100001001101001011011100

110000101110010011110010

0100000010000001011011100

110000101101100011110010

111001101101001011110011

SE 421 Spring 2020

Binary Analysis

SE 421 Spring 2020

Introduction

- Question: Does the control structure change from source code to binary? Can you verify code based on the disassembled binary?
- Walk through of work so far
    - How SE 421 can start to be applied in different settings

Approach

- Compare CFGs from source to CFGs from binary

Binary → $\boxed{\text{Disassembler}}$ → Decompiler → Source Code

Tool Chain Used:

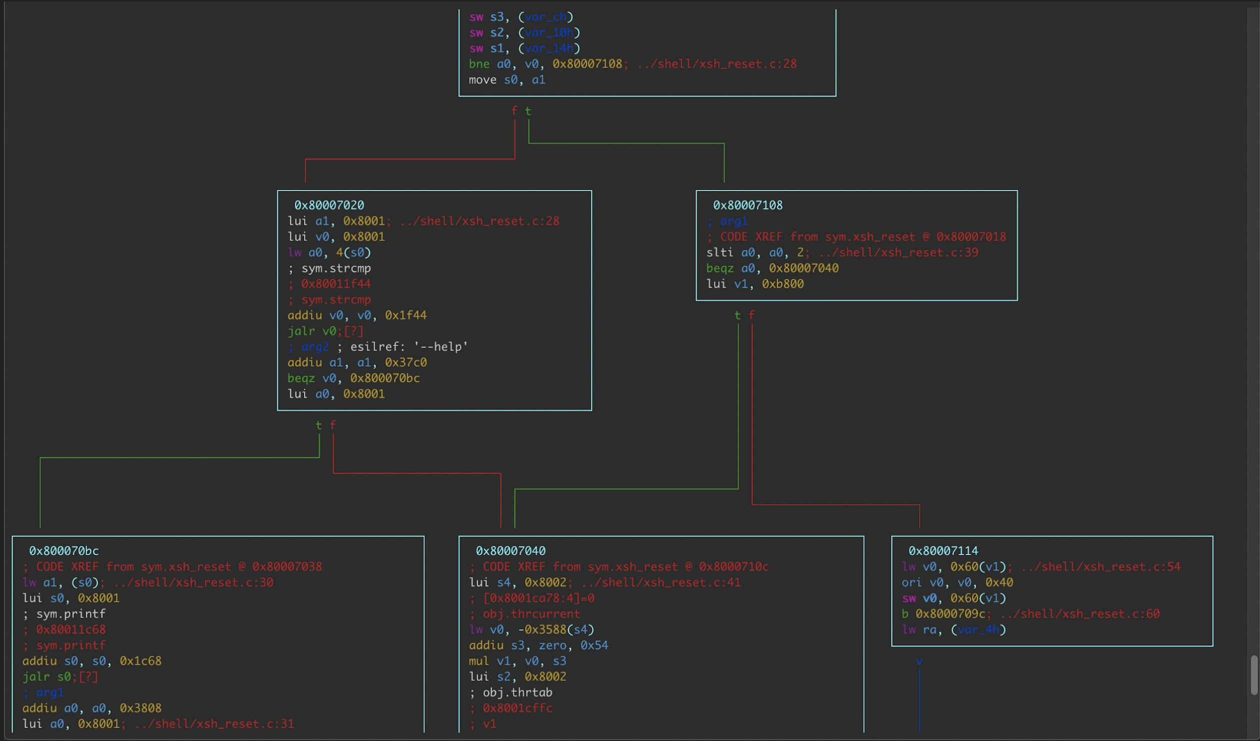
Compiled Xinu → Radare → Binary Analyzer using Atlas API → Atlas Graph DB → SE 421 Project + Other Tools

Test Case: XINU

- Working with XINU as a test case
- Slightly different than version used in class
- Allows for multiple things:
    - Source code is openly available
    - Already have verified the source
    - Verified source gives something for initial comparison

Disassembling the Binary

- Many tools available
  - Radare, open source
  - IDA Pro, $$$
  - Ghidra, developed by NSA
- Made use of Radare
  - Disassemble
  - Performs Analysis
  - Generates CFG's for the binary



```
sw s3, (var_ch)
sw s2, (var_10h)
sw s1, (var_14h)
bne a0, v0, 0x80007108; ../shell/xsh_reset.c:28
move s0, a1
```

```
 0x80007020
lui a1, 0x8001; ../shell/xsh_reset.c:28
lui v0, 0x8001
lw a0, 4(s0)
; sym.strcmp
; 0x80011f44
; sym.strcmp
addiu v0, v0, 0x1f44
jalr v0;[?]
; arg2 ; esilref: '--help'
addiu a1, a1, 0x37c0
beqz v0, 0x800070bc
lui a0, 0x8001
```

```
 0x80007108
; arg1
; CODE XREF from sym.xsh_reset @ 0x80007018
slti a0, a0, 2; ../shell/xsh_reset.c:39
beqz a0, 0x80007040
lui v1, 0xb800
```

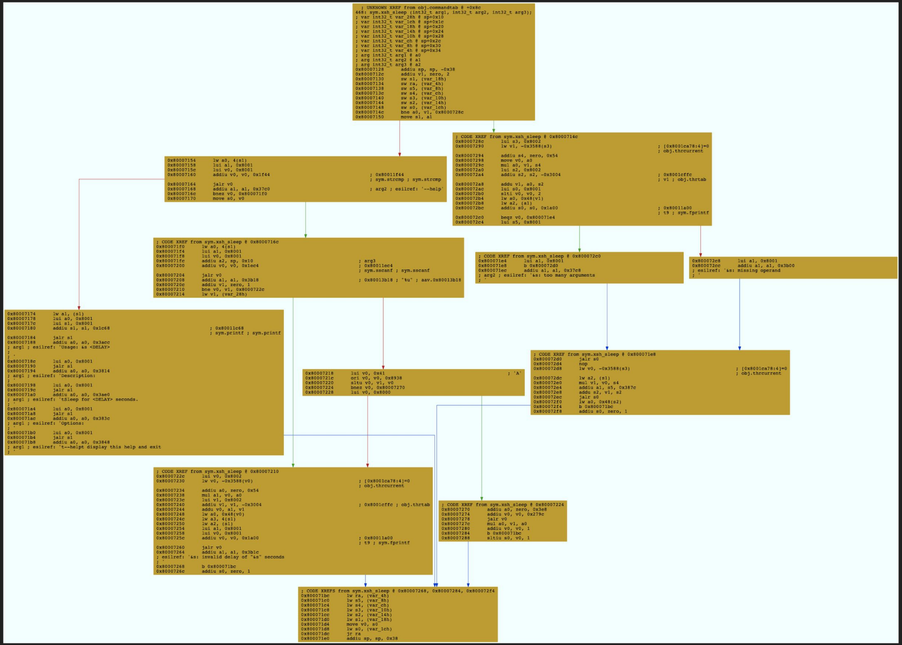
```
 0x800070bc
; CODE XREF from sym.xsh_reset @ 0x80007038
lw a1, (s0); ../shell/xsh_reset.c:30
lui s0, 0x8001
; sym.printf
; 0x80011c68
; sym.printf
addiu s0, s0, 0x1c68
jalr s0;[?]
; arg1
addiu a0, a0, 0x3808
lui a0, 0x8001; ../shell/xsh_reset.c:31
```

```
 0x80007040
; CODE XREF from sym.xsh_reset @ 0x8000710c
lui s4, 0x8002; ../shell/xsh_reset.c:41
; [0x8001ca78:4]=0
; obj.thrcurrent
lw v0, -0x3588(s4)
addiu s3, zero, 0x54
mul v1, v0, s3
lui s2, 0x8002
; obj.thrtab
; 0x8001cffc
; v1
```

```
 0x80007114
lw v0, 0x60(v1); ../shell/xsh_reset.c:54
ori v0, v0, 0x40
sw v0, 0x60(v1)
b 0x8000709c; ../shell/xsh_reset.c:60
lw ra, (var_4h)
```

CLI- Not Great. What Next?

