Reverse Engineering
ARM Based Devices

Job de Haas
ITSX
Overview

• What is ARM?
• Reversing and Decompiling.
• Idioms and examples.
• Conclusions.
What is ARM?

- Reduced Instruction Set Computer RISC.
- Designs and ARM Architecture licensed by ARM Ltd.
- Produced by: Intel, Philips, TI, Sharp, etc. etc.
- Popular cores: ARM7TDMI.
ARM cores and archs

• Architectures:
  – Specify instruction sets + extensions.

• Cores:
  – Eg. ARM7TDMI, ARM940T, ARM966E-S, ARM1022E.
  – Specify licensed IP cores.

• Families:
  – ARM7, ARM9, ARM9E, ARM10E, ARM11.
ARM core examples

- Nokia DCT-3 phones
  - ARM7TDMI + TMS320C54x

- HTC PDA with GSM
  - Intel Xscale
  - ARM7TDMI + TMS320C54x or
  - ARM9TDMI + TMS320C55x (OMAP)
What is running on ARM?

- Windows CE / PocketPC
- Symbian / EPOC
- Nucleus
- PalmOS
- Proprietary: older Nokia etc.
ARM Architecture

- ARM mode (32 bit).
- THUMB mode (16 bit).
- 16 Registers + Flags.
- Basic instruction model for most instructions:
  - Can choose to set flags.
  - Can choose to use conditions.
  - Can use the different addressing modes.
ARM addressing modes

MOV  R3, #0xA0
LDR  R4, [R0]
LDRH R11, [R4,R3]
STRB R11, [R0,#1]
ADD  R0, R4, #2
MOV  R3, R1,LSL#16
ADD  R0, R2, R3,LSL#8
STMFD SP!, {R4-R7,R11,LR}

LDR  R3, =DOC_func1
LDR  R6, off_90052208
MOV  R1, R0
MOV  PC, R3
MOVS R3, R2
MOVEQ R0, #1
MOVNE R0, #0
ARM / DSP Combo

- TMS320C54x is popular.
- Has internal RAM / ROM.
- Communicates through Dual Port RAM and signal lines (GPIO).
- Security can sometimes be compromised by running code in Dual Port RAM: reading out ROM and RAM.
DSP/ARM Block Diagram

- CLK PLL
- McBSP0
- McBSP1
- Timer
- MEM IF
- DMA
- SDRAM & SRAM IF
- 10/100 MAC (VC5471)
- C54x DSP (100 MIPS)
  - SARAM 56K words
  - DARAM 16K words (+API)
- ARM7TDMI RISC (47.5 MHz)
  - Ethernet State Machine (VC5471)
  - 16K-byte buffer
- RAM 16 KB
- CLK PLL
- Keypad IF
- GPIO
- SPI
- I²C
- UART
- UART IrDA
- Timers (2)
- Timer–WD
- JTAG

- Codec
- DSP SRAM (optional)
- RISC RAM/ROM
- LAN
- 8x8 keypad
- LEDs, etc.
- LCD display

Reverse Engineering
ARM Based Devices
Job de Haas
ITSX BV
BlackHat
Amsterdam 2004
Example reading out DSP

- Nokia DCT-3 phones allow flashing with custom ARM code.
- From ARM write DSP code at start address in dual port RAM.
- Reboot DSP.
- Read DATA and PROGRAM areas.
Overview

✓ What is ARM?
  • Reversing and Decompiling.
  • Idioms and examples.
  • Conclusions
Reversing: reaching the code

- Reading memory from a program.
- Using the bootloader / monitor.
- Unpacking upgrades.
- Dumping memory directly through hardware means.
Reversing issues

- Trying to understand the software.
- Device software is getting big: 4MB for a GSM phone to 32MB for WinCE.
- Production code lacks symbols.
- Most production code still contains a lot of debug features.
Hardware assisted RE

- JTAG Boundary Scan protocol.
- Still available in most consumer devices.
- Can be hard to trace out.
- Is supported with in-circuit debuggers and tracers.
Locating JTAG, cooking the PDA

http://www.xda-developers.com/jtag/
Trace the board
JTAG trace: connect and go
Using JTAG

- Reading memory on the fly.
- Modifying memory, but also inputs!
- In circuit ‘debugging’ of code.
- Proprietary extensions per manufacturer.
- Recognized as a security problem:
  - Test pads sometimes get removed.
  - Fuses may be used to destroy logic.
Tools

