



Aircanner



Embedded Reverse Engineering: Cracking Mobile Binaries

Seth Fogie @ Defcon 11

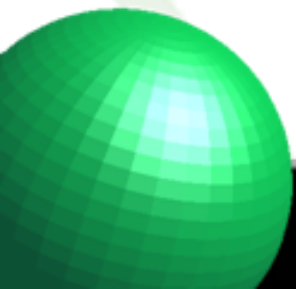
Who am I and what am I doing?



Aircanner



- **Aircanner.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**





- **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...



- **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**



- **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**



- **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



- **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**

- **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



Airscanner

Registers

```
R0 = 0ED04716 R1 = 00000000 R2 = 2206FEF8  
R3 = 00000005 R4 = 00011630 R5 = 0ED04716  
R6 = 00000000 R7 = 2206FEF8 R8 = 00011630  
R9 = 1E17FD40 R10 = 8COD4240  
R11 = 2206FEDC R12 = 01FA137C  
Sp = 2206FEBC Lr = 23FA17F8 Pc = 22011630  
Psr = 8000001F
```

```
Negative=1 Zero=0 Carry=0 Overflow=0
```

```
IRQ=0 FIQ=0 Thumb=0
```

```
Mode=F
```

ARM Opcodes – MOV, CMP



Airscanner

- **Move (MOV) – XX XX A0 EX**
 - MOV R3, R1: 01 30 A0 E**1**
 - MOV R2, #1: 01 20 A0 E**3**

- **Compare (CMP) – XX XX 5X EX**
 - CMP R2, R3: 03 00 52 E**1**
 - CMP R4, #1: 01 00 54 E**3**

ARM Status Flags



Airscanner

- **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



Aircanner



- 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



Airscanner



- **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)**



- **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**

- **STMED SPI, {R4,R5,LR}**

- **LDMED SPI, {R4,R5,LR}**

Shifting



Aircanner

- **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



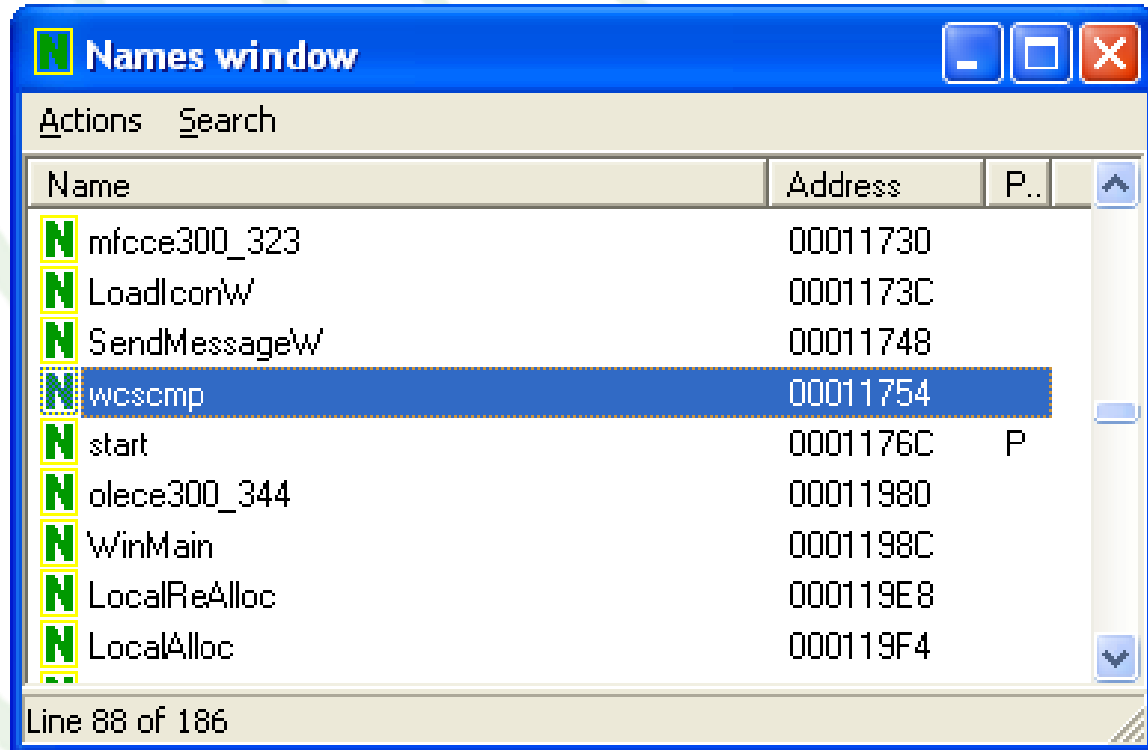
- **Hex Editor**
 - Needed to make changes to program files
 - UltraEdit32
- **Disassembler**
 - Converts program file into ASM code
 - IDA Pro
- **Debugger**
 - **USB connection SLOW!** (Pocket Hosts + W/LAN)
 - Allows real time execution and walk through of code
 - Microsoft eMbedded Visual C++ 3.0

