Week 7 Tuesday
Assembly II

Fill in the check-in form on cs107a.stanford.edu!

https://twitter.com/programmersmeme/status/1108073617983156226
git clone /afs/ir/class/archive/cs/cs107a/cs107a.1226/WWW/exercises/assembly2
Announcements

● Get extra credit for filling out an evaluation: https://edstem.org/us/courses/20849/discussion/1480721
● CS 107 lecture schedule
  ○ Done with assembly, Friday lecture on assign5, no lecture Monday!
● Midterm grading finishing up this week
● 2nd round of 1:1s, sign up NOW
  ○ https://calendly.com/adbenson/cs107a-1-1
● Please fill out a mid-quarter evaluation for CS 107A
  ○ Link on CS 107A homepage or https://forms.gle/1swzDjS4y6nB7SfX6
Unix Tip Spotlight

- **Spooking your friends on the Myth machines via `write`**
  - You and a friend must be on the same Myth machine
  - Can specify a Myth machine like
    ```
    ssh adbenson@myth66.stanford.edu
    ```
  - To initiate a message, `write <sunet>`
  - Type into the terminal, and it’ll be sent line-by-line to the other user
  - `<CTRL+D>` to end (EOF)
  - Kind of annoying, don’t do if your friend is trying to make a deadline
Agenda

- x86 Conditional and Unconditional Jumps
- Practical Advice for Assembly Puzzles
- nanobomb1 walkthrough
- nanobomb2
x86 Conditional and Unconditional Jumps
Jumps with cmp (jump.c)

```
0x555555555229 <main>   endbr64
0x55555555522d <main+4> push  %rbx
0x55555555522e <main+5> mov   $0x0, %eax
0x555555555233 <main+10> callq 0x5555555553c7 <risky_read_int>
0x555555555238 <main+15> mov   %eax, %ebx
0x55555555523a <main+17> cmp   $0x4, %eax
0x55555555523d <main+20> jg   0x55555555524d <main+36>
0x55555555523f <main+22> mov   $0x0, %eax
0x555555555244 <main+27> callq 0x5555555553c7 <risky_read_int>
0x555555555249 <main+32> add   %ebx, %eax
0x55555555524b <main+34> pop    %rbx
0x55555555524c <main+35> retq
0x55555555524d <main+36> mov   $0x0, %eax
0x555555555252 <main+41> callq 0x5555555553c7 <risky_read_int>
0x555555555257 <main+46> imul   %ebx, %eax
0x55555555525a <main+49> jmp   0x55555555524b <main+34>
```

Jumps with `cmp` (`jump.c`)
Jumps with `cmp` *(jump.c)*

- **Unconditional jumps**
  - `jmp` - always jump to the specified address

- **Conditional jumps**
  - Lots of types - jump to specified address only if condition directly prior and referenced by `cmp` is true
  - Ex) `jg`, `je`, `jne`, `jle`, `jl`, ...
  - Example of how to read these:
    - `cmp $0x4,%eax; jg 0x555555555524d <main+36>`
    - ^Read this as “jump to main+36 if %eax is greater than 4”
Conditional Jumps without `cmp` *(shift_jump.c)*

```assembly
0x55555555522d <main+4>      push   %rbx
0x55555555522e <main+5>      mov    $0x0,%eax
0x555555555233 <main+10>     callq  0x5555555553c7 <risky_read_int>
0x555555555238 <main+15>     mov    %eax,%ebx
0x55555555523a <main+17>     shr    $0x3,%eax
0x55555555523d <main+20>     je      0x55555555524d <main+36>
0x55555555523f <main+22>     mov    $0x0,%eax
0x555555555244 <main+27>     callq  0x5555555553c7 <risky_read_int>
0x555555555249 <main+32>     add    %ebx,%eax
0x55555555524b <main+34>     pop     %rbx
0x55555555524c <main+35>     retq
0x55555555524d <main+36>     mov    $0x0,%eax
0x555555555252 <main+41>     callq  0x5555555553c7 <risky_read_int>
0x555555555257 <main+46>     imul   %ebx,%eax
0x55555555525a <main+49>     jmp    0x55555555524b <main>
```
Conditional Jumps without `cmp` *(shift_jump.c)*

- Conditional jumps like `je` depend on flags set in the eflags register, which could be set by `cmp`, but also can be set by any arithmetic or logical instruction.
- You can still fluently read these by assuming a comparison to 0.
- Example of how to read these:
  - `shr $0x3, %eax; je 0x55555555524d <main+36>`
  - "^Read this as “jump to main+36 if (%eax >> 3) is equal to 0”"
Conditional Jumps with `test` on the same operand (`null_jump.c`)

```
0x555555555229 <main>       endbr64
0x55555555522d <main+4>     sub    $0x8,%rsp
0x555555555231 <main+8>     mov    $0x0,%eax
0x555555555236 <main+13>    callq  0x555555555390 <risky_read_string>
0x55555555523b <main+18>    test   %rax,%rax
0x55555555523e <main+21>    je      0x555555555248 <main+31>
0x555555555240 <main+23>    mov    %rax,%rdi
0x555555555243 <main+26>    callq  0x5555555550c0 <free@plt>
0x555555555248 <main+31>    mov    $0x0,%eax
0x55555555524d <main+36>    add    $0x8,%rsp
```
Conditional Jumps with `test` on the same operand (`null_jump.c`)

- The test instruction bitwise-ANDs (&) its arguments and sets flags based on it.
- We can use the same interpretation from before - “jump to X if (A & A) is equal to 0” or whatever.
- But we can simplify in this common case: the bitwise AND of something with itself is always itself (why?)
- Thus we’re simply jumping based on the value of an operand.
- Example of how to read these:
  - `test %rax,%rax; je 0x555555555248 <main+31>`
  - ^Read this as “jump to main+31 if %rax is equal to 0”
Practical Advice for Assembly Puzzles
nanobomb1
walkthrough