Cosc 69.16: Reverse Engineering

Prishita Dharampal

Problem 1. For this lab assignment, you will be taking the provided assembly code listing and writing the equivalent in pseudocode followed by a few questions about the listing. The pseudocode doesn't have to compile, but it should be understandable to someone who knows C.

Listing

The following listing was generated using `objdump -M intel -D <file>`. The first column is the instruction virtual address. The next column is the machine code, and the final column is the disassembled instruction.

```
0000000000001149 <sub_1149>:
1149: f3 0f 1e fa        endbr64
114d: 55                 push rbp
114e: 48 89 e5           mov rbp, rsp
1151: 89 7d fc           mov DWORD PTR [rbp-0x4], edi
1154: 89 75 f8           mov DWORD PTR [rbp-0x8], esi
1157: 8b 55 fc           mov edx, DWORD PTR [rbp-0x4]
115a: 8b 45 f8           mov eax, DWORD PTR [rbp-0x8]
115d: 01 d0              add eax, edx
115f: 5d                 pop rbp
1160: c3                 ret
```

```
0000000000001161 <sub_1161>:
1161: f3 0f 1e fa        endbr64
1165: 55                 push rbp
1166: 48 89 e5           mov rbp, rsp
1169: 48 83 ec 08        sub rsp, 0x8
116d: 89 7d fc           mov DWORD PTR [rbp-0x4], edi
1170: 81 7d fc ff 00 00 00
                         cmp DWORD PTR [rbp-0x4], 0xff
1177: 7e 11              jle 118a <sub_1161+0x29>
1179: 8b 45 fc           mov eax, DWORD PTR [rbp-0x4]
117c: be ad de 00 00     mov esi, 0xdead
1181: 89 c7              mov edi, eax
1183: e8 c1 ff ff ff     call 1149 <sub_1149>
1188: eb 0f              jmp 1199 <sub_1161+0x38>
118a: b8 fe ca 00 00     mov eax, 0xcafe
118f: 8b 75 fc           mov esi, DWORD PTR [rbp-0x4]
1192: 89 c7              mov edi, eax
1194: e8 b0 ff ff ff     call 1149 <sub_1149>
1199: c9                 leave
119a: c3                 ret
```

1. Were there any instructions that you hadn't seen before?
2. Using the Intel Instruction Set Reference, give a brief explanation of what they do.
3. What is the return value if the argument to the `sub_1161` function is `0x50`? `0x200`?

Solution.

Reconstructed Pseudocode:

```
int sub_1149(int a, int b) {
    return a + b;
}
```

```
int sub_1161(int a) {
    if (a <= 0xff){
        return sub_1149(a, 0xcafe);
    }
    else {
        return sub_1149(0xdead, a);
    }

}
```

1. I wasn't familiar with the `endbr64` instruction.

2. According to Intel's Website, the processor implements a state machine that tracks indirect `JMP` and `CALL` instructions. When `endbr64` is seen, the state machine moves from `IDLE` to `WAIT_FOR_ENDBRANCH` state.

3. The return value for `sub_1161(0x50)` is `0xcafe + 0x50 = 0xcb4e`. The return value for `sub_1161(0x200)` is `0xdead + 0x200 = 0xe0ad`.