- Radare allows you to export generated CFGs
- Little to no functionality or usability outside of better visualization



```
  ; UNKNOWN XREF from obj.commandtab @ +0x8c
468: sym.xsh_sleep (int32_t arg1, int32_t arg2, int32_t arg3);
; var int32_t var_28h @ sp+0x10
; var int32_t var_1ch @ sp+0x1c
; var int32_t var_18h @ sp+0x20
; var int32_t var_14h @ sp+0x24
; var int32_t var_10h @ sp+0x28
; var int32_t var_ch @ sp+0x2c
; var int32_t var_8h @ sp+0x30
; var int32_t var_4h @ sp+0x34
; arg int32_t arg1 @ a0
; arg int32_t arg2 @ a1
; arg int32_t arg3 @ a2
0x80007128      addiu sp, sp, -0x38
0x8000712c      addiu v1, zero, 2
0x80007130      sw s1, (var_18h)
0x80007134      sw ra, (var_4h)
0x80007138      sw s5, (var_8h)
0x8000713c      sw s4, (var_ch)
0x80007140      sw s3, (var_10h)
0x80007144      sw s2, (var_14h)
0x80007148      sw s0, (var_1ch)
0x8000714c      bne a0, v1, 0x8000728c
0x80007150      move s1, a1
```

```
; CODE XREF from sym.xsh_sleep @ 0x8000714c
0x8000728c      lui s3, 0x8002
0x80007290      lw v1, -0x3588(s3)                                        ; [0x8001ca78:4]=0
                                                                          ; obj.thrcurrent
0x80007294      addiu s4, zero, 0x54
0x80007298      move v0, a0
0x8000729c      mul a0, v1, s4
0x800072a0      lui s2, 0x8002
0x800072a4      addiu s2, s2, -0x3004                                     ; 0x8001cffc
                                                                          ; v1 ; obj.thrtab
0x800072a8      addu v1, a0, s2
0x800072ac      lui s0, 0x8001
0x800072b0      slti v0, v0, 2
0x800072b4      lw a0, 0x48(v1)
0x800072b8      lw a2, (a1)
0x800072bc      addiu s0, s0, 0x1a00                                      ; 0x80011a00
                                                                          ; t9 ; sym.fprintf
0x800072c0      beqz v0, 0x800071e4
0x800072c4      lui s5, 0x8001
```

```
0x80007154      lw a0, 4(s1)
0x80007158      lui a1, 0x8001
0x8000715c      lui v0, 0x8001
0x80007160      addiu v0, v0, 0x1f44                                      ; 0x80011f44
                                                                          ; sym.strcmp ; sym.strcmp
0x80007164      jalr v0
0x80007168      addiu a1, a1, 0x37c0                                      ; arg2 ; esilref: '--help'
0x8000716c      bnez v0, 0x800071f0
0x80007170      move s0, v0
```

```
; CODE XREF from sym.xsh_sleep @ 0x8000716c
0x800071f0      lw a0, 4(s1)
0x800071f4      lui a1, 0x8001
0x800071f8      lui v0, 0x8001
0x800071fc      addiu a2, sp, 0x10                                        ; arg3
0x80007200      addiu v0, v0, 0x1ec4                                      ; 0x80011ec4
                                                                          ; sym.sscanf ; sym.sscanf
0x80007204      jalr v0
0x80007208      addiu a1, a1, 0x3b18                                      ; 0x80013b18 ; "%u" ; aav.0x80013b18
0x8000720c      addiu v1, zero, 1
0x80007210      bne v0, v1, 0x8000722c
0x80007214      lw v1, (var_28h)
```

```
; CODE XREF from sym.xsh_sleep @ 0x800072c0
0x800071e4      lui a1, 0x8001
0x800071e8      b 0x800072d0
0x800071ec      addiu a1, a1, 0x37c8
; arg2 ; esilref: '&s: too many arguments
; '
```

```
0x800072c8      lui a1, 0x8001
0x800072cc      addiu a1, a1, 0x3b00
; esilref: '&s: missing operand
; '
```

```
0x80007174      lw a1, (s1)
0x80007178      lui a0, 0x8001
0x8000717c      lui s1, 0x8001
0x80007180      addiu s1, s1, 0x1c68                                      ; 0x80011c68
                                                                          ; sym.printf ; sym.printf
0x80007184      jalr s1
0x80007188      addiu a0, a0, 0x3acc
; arg1 ; esilref: 'Usage: &s <DELAY>
;  .
;
0x8000718c      lui a0, 0x8001
0x80007190      jalr s1
0x80007194      addiu a0, a0, 0x3814
; arg1 ; esilref: 'Description:
; '
0x80007198      lui a0, 0x8001
0x8000719c      jalr s1
0x800071a0      addiu a0, a0, 0x3ae0
; arg1 ; esilref: 'tSleep for <DELAY> seconds.
; '
0x800071a4      lui a0, 0x8001
0x800071a8      jalr s1
0x800071ac      addiu a0, a0, 0x383c
; arg1 ; esilref: 'Options:
; '
0x800071b0      lui a0, 0x8001
0x800071b4      jalr s1
0x800071b8      addiu a0, a0, 0x3848
; arg1 ; esilref: 't--helpt display this help and exit
; '
```

```
0x80007218      lui v0, 0x41                                              ; 'A'
0x8000721c      ori v0, v0, 0x8938
0x80007220      sltu v0, v1, v0
0x80007224      bnez v0, 0x80007270
0x80007228      lui v0, 0x8000
```

```
; CODE XREF from sym.xsh_sleep @ 0x800071e8
0x800072d0      jalr s0
0x800072d4      nop
0x800072d8      lw v0, -0x3588(s3)                                        ; [0x8001ca78:4]=0
                                                                          ; obj.thrcurrent
0x800072dc      lw a2, (s1)
0x800072e0      mul v1, v0, s4
0x800072e4      addiu a1, s5, 0x387c
0x800072e8      addu s2, v1, s2
0x800072ec      jalr s0
0x800072f0      lw a0, 0x48(s2)
0x800072f4      b 0x800071bc
0x800072f8      addiu s0, zero, 1
```

```
; CODE XREF from sym.xsh_sleep @ 0x80007210
0x8000722c      lui v0, 0x8002
0x80007230      lw v0, -0x3588(v0)                                        ; [0x8001ca78:4]=0
                                                                          ; obj.thrcurrent
0x80007234      addiu a0, zero, 0x54
0x80007238      mul a1, v0, a0
0x8000723c      lui v1, 0x8002
0x80007240      addiu v1, v1, -0x3004                                     ; 0x8001cffc ; obj.thrtab
0x80007244      addu v0, a1, v1
0x80007248      lw a0, 0x48(v0)
0x8000724c      lw a3, 4(s1)
0x80007250      lw a2, (s1)
0x80007254      lui a1, 0x8001
0x80007258      lui v0, 0x8001
0x8000725c      addiu v0, v0, 0x1a00                                      ; 0x80011a00
                                                                          ; t9 ; sym.fprintf
0x80007260      jalr v0
0x80007264      addiu a1, a1, 0x3b1c
; esilref: '&s: invalid delay of "&s" seconds
; '
0x80007268      b 0x800071bc
0x8000726c      addiu s0, zero, 1
```

```
; CODE XREF from sym.xsh_sleep @ 0x80007224
0x80007270      addiu a0, zero, 0x3e8
0x80007274      addiu v0, v0, 0x279c
0x80007278      jalr v0
0x8000727c      mul a0, v1, a0
0x80007280      addiu v0, v0, 1
0x80007284      b 0x800071bc
0x80007288      sltiu s0, v0, 1
```