- Disassemblers
  - IDA Pro Advanced (for DSP)
  - Disarm
  - GNU

- Debuggers
  - Microsoft Embedded Visual Tools
  - RealView and ICE hardware debugging
  - GDB
Tools 2

- Emulators
  - ARMulator from ARM Ltd
  - SkyEye [http://www.skeyeye.org](http://www.skeyeye.org)
  - Nokia 5110 simulator WinArm

- Decompilers
  - Desquirr
Decompilation

- Generally based on work by Cristina Cifuentes.
- Signatures for subroutine / arg identification.
- Dataflow analysis.
- Data type analysis.
- Execution flow analysis.
Desquirr

• Plugin for IDA, developed by David Eriksson for his Master thesis.
• Does data flow analysis.
• Adapted for ARM instructions.
• Added ARM compiler idioms.
Overview

✓ What is ARM?
✓ Reversing and Decompiling.
• Idioms and examples.
• Conclusions
Idioms

• Compiler specific solutions and optimizations for common code.
• Can result in large simplifications of assembler.
• Can provide additional information to aid further understanding code.
• Used often with RISC instructions.
Idioms: function calls

```
STMFD   SP!, {R4-R11,LR}
SUB     SP, SP, #0x28
MOV     R5, R2
MOV     R2, #0
MOV     R4, R3
STR     R2, [SP,#0x4C+var_40]
MOV     R7, R0
LDR     R0, [SP,#0x4C+arg_0]
    LDR     R3, =DOC_func1
    MOV     R1, R0
    LDR     R3, [R3]
    MOV     R2, R4
    MOV     R0, R5
    MOV     LR, PC
    MOV     PC, R3
    ----------
    MOV     R0, #0
    LDMFD   SP!, {R4,R5,PC}
```
Idiom: type casting

**THUMB:**

MOV R0, #0x20  
ORR R0, R7  
LSL R0, R0, #0x18  
LSR R7, R0, #0x18

**ARM:**

ADD R1, R2, #1  
MOV R3, R1,LSL#16  
MOV R2, R3,LSR#16
Example: XDA lock protection

• Version 1: read lock code directly from memory through AT command.
• Look for lock related code.
• Look for device specific AT extensions.
• Result: lock is plain text readable by an AT command: AT%UREG?3FE00C,4
read_UREG:

@ DATA XREF: ROM:001AFD04

PUSH {R4,LR}
ADD R4, R1, #0
LDR R0, [R4]
LDR R1, =unk_3FE000
CMP R0, R1
BCC exit0
LDR R1, =unk_3FE007
CMP R0, R1
BHI exit0
LDR R1, [R4,#8]
ADD R2, R1, R0
LDR R3, =unk_3FE000
CMP R2, R3
BCC exit0
LDR R3, =unk_3FE007
CMP R2, R3
BLS ok

exit0:

@ CODE XREF: read UREG+A

MOV R0, #0
POP {R4,PC}

ok:

@ CODE XREF: read_UREG+20
obfuscate1:

```
MOV    R1, #2
ROR    R0, R1
LSL    R1, R0, #0x18
LSR    R1, R1, #0x18
LSR    R2, R0, #0x10
LSL    R2, R2, #0x18
LSR    R2, R2, #0x18
LSL    R2, R2, #8
ORR    R1, R2
LSL    R2, R1, #8
LSR    R1, R0, #0x18
LSL    R1, R1, #0x18
LSR    R1, R1, #0x18
ORR    R1, R2
LSL    R1, R1, #8
LSR    R0, R0, #8
LSL    R0, R0, #0x18
LSR    R0, R0, #0x18
ORR    R0, R1
MOV    R1, #0x1D
ROR    R0, R1
BX     LR
```
XDA lock protection 2

- Version 2: Blocks AT command and obfuscates lock.
- The AT%UREG address is blocked, but does not take ‘roll-over’ into account: AT%UREG?3FE004,FFFFFFFE
- Obfuscate is a simply reversible process.
From obfuscate2

...  

```
LSL     R2, R0, #0x1C            ; R2 = 0xC0000000
LSR     R2, R2, #0x1C           ; R2 = 0x0000000C
LSL     R2, R2, #0x18           ; R2 = 0x00000000
ORR     R2, R3                 ; R2 = 0x00C00DBA0A
STR     R2, [R1,#4]            ; Save the result
MOV     R2, #0x1D               
ROR     R0, R2                 ; assume R0 = 0x12345678
LDR     R2, =0x7D00039F         
STR     R2, [R1]               
LDR     R2, =0xE0A060           
STR     R2, [R1,#4]            ; Why throw away the result???
LSR     R2, R0, #4              ; R2 = 0x01234567
LSL     R5, R0, #4              ; R5 = 0x23456780
MOV     R4, #0xF0               
AND     R4, R5                 ; R4 = 0x00000080
LSL     R3, R2, #0x1C           ; R3 = 0x70000000
LSR     R6, R3, #0x1C           ; R6 = 0x00000007
LDR     R3, [R1]               
ORR     R6, R3                 
...```

Reverse Engineering  Job de Haas  BlackHat  
ARM Based Devices    ITSX BV       Amsterdam 2004
Usage:

