoxW**
  - **wcscmp**
  - **wcslen**



Practical RVE – Locate Example

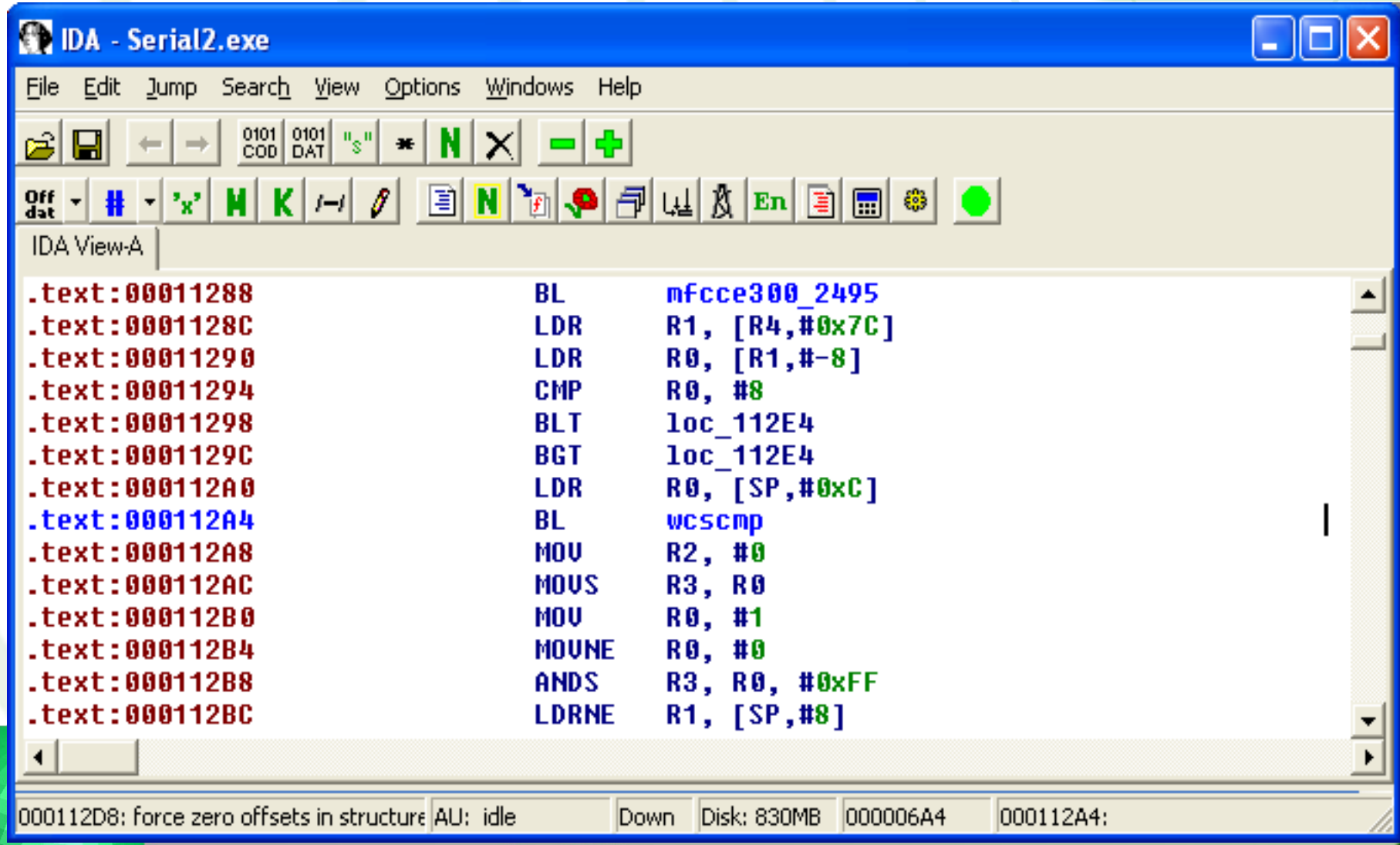
Practical RVE – Locating Weakness

Airscanner



Practical RVE – The Code

Airscanner



Practical RVE – The Code II

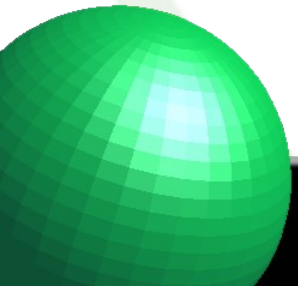





```
1.   LDR    R1, =unk_131A4
2.   ADD    R0, SP, #0xC
3.   BL     CString::operator=(us)
4.   LDR    R1, =unk_131B0
5.   ADD    R0, SP, #8
6.   BL     CString::operator=(us)
7.   LDR    R1, =unk_131E0
8.   ADD    R0, SP, #4
9.   BL     CString::operator=(us)
10.  LDR    R1, =unk_1321C
11.  ADD    R0, SP, #0
12.  BL     CString::operator=(us)
13.  MOV    R1, #1
14.  MOV    R0, R4
15.  BL     CWnd::UpdateData(int)
16.  LDR    R1, [R4,#0x7C]
17.  LDR    R0, [R1,#-8]
```

```
18.  CMP    R0, #8
19.  BLT    loc_112E4
20.  BGT    loc_112E4
21.  LDR    R0, [SP,#0xC]
22.  BL     wcscmp
23.  MOV    R2, #0
24.  MOVS           R3, R0
25.  MOV    R0, #1
26.  MOVNE          R0, #0
27.  ANDS   R3, R0, #0xFF
28.  LDRNE          R1, [SP,#8]
29.  MOV    R0, R4
30.  MOV    R3, #0
31.  BNE    loc_112F4
32.  LDR    R1, [SP,#4]
33.  B      loc_112F4
```