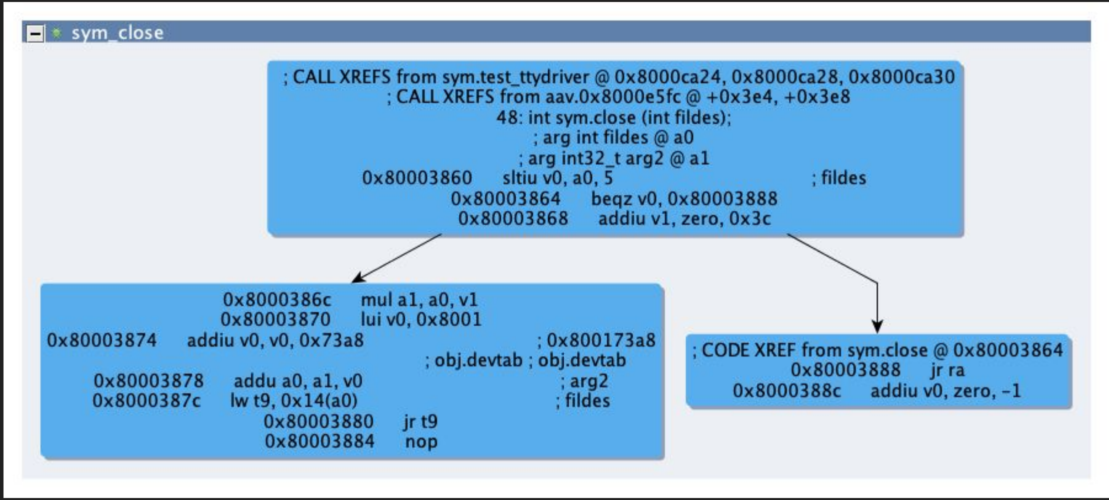
```
; CODE XREFS from sym.xsh_sleep @ 0x80007268, 0x80007284, 0x800072f4
0x800071bc      lw ra, (var_4h)
0x800071c0      lw s5, (var_8h)
0x800071c4      lw s4, (var_ch)
0x800071c8      lw s3, (var_10h)
0x800071cc      lw s2, (var_14h)
0x800071d0      lw s1, (var_18h)
0x800071d4      move v0, s0
0x800071d8      lw s0, (var_1ch)
0x800071dc      jr ra
0x800071e0      addiu sp, sp, 0x38
```

Enter: Atlas

- Parse the generated graph files to create Atlas graphs
- Opens up the ability to use Atlas analyzers
  - Reuse C and Java code analyzer for binary
- Improved visualization and usability
  - More interactive
- Allows for side-by-side comparison to source code

Example 1: Basic Comparison

- Function: `close.c`
    - Found in the system folder
- Used my tool to load Radare data into Atlas
- Points of Interest:
    - # of Nodes: 3
    - # of Edges: 2
    - # of Paths: 2



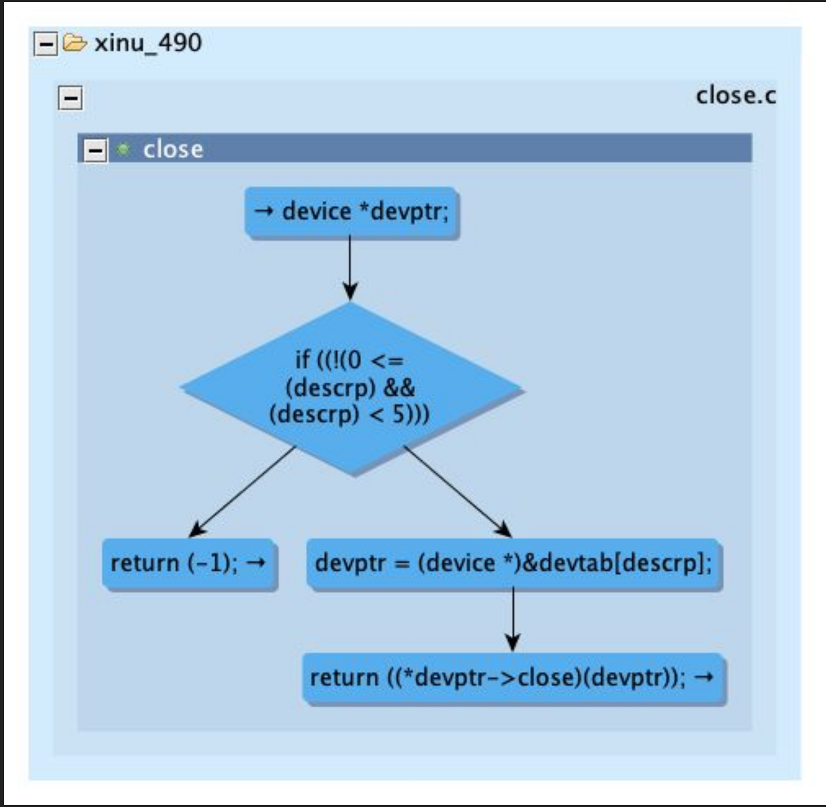
```
; CALL XREFS from sym.test_ttydriver @ 0x8000ca24, 0x8000ca28, 0x8000ca30
; CALL XREFS from aav.0x8000e5fc @ +0x3e4, +0x3e8
48: int sym.close (int fildes);
; arg int fildes @ a0
; arg int32_t arg2 @ a1
0x80003860      sltiu v0, a0, 5                    ; fildes
0x80003864      beqz v0, 0x80003888
0x80003868      addiu v1, zero, 0x3c
```

```
0x8000386c      mul a1, a0, v1
0x80003870      lui v0, 0x8001
0x80003874      addiu v0, v0, 0x73a8               ; 0x800173a8
                                   ; obj.devtab ; obj.devtab
0x80003878      addu a0, a1, v0                    ; arg2
0x8000387c      lw t9, 0x14(a0)                    ; fildes
0x80003880      jr t9
0x80003884      nop
```

```
; CODE XREF from sym.close @ 0x80003864
0x80003888      jr ra
0x8000388c      addiu v0, zero, -1
```

Example 1: Basic Comparison

- Source CFG for `close.c`
- Points of Interest:
  - # of Nodes: 5
  - # of Edges: 4
  - # of Paths: 2

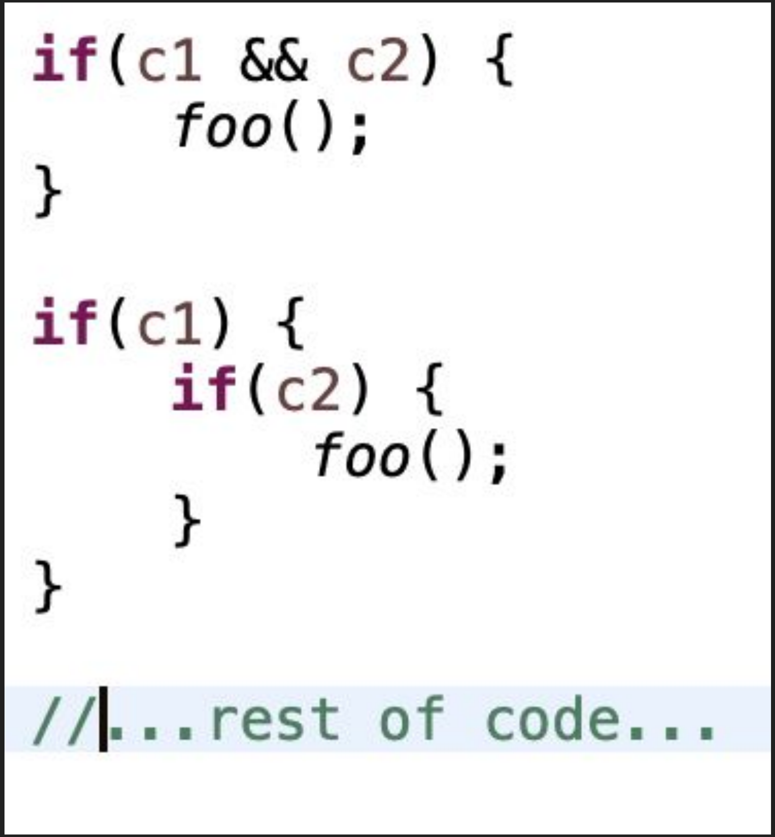


xinu_490
close.c
close
→ device *devptr;
if ((!(0 <= (descrp) && (descrp) < 5)))
return (-1); →
devptr = (device *)&devtab[descrp];
return ((*devptr->close)(devptr)); →

Is this always going to be the case?