Practical RVE - Entry



Aircanner

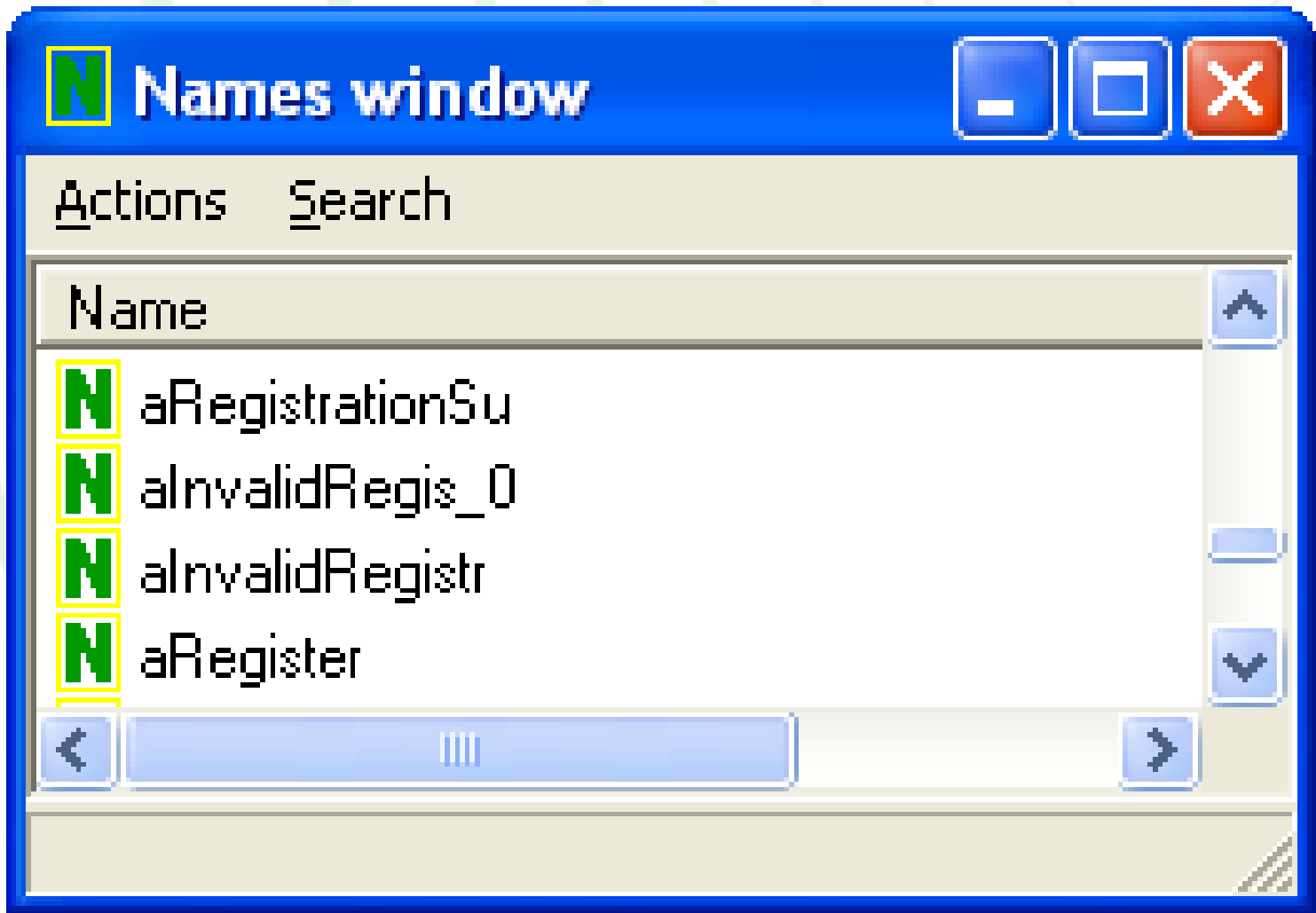
- **Load it in Disassembler**
- **Locate Needed Files!**
- **Note Names of Functions**
 - **LoadStringW**
 - **MessageBoxW**
 - **wcsmp**
 - **wcslen**