Practical RVE – The Code III



- **IDA Pro < 4.5**
  - **IDT files to convert function calls**
  - **http://www.dataworm.net/reverse/**
- **Load serial from program into memory**
  - **Typically serial is encrypted or dynamic**
  - **Most serial protection is useless**
- **Load two error messages into memory**
- **Load success messages into memory**
- **Get entered serial number**
- **Start validation check!**

Practical RVE – The Debugger

- **Load file**
  - **Copy PPC file to PC**
  - **Open PC file**
  - **Alter settings to copy it to PPC**
  - **Hit F11 to load**
  - **Set BREAKPOINTS**
    - **Use disassembler to determine BP value**
    - **0001127C → EVC 0x##01127C**
    - **BP value will change based on programs location in RAM**

Practical RVE – Program Walkthrough

1. Compare length
2. Branch depending on results
   - If length = 8 continue else jump to message
3. Compare serials
   - wcscmp returns 0 if false or 1 if true
4. Check results
   - Use AND calc to update status flag (Zero)
   - TRUE and TRUE = TRUE
5. Output message

1	1	0
1	0	0
1	0	0

Practical RVE – The 'Cracks' PI

- **Crack 1: Slight of Hand**
  - **CMP R0, #8**
    - **Compares two values**
    - **Updates status flags**
      - **Z=1 (True): Correct length**
      - **Z=0 (False): Incorrect Length**
    - **Alter it to make it always True**

CMP R0, #8 → CMP R0, R0

Practical RVE – The 'Cracks' PI…

- **Crack 1: Slight of Hand Cont.**
  - **MOVNE R0, #0**
    - **MOVS R3, R0**
    - **Moves 0 into R0 if Z = 0 else R0 remains 1**
    - **Alter it to make it update to 1**

MOVNE R0, #0 → MOVNE R0, #1

Practical RVE – The 'Cracks' PI…

- **Update Hex Value**
  - **Determine address from IDA**
  - **Open local file in Hex editor**
  - **Use address to locate point in code**
  - **Deduce the required change**
  - **Update Hex code**
  - **Save file and upload to PPC**

IDA Addr	Hex Addr	Orig Opcode	Org Hex	New Opcode	New Hex
0x11294	0x694	Cmp r0, #8	08 00 50 E3	Cmp r0, r0	00 00 50 E1
0x112B4	0x6B4	Monve r0, #0	00 00 A0 13	Movne r0, #1	01 00 A0 13

Practical RVE – The 'Cracks' PII

- **Crack 2: The Slide**
  - **CMP R0, #8 → Fails then Status flags are set.**
  - **BLT loc_112E4**
  - **BGT loc_112E4**
  - **Remove these lines**
  - **NOP Slide**
    - **Use in buffer overflow attacks**
    - **Handy for a space filler and to remove other values**

BLT loc_112E4 → NOP

BGT loc_112E4 → NOP

Practical RVE – The 'Cracks' PII…

- **Crack 2: The Slide Cont.**
  - **The Traditional 90**
    - **Doesn't work! UMULLLSS R9, R0, R0, R0**
    - **Unsigned Multiple Long if LS and update status flags**
  - **MOV R1, R1**
    - **Virtual NOP**
  - **Still have to patch actual serial check**

IDA Addr	Hex Addr	Orig Opcode	Org Hex	New Opcode	New Hex
0x11298 0x1129C	0x698 0x69C	BLT & BGT loc_112E4	11 00 00 BA 10 00 00 CA	Mov r1, r1 Mov r1, r1	01 10 A0 E1 01 10 A0 E1
0x112B4	0x6B4	Movne r0, #0	00 00 A0 13	Movne r0, #1	01 00 A0 13

Practical RVE – The 'Cracks' PIII

- **Crack 3: Preventative Maintenance**
  - **0x1128C sets R1 = entered serial**
  - **If R0 can be set to correct serial, why not R1?**
  - **Prevent a problem before it becomes one**

IDA Addr	Hex Addr	Orig Opcode	Org Hex	New Opcode	New Hex
0x1128C	0x68C	LDR R1, [R4, #0x7C]	7C 10 94 E5	LDR R1, [SP,#0xC]	0C 10 9D E5

The Real Code

```
strSerial="12345678";
strValid="Correct serial number. Thanks for registering.";
strInvalid="Incorrect serial number. Please contact technical support.";
strToShort="Incorrect serial number. Please verify it was typed it correctly.";
UpdateData(TRUE);
if ((m_Serial.GetLength() < 8) || (m_Serial.GetLength() > 8)){
    MessageBox(strToShort);
    }else{
    if (strSerial == m_Serial){
    MessageBox(strValid);
    }else{
    MessageBox(strInvalid);
}}}
```

Summary

- **Tools**
  - **Disassembler**
  - **Debugger**
  - **Hex Editor**
- **ARM Processor**
  - **Opcodes**
  - **Registers**
- **Reverse it**
  - **Locate weakness**
  - **Watch execution**
  - **Patch it**

References

- www.ka0s.net
- www.dataworm.net
- http://www.eecs.umich.edu/speech/docs/arm/ARM7TDMIvE.pdf
- http://www.ra.informatik.uni-stuttgart.de/~ghermanv/Lehre/SOC02/ARM_Presentation.pdf
- class.et.byu.edu/eet441/notes/arminst.ppt
- http://www.ngine.de/gbadoc/armref.pdf
- http://wheelie.tees.ac.uk/users/a.clements/ARMinfo/ARMnote.htm
- http://www3.mb.sympatico.ca/~reimann/andrew/asm/armref.pdf
- www.arm.com
- www.airscanner.com