Short Circuiting

- Only evaluate as much of an expression as you have to
- Compiler does this by breaking up conditionals
- Implications of this can be seen in XINU



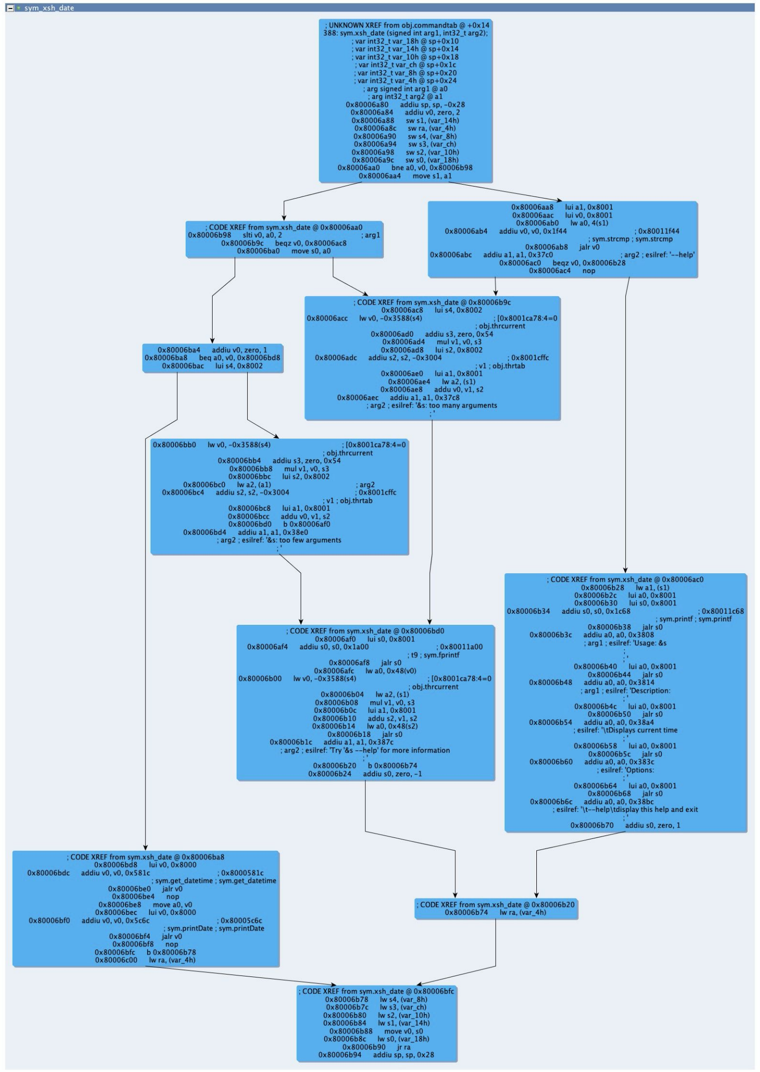
```
if(c1 && c2) {
    foo();
}

if(c1) {
    if(c2) {
        foo();
    }
}

//...rest of code...
```

Xinu Short Circuit

- Pick another function and check those results
  - Chose `xsh_date.c`
- Loaded the Radare CFG into Atlas
- Points of Interest:
  - # of Nodes: 11
  - # of Edges: 14
  - # of Paths: 5



```
; UNKNOWN XREF from obj.commandtab @ +0x14
388: sym.xsh_date (signed int arg1, int32_t arg2);
; var int32_t var_18h @ sp+0x10
; var int32_t var_14h @ sp+0x14
; var int32_t var_10h @ sp+0x18
; var int32_t var_ch @ sp+0x1c
; var int32_t var_8h @ sp+0x20
; var int32_t var_4h @ sp+0x24
; arg signed int arg1 @ a0
; arg int32_t arg2 @ a1
0x80006a80     addiu sp, sp, -0x28
0x80006a84     addiu v0, zero, 2
0x80006a88     sw s1, (var_14h)
0x80006a8c     sw ra, (var_4h)
0x80006a90     sw s4, (var_8h)
0x80006a94     sw s3, (var_ch)
0x80006a98     sw s2, (var_10h)
0x80006a9c     sw s0, (var_18h)
0x80006aa0     bne a0, v0, 0x80006b98
0x80006aa4     move s1, a1
```

```
; CODE XREF from sym.xsh_date @ 0x80006aa0
0x80006b98     slti v0, a0, 2                     ; arg1
0x80006b9c     beqz v0, 0x80006ac8
0x80006ba0     move s0, a0
```

```
0x80006aa8     lui a1, 0x8001
0x80006aac     lui v0, 0x8001
0x80006ab0     lw a0, 4(s1)
0x80006ab4     addiu v0, v0, 0x1f44               ; 0x80011f44
                                                  ; sym.strcmp ; sym.strcmp
0x80006ab8     jalr v0
0x80006abc     addiu a1, a1, 0x37c0               ; arg2 ; esilref: '--help'
0x80006ac0     beqz v0, 0x80006b28
0x80006ac4     nop
```

```
; CODE XREF from sym.xsh_date @ 0x80006b9c
0x80006ac8     lui s4, 0x8002
0x80006acc     lw v0, -0x3588(s4)                 ; [0x8001ca78:4=0
                                                  ; obj.thrcurrent
0x80006ad0     addiu s3, zero, 0x54
0x80006ad4     mul v1, v0, s3
0x80006ad8     lui s2, 0x8002
0x80006adc     addiu s2, s2, -0x3004              ; 0x8001cffc
                                                  ; v1 ; obj.thrtab
0x80006ae0     lui a1, 0x8001
0x80006ae4     lw a2, (s1)
0x80006ae8     addu v0, v1, s2
0x80006aec     addiu a1, a1, 0x37c8
; arg2 ; esilref: '&s: too many arguments
; '
```

```
0x80006ba4     addiu v0, zero, 1
0x80006ba8     beq a0, v0, 0x80006bd8
0x80006bac     lui s4, 0x8002
```

```
0x80006bb0     lw v0, -0x3588(s4)                 ; [0x8001ca78:4=0
                                                  ; obj.thrcurrent
0x80006bb4     addiu s3, zero, 0x54
0x80006bb8     mul v1, v0, s3
0x80006bbc     lui s2, 0x8002
0x80006bc0     lw a2, (a1)                        ; arg2
0x80006bc4     addiu s2, s2, -0x3004              ; 0x8001cffc
                                                  ; v1 ; obj.thrtab
0x80006bc8     lui a1, 0x8001
0x80006bcc     addu v0, v1, s2
0x80006bd0     b 0x80006af0
0x80006bd4     addiu a1, a1, 0x38e0
; arg2 ; esilref: '&s: too few arguments
; '
```

```
; CODE XREF from sym.xsh_date @ 0x80006bd0
0x80006af0     lui s0, 0x8001
0x80006af4     addiu s0, s0, 0x1a00               ; 0x80011a00
                                                  ; t9 ; sym.fprintf
0x80006af8     jalr s0
0x80006afc     lw a0, 0x48(v0)
0x80006b00     lw v0, -0x3588(s4)                 ; [0x8001ca78:4=0
                                                  ; obj.thrcurrent
0x80006b04     lw a2, (s1)
0x80006b08     mul v1, v0, s3
0x80006b0c     lui a1, 0x8001
0x80006b10     addu s2, v1, s2
0x80006b14     lw a0, 0x48(s2)
0x80006b18     jalr s0
0x80006b1c     addiu a1, a1, 0x387c
; arg2 ; esilref: 'Try '&s --help' for more information
; '
0x80006b20     b 0x80006b74
0x80006b24     addiu s0, zero, -1
```

```
; CODE XREF from sym.xsh_date @ 0x80006ac0
0x80006b28     lw a1, (s1)
0x80006b2c     lui a0, 0x8001
0x80006b30     lui s0, 0x8001
0x80006b34     addiu s0, s0, 0x1c68               ; 0x80011c68
                                                  ; sym.printf ; sym.printf
0x80006b38     jalr s0
0x80006b3c     addiu a0, a0, 0x3808
; arg1 ; esilref: 'Usage: &s
;
; '
0x80006b40     lui a0, 0x8001
0x80006b44     jalr s0
0x80006b48     addiu a0, a0, 0x3814
; arg1 ; esilref: 'Description:
; '
0x80006b4c     lui a0, 0x8001
0x80006b50     jalr s0
0x80006b54     addiu a0, a0, 0x38a4
; esilref: '\tDisplays current time
; '
0x80006b58     lui a0, 0x8001
0x80006b5c     jalr s0
0x80006b60     addiu a0, a0, 0x383c
; esilref: 'Options:
; '
0x80006b64     lui a0, 0x8001
0x80006b68     jalr s0
0x80006b6c     addiu a0, a0, 0x38bc
; esilref: '\t--help\tdisplay this help and exit
; '
0x80006b70     addiu s0, zero, 1
```

```
; CODE XREF from sym.xsh_date @ 0x80006ba8
0x80006bd8     lui v0, 0x8000
0x80006bdc     addiu v0, v0, 0x581c               ; 0x8000581c
                                                  ; sym.get_datetime ; sym.get_datetime
0x80006be0     jalr v0
0x80006be4     nop
0x80006be8     move a0, v0
0x80006bec     lui v0, 0x8000
0x80006bf0     addiu v0, v0, 0x5c6c               ; 0x80005c6c
                                                  ; sym.printDate ; sym.printDate
0x80006bf4     jalr v0
0x80006bf8     nop
0x80006bfc     b 0x80006b78
0x80006c00     lw ra, (var_4h)
```