```
rbmc [FileName [StartAddr [Len]]]
```

Read back the memory content from the specified address to the host and save the data to specified file name.

- **FileName**: Full file path for save data of memory (default=c:\temp\Mem.nb).
- **StartAddr**: Start address of memory (default(hex)=A0000000).
- **Len**: How many bytes will be read. And if not given value, it will be Total ROM size on board - ((StartAddress & 0xFFFFFFFF) - (ROM base address(0) & 0xFFFFFFFF)).

Examples:
- rchecksum 00000000 00000000
- rwdata 00040000 00000000
- rerase 00040000 00000000
- rrbmc 1.nb 00000000 00000000
XDA II lock protection 1

- Version 3: Uses new obfuscate and blocks AT command, but does allow bootloader access.
- Look for new bootloader commands: rrbmc test, 0x3fe302, 8
- Reverse obfuscation.
doSECURITY:

...                      key1       DCB "%Ag2gWp", 0x24
ADD R0, SP, #0x70       key2       DCB "5 (EvO^9,
ADD R1, SP, #0x60       key3       DCB "rG*344@T"
LDMIA R1!, {R2,R3}
STMIA R0!, {R2,R3}
ADD R0, SP, #0x70
LDR R1, =key1
BL DESdecrypt
ADD R0, SP, #0x70
ADD R1, SP, #0x68
MOV R2, #8
BL memcmp
CMP R0, #0
BNE notkey1
LDR R5, =valLOCKTIME
MOV R0, #0
STRB R0, [R5]
LDR R0, =isSetAllowed
MOV R1, #1
STRB R1, [R0]
MOV R4, #0xFF
B printvalue
obfuscate2:

... ORR R2, R3 ORR R4, R2 ORR R1, R2 LSR R0, R0, #8 LSR R0, R0, #8 LSR R0, R0, #0x1C LSL R0, R0, #0x14 LSL R0, R0, #8 ORR R0, R4 ORR R0, R1 STR R0, [R1,#4] STR R0, [R4,#4] POP {R4-R6,PC} MOV R0, R12

CMP R0, #1 BNE exit BNE exit
BL encodekey BL encodekey
ADD R0, R4, #0 ADD R0, R4, #0 LDR R1, =lockkey LDR R1, =lockkey
BL DES-decrypt BL DES-decrypt

exit:

ADD SP, SP, #4 ADD SP, SP, #4 POP {R4-R7,PC} POP {R4-R7,PC}
XDA II lock protection 2

- Version 4: blocks bootloader reading locks and improves obfuscation.
- Look at bootloader block.
- Look at obfuscate.
- Relook at locking code.
- Reverse AT%SECURITY code.
- Turn on GOD mode.
XDA II lock protection

• Version X?
• Example: Use MD5 hash with the IMEI and a long code.
• Prevent all write and read access to the lock area.
• Make sure all developers know what they are doing?
Overview

✓ What is ARM?
✓ Reversing and Decompiling.
✓ Idioms and examples.
• Conclusions
Conclusions

• ARM is a much used processor and very common in a lot of consumer devices.
• The simple RISC instruction set:
  – can lead to hard to follow, bitswapping code.
  – lends itself to decompilation.
• Gaining control over the lowlevel ARM code can lead to interesting possibilities.
Future outlook

- Improvements in decompilation will ease Reverse Engineering.
- Embedded systems still increase the unjustified feeling it will be ‘hard’ to break in to them.
- More and more developing for embedded systems becomes ‘easy’.
  \[\Rightarrow\] increase bad apps, increase attackers.
Resources

- PocketPC reversing:
  - http://www.ka0s.net
  - http://xda-developers.com

- Symbian reversing
  - http://phantasm.50megs.com

- Nokia reversing:
  - http://www.blacksphere.tk
  - http://nokiafree.org
  - http://www.mados-technology.com/mados/
Resources 2

- Decompilation
  - http://www.program-transformation.org/
  - http://boomerang.sourceforge.net/
  - http://desquirr.sourceforge.net/