Practical RVE – Locate Example



Aircanner



Practical RVE – Locating Weakness



```
.text:00011754 ; :::::::::::::::::::: S U B R O U T I N E ::::::::::::::::::::
.text:00011754
.text:00011754
.text:00011754 wscmp ; CODE XREF: .text:000112A4↑p
.text:00011754          LDR    R12, =__imp_wscmp
.text:00011758          LDR    PC, [R12]
.text:00011758 ; End of function wscmp
.text:00011758 ; -----
.text:0001175C off_1175C          DCD    __imp_wscmp ; DATA XREF: wscmp↑r
.text:00011760 ; -----
.text:00011760 loc_11760
.text:00011760          ; CODE XREF: sub_11358+8↑j
.text:00011760          ; sub_11364+8↑j
.text:00011760          LDR    R12, =EnableWindow
.text:00011764          LDR    PC, [R12]
```

Loading type libraries...
Autoanalysis subsystem is initialized.
Database for file 'Serial.exe' is loaded.
Compiling file 'C:\Program Files\IDA Pro\IDe\ida\ida.exe'

AU: idle Down Disk: 834MB 00000B54 00011754: wscmp

Practical RVE – The Code



Airscanner

```
IDA View-A
.text:00011288      BL      mfcce300_2495
.text:0001128C      LDR     R1, [R4,#0x7C]
.text:00011290      LDR     R0, [R1,#-8]
.text:00011294      CMP     R0, #8
.text:00011298      BLT     loc_112E4
.text:0001129C      BGT     loc_112E4
.text:000112A0      LDR     R0, [SP,#0xC]
.text:000112A4      BL      wcscmp
.text:000112A8      MOU     R2, #0
.text:000112AC      MOUS   R3, R0
.text:000112B0      MOU     R0, #1
.text:000112B4      MOUNE   R0, #0
.text:000112B8      ANDS   R3, R0, #0xFF
.text:000112BC      LDRNE   R1, [SP,#8]

000112D8: force zero offsets in structure AU: idle      Down  Disk: 830MB  000006A4  000112A4:
```

Practical RVE – The Code II



Airscanner



1.	LDR	R1, =unk_131A4	18.	CMP	R0, #8
2.	ADD	R0, SP, #0xC	19.	BLT	loc_112E4
3.	BL	CString::operator=(us)	20.	BGT	loc_112E4
4.	LDR	R1, =unk_131B0	21.	LDR	R0, [SP,#0xC]
5.	ADD	R0, SP, #8	22.	BL	wcscmp
6.	BL	CString::operator=(us)	23.	MOV	R2, #0
7.	LDR	R1, =unk_131E0	24.	MOVS	R3, R0
8.	ADD	R0, SP, #4	25.	MOV	R0, #1
9.	BL	CString::operator=(us)	26.	MOVNE	R0, #0
10.	LDR	R1, =unk_1321C	27.	ANDS	R3, R0, #0xFF
11.	ADD	R0, SP, #0	28.	LDRNE	R1, [SP,#8]
12.	BL	CString::operator=(us)	29.	MOV	R0, R4
13.	MOV	R1, #1	30.	MOV	R3, #0
14.	MOV	R0, R4	31.	BNE	loc_112F4
15.	BL	CWnd::UpdateData(int)	32.	LDR	R1, [SP,#4]
16.	LDR	R1, [R4,#0x7C]	33.	B	loc_112F4
17.	LDR	R0, [R1,#-8]			



- **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!**



- **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**



1. Compare length

2. Branch depending on results

- If length = 8 continue else jump to message

3. Compare serials

- `wscmp` returns 0 if false or 1 if true

4. Check results

- Use AND calc to update status flag (Zero)
- TRUE and TRUE = TRUE

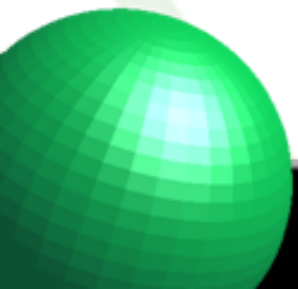
5. Output message

1	1	0
1	0	0
1	0	0



- **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**





- **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



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



- **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



- **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	0x698	BLT & BGT	11 00 00 BA	Mov r1, r1	01 10 A0 E1
0x1129C	0x69C	loc_112E4	10 00 00 CA	Mov r1, r1	01 10 A0 E1
0x112B4	0x6B4	Movne r0, #0	00 00 A0 13	Movne r0, #1	01 00 A0 13



- **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

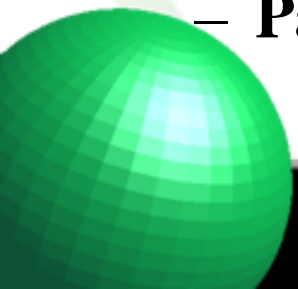


Aircanner

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




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



References



Aircanner



- 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