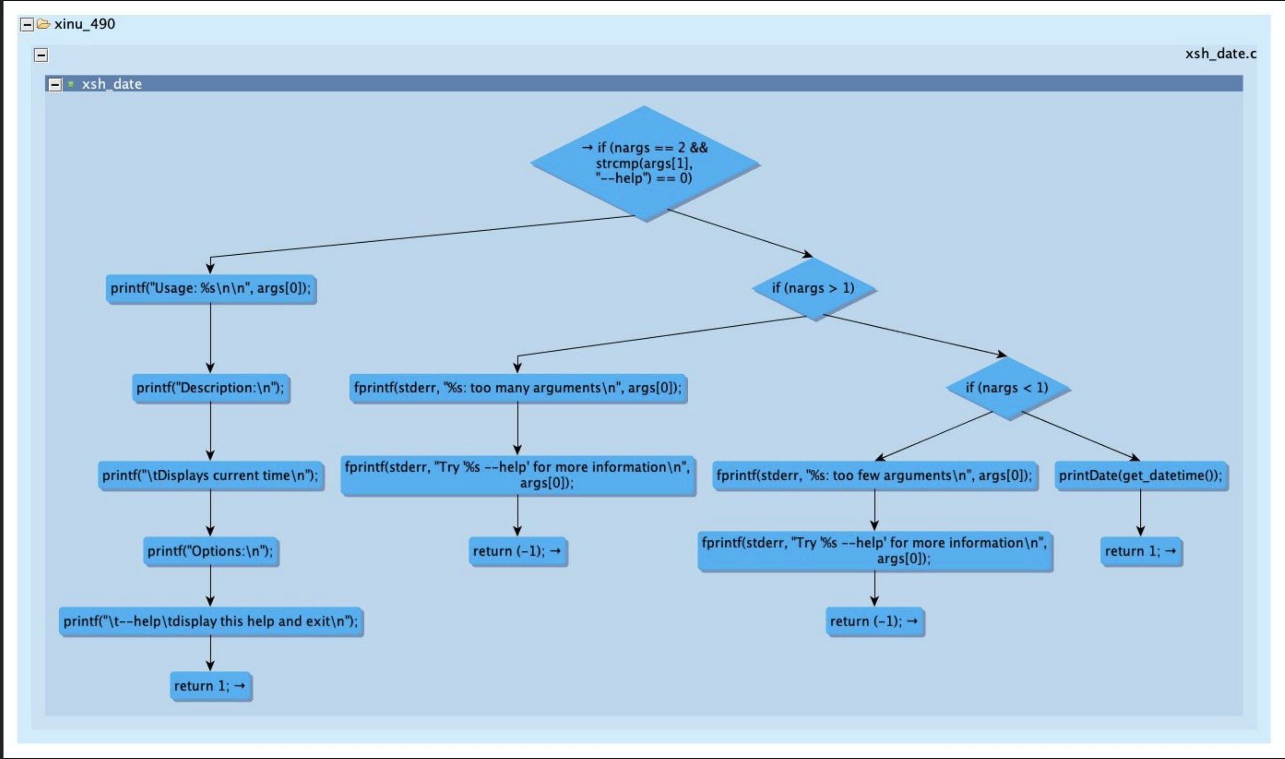
```
; CODE XREF from sym.xsh_date @ 0x80006b20
0x80006b74     lw ra, (var_4h)
```

```
; CODE XREF from sym.xsh_date @ 0x80006bfc
0x80006b78     lw s4, (var_8h)
0x80006b7c     lw s3, (var_ch)
0x80006b80     lw s2, (var_10h)
0x80006b84     lw s1, (var_14h)
0x80006b88     move v0, s0
0x80006b8c     lw s0, (var_18h)
0x80006b90     jr ra
0x80006b94     addiu sp, sp, 0x28
```

Source CFG: `xsh_date.c`

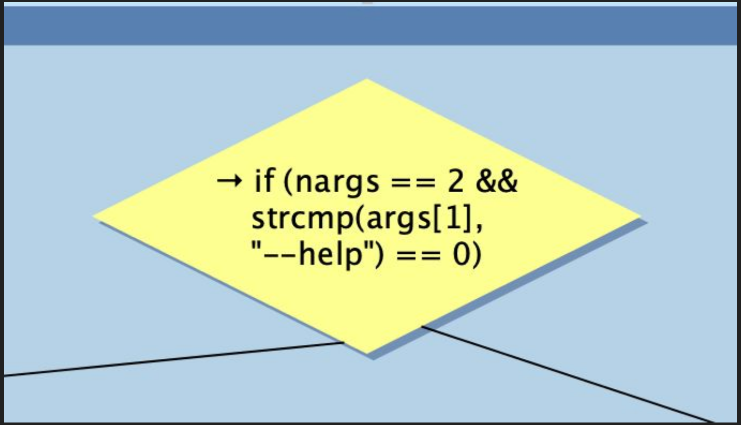
Points of Interest:

- # of Nodes: 17
- # of Edges: 16
- # of Paths: **4**

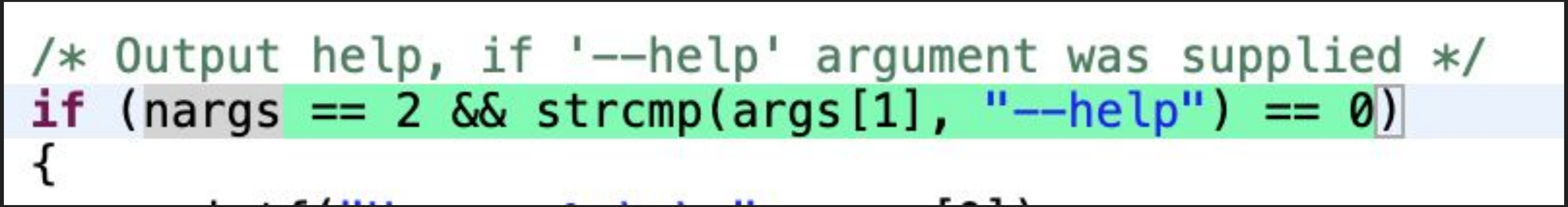

xinu_490
xsh_date.c
xsh_date
→ if (nargs == 2 && strcmp(args[1], "--help") == 0)
printf("Usage: %s\n\n", args[0]);
if (nargs > 1)
printf("Description:\n");
fprintf(stderr, "%s: too many arguments\n", args[0]);
if (nargs < 1)
printf("\tDisplays current time\n");
fprintf(stderr, "Try '%s --help' for more information\n", args[0]);
fprintf(stderr, "%s: too few arguments\n", args[0]);
printDate(get_datetime());
printf("Options:\n");
return (-1); →
fprintf(stderr, "Try '%s --help' for more information\n", args[0]);
return 1; →
printf("\t--help\tdisplay this help and exit\n");
return (-1); →
return 1; →

Analysis

- `xsh_date` disassembled binary has 1 more path
    - Source: 4
    - Binary: 5
- Why?



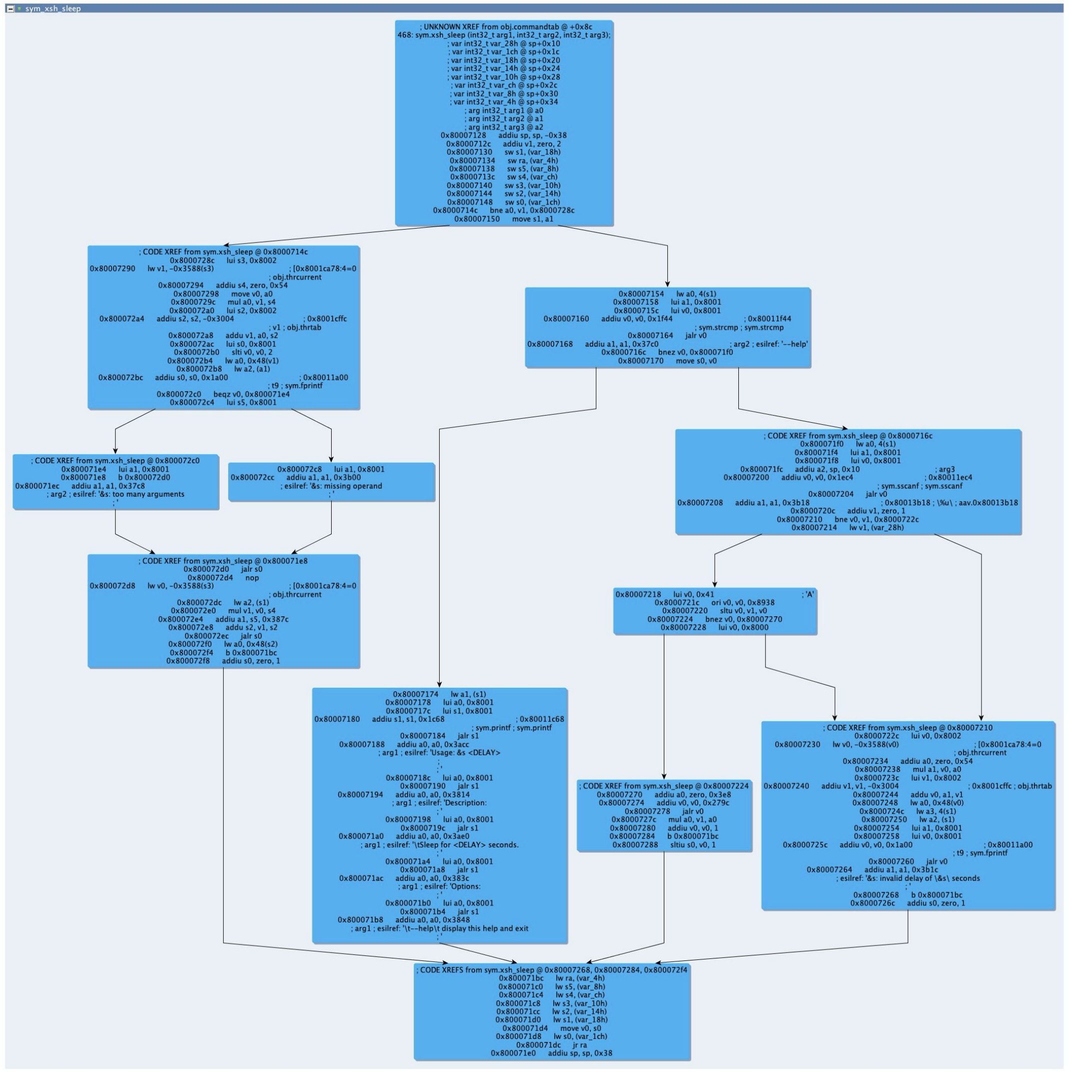
```
→ if (nargs == 2 &&
strcmp(args[1],
"--help") == 0)
```



```
/* Output help, if '--help' argument was supplied */
if (nargs == 2 && strcmp(args[1], "--help") == 0)
{
```

Where to next?

- Pick a function and go from ther
  - Started with `xsh_sleep.c`
- Same Radare CFG, but now loaded into Atlas
- Points of Interest:
  - # of Nodes: 12
  - # of Edges: 25
  - # of Paths: 6



```
; UNKNOWN XREF from obj.commandtab @ +0x8c
468: sym.xsh_sleep (int32_t arg1, int32_t arg2, int32_t arg3);
; var int32_t var_28h @ sp+0x10
; var int32_t var_1ch @ sp+0x1c
; var int32_t var_18h @ sp+0x20
; var int32_t var_14h @ sp+0x24
; var int32_t var_10h @ sp+0x28
; var int32_t var_ch @ sp+0x2c
; var int32_t var_8h @ sp+0x30
; var int32_t var_4h @ sp+0x34
; arg int32_t arg1 @ a0
; arg int32_t arg2 @ a1
; arg int32_t arg3 @ a2
0x80007128      addiu sp, sp, -0x38
0x8000712c      addiu v1, zero, 2
0x80007130      sw s1, (var_18h)
0x80007134      sw ra, (var_4h)
0x80007138      sw s5, (var_8h)
0x8000713c      sw s4, (var_ch)
0x80007140      sw s3, (var_10h)
0x80007144      sw s2, (var_14h)
0x80007148      sw s0, (var_1ch)
0x8000714c      bne a0, v1, 0x8000728c
0x80007150      move s1, a1
```

```
; CODE XREF from sym.xsh_sleep @ 0x8000714c
0x8000728c      lui s3, 0x8002
0x80007290      lw v1, -0x3588(s3)                ; [0x8001ca78:4=0
                                                  ; obj.thrcurrent
0x80007294      addiu s4, zero, 0x54
0x80007298      move v0, a0
0x8000729c      mul a0, v1, s4
0x800072a0      lui s2, 0x8002
0x800072a4      addiu s2, s2, -0x3004             ; 0x8001cffc
                                                  ; v1 ; obj.thrtab
0x800072a8      addu v1, a0, s2
0x800072ac      lui s0, 0x8001
0x800072b0      slti v0, v0, 2
0x800072b4      lw a0, 0x48(v1)
0x800072b8      lw a2, (a1)
0x800072bc      addiu s0, s0, 0x1a00              ; 0x80011a00
                                                  ; t9 ; sym.fprintf
0x800072c0      beqz v0, 0x800071e4
0x800072c4      lui s5, 0x8001
```

```
0x80007154      lw a0, 4(s1)
0x80007158      lui a1, 0x8001
0x8000715c      lui v0, 0x8001
0x80007160      addiu v0, v0, 0x1f44              ; 0x80011f44
                                                  ; sym.strcmp ; sym.strcmp
0x80007164      jalr v0
0x80007168      addiu a1, a1, 0x37c0              ; arg2 ; esilref: '--help'
0x8000716c      bnez v0, 0x800071f0
0x80007170      move s0, v0
```

```
; CODE XREF from sym.xsh_sleep @ 0x8000716c
0x800071f0      lw a0, 4(s1)
0x800071f4      lui a1, 0x8001
0x800071f8      lui v0, 0x8001
0x800071fc      addiu a2, sp, 0x10                ; arg3
0x80007200      addiu v0, v0, 0x1ec4              ; 0x80011ec4
                                                  ; sym.sscanf ; sym.sscanf
0x80007204      jalr v0
0x80007208      addiu a1, a1, 0x3b18              ; 0x80013b18 ; \%u\ ; aav.0x80013b18
0x8000720c      addiu v1, zero, 1
0x80007210      bne v0, v1, 0x8000722c
0x80007214      lw v1, (var_28h)
```

```
; CODE XREF from sym.xsh_sleep @ 0x800072c0
0x800071e4      lui a1, 0x8001
0x800071e8      b 0x800072d0
0x800071ec      addiu a1, a1, 0x37c8
; arg2 ; esilref: '&s: too many arguments
; '
```

```
0x800072c8      lui a1, 0x8001
0x800072cc      addiu a1, a1, 0x3b00
; esilref: '&s: missing operand
; '
```

```
0x80007218      lui v0, 0x41                      ; 'A'
0x8000721c      ori v0, v0, 0x8938
0x80007220      sltu v0, v1, v0
0x80007224      bnez v0, 0x80007270
0x80007228      lui v0, 0x8000
```

```
; CODE XREF from sym.xsh_sleep @ 0x800071e8
0x800072d0      jalr s0
0x800072d4      nop
0x800072d8      lw v0, -0x3588(s3)                ; [0x8001ca78:4=0
                                                  ; obj.thrcurrent
0x800072dc      lw a2, (s1)
0x800072e0      mul v1, v0, s4
0x800072e4      addiu a1, s5, 0x387c
0x800072e8      addu s2, v1, s2
0x800072ec      jalr s0
0x800072f0      lw a0, 0x48(s2)
0x800072f4      b 0x800071bc
0x800072f8      addiu s0, zero, 1
```

```
0x80007174      lw a1, (s1)
0x80007178      lui a0, 0x8001
0x8000717c      lui s1, 0x8001
0x80007180      addiu s1, s1, 0x1c68              ; 0x80011c68
                                                  ; sym.printf ; sym.printf
0x80007184      jalr s1
0x80007188      addiu a0, a0, 0x3acc
; arg1 ; esilref: 'Usage: &s <DELAY>
;
; '
0x8000718c      lui a0, 0x8001
0x80007190      jalr s1
0x80007194      addiu a0, a0, 0x3814
; arg1 ; esilref: 'Description:
; '
0x80007198      lui a0, 0x8001
0x8000719c      jalr s1
0x800071a0      addiu a0, a0, 0x3ae0
; arg1 ; esilref: '\tSleep for <DELAY> seconds.
; '
0x800071a4      lui a0, 0x8001
0x800071a8      jalr s1
0x800071ac      addiu a0, a0, 0x383c
; arg1 ; esilref: 'Options:
; '
0x800071b0      lui a0, 0x8001
0x800071b4      jalr s1
0x800071b8      addiu a0, a0, 0x3848
; arg1 ; esilref: '\t--help\t display this help and exit
; '
```

```
; CODE XREF from sym.xsh_sleep @ 0x80007224
0x80007270      addiu a0, zero, 0x3e8
0x80007274      addiu v0, v0, 0x279c
0x80007278      jalr v0
0x8000727c      mul a0, v1, a0
0x80007280      addiu v0, v0, 1
0x80007284      b 0x800071bc
0x80007288      sltiu s0, v0, 1
```

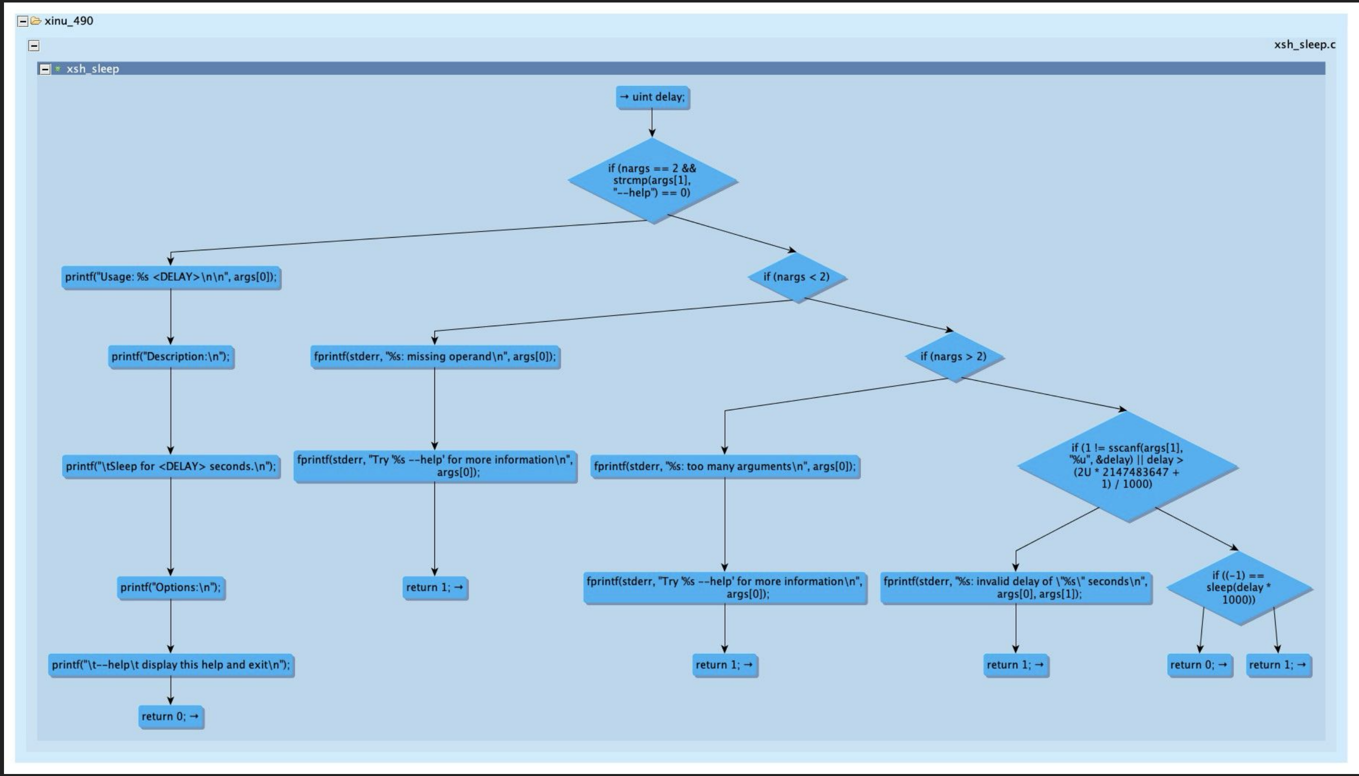
```
; CODE XREF from sym.xsh_sleep @ 0x80007210
0x8000722c      lui v0, 0x8002
0x80007230      lw v0, -0x3588(v0)                ; [0x8001ca78:4=0
                                                  ; obj.thrcurrent
0x80007234      addiu a0, zero, 0x54
0x80007238      mul a1, v0, a0
0x8000723c      lui v1, 0x8002
0x80007240      addiu v1, v1, -0x3004             ; 0x8001cffc ; obj.thrtab
0x80007244      addu v0, a1, v1
0x80007248      lw a0, 0x48(v0)
0x8000724c      lw a3, 4(s1)
0x80007250      lw a2, (s1)
0x80007254      lui a1, 0x8001
0x80007258      lui v0, 0x8001
0x8000725c      addiu v0, v0, 0x1a00              ; 0x80011a00
                                                  ; t9 ; sym.fprintf
0x80007260      jalr v0
0x80007264      addiu a1, a1, 0x3b1c
; esilref: '&s: invalid delay of \&s\ seconds
; '
0x80007268      b 0x800071bc
0x8000726c      addiu s0, zero, 1
```

```
; CODE XREFS from sym.xsh_sleep @ 0x80007268, 0x80007284, 0x800072f4
0x800071bc      lw ra, (var_4h)
0x800071c0      lw s5, (var_8h)
0x800071c4      lw s4, (var_ch)
0x800071c8      lw s3, (var_10h)
0x800071cc      lw s2, (var_14h)
0x800071d0      lw s1, (var_18h)
0x800071d4      move v0, s0
0x800071d8      lw s0, (var_1ch)
0x800071dc      jr ra
0x800071e0      addiu sp, sp, 0x38
```

Source CFG

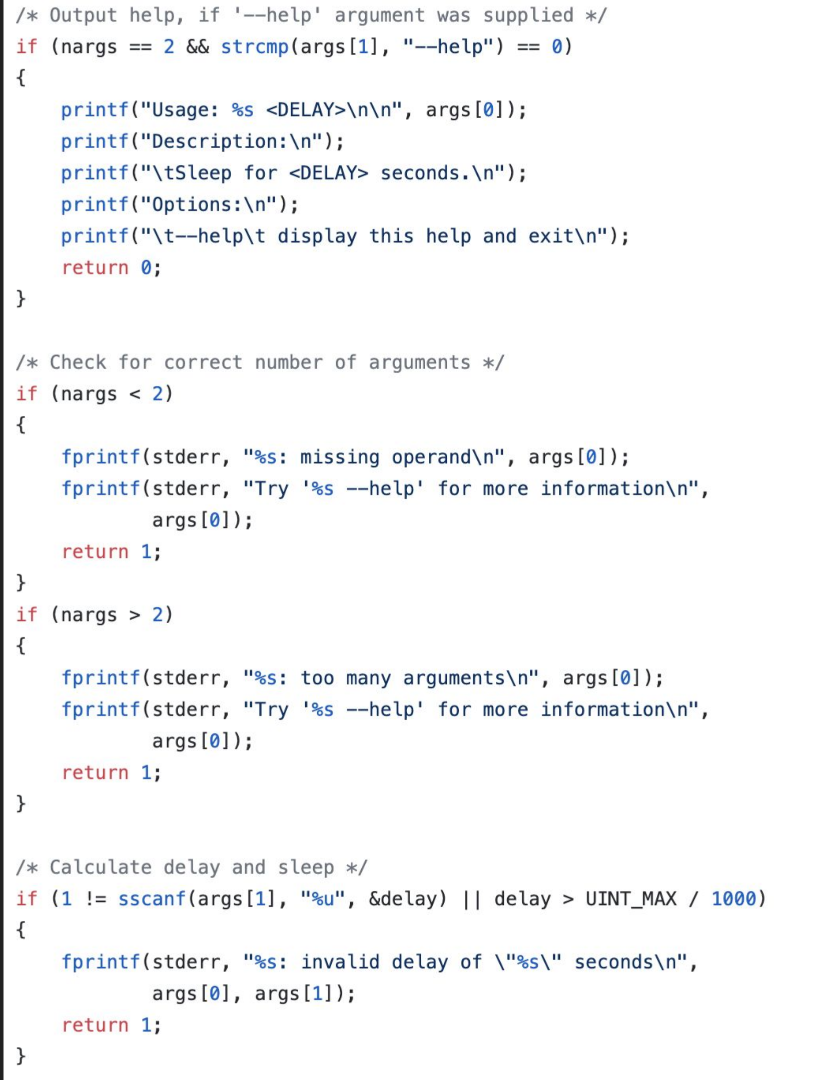
Points of Interest:

- # of Nodes: 22
- # of Edges: 21
- # of Paths: 6


xinu_490
xsh_sleep.c
xsh_sleep
→ uint delay;
if (nargs == 2 && strcmp(args[1], "--help") == 0)
printf("Usage: %s <DELAY>\n\n", args[0]);
if (nargs < 2)
printf("Description:\n");
fprintf(stderr, "%s: missing operand\n", args[0]);
if (nargs > 2)
printf("\tSleep for <DELAY> seconds.\n");
fprintf(stderr, "Try '%s --help' for more information\n", args[0]);
fprintf(stderr, "%s: too many arguments\n", args[0]);
if (1 != sscanf(args[1], "%u", &delay) || delay > (2U * 2147483647 + 1) / 1000)
printf("Options:\n");
return 1; →
fprintf(stderr, "Try '%s --help' for more information\n", args[0]);
fprintf(stderr, "%s: invalid delay of \"%s\" seconds\n", args[0], args[1]);
if ((-1) == sleep(delay * 1000))
printf("\t--help\t display this help and exit\n");
return 1; →
return 1; →
return 0; →
return 1; →
return 0; →

Analysis

- Compiler is able to short circuit and optimize based on first condition
- Break up the first condition into short circuit as we already saw
- If nargs == 2 → True, but strcmp != 0
    - Skip the next two conditionals
    - This optimizes the code



```c
/* Output help, if '--help' argument was supplied */
if (nargs == 2 && strcmp(args[1], "--help") == 0)
{
    printf("Usage: %s <DELAY>\n\n", args[0]);
    printf("Description:\n");
    printf("\tSleep for <DELAY> seconds.\n");
    printf("Options:\n");
    printf("\t--help\t display this help and exit\n");
    return 0;
}

/* Check for correct number of arguments */
if (nargs < 2)
{
    fprintf(stderr, "%s: missing operand\n", args[0]);
    fprintf(stderr, "Try '%s --help' for more information\n",
            args[0]);
    return 1;
}
if (nargs > 2)
{
    fprintf(stderr, "%s: too many arguments\n", args[0]);
    fprintf(stderr, "Try '%s --help' for more information\n",
            args[0]);
    return 1;
}

/* Calculate delay and sleep */
if (1 != sscanf(args[1], "%u", &delay) || delay > UINT_MAX / 1000)
{
    fprintf(stderr, "%s: invalid delay of \"%s\" seconds\n",
            args[0], args[1]);
    return 1;
}
```

Putting It All Together

- Able to apply the same knowledge learned in SE 421 to this research project
- Can use the same tools used in SE 421
    - Path Counter Project
    - Atlas Shell + API
- Allows for more investigation into legacy code or binary analysis of things like malware

What Next?

- Automate more of functionality
  - Use SE 421 Path Counter to compare source paths to binary paths
- Build Data Flow Analysis
- Test against additional source + binary combinations
- Investigate Stripped Binaries

Thank You

Questions?