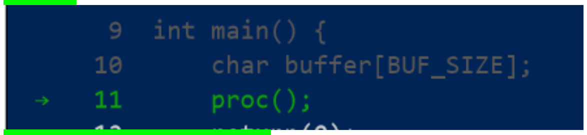
Analysis stack frame

Sunday, 29 March 2020 3:16 PM

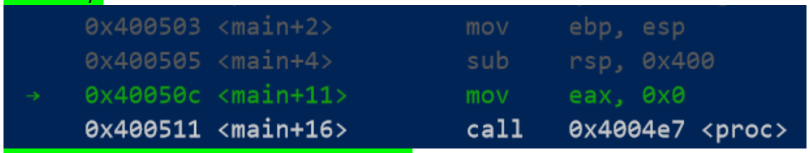
C code:



```
    int main() {
        char buffer[BUF_SIZE];
→       proc();
```

Create buffer of 1024 bytes
Call proc()

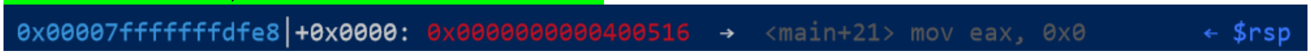
Assembly:



```
   0x400503 <main+2>          mov    ebp, esp
   0x400505 <main+4>          sub    rsp, 0x400
→  0x40050c <main+11>         mov    eax, 0x0
   0x400511 <main+16>         call   0x4004e7 <proc>
```

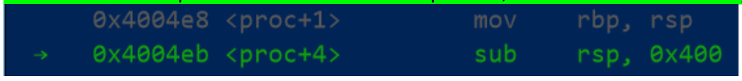
Create buffer of 1024 bytes, 0x400 = 1024
Call proc

When function is called, return address is saved into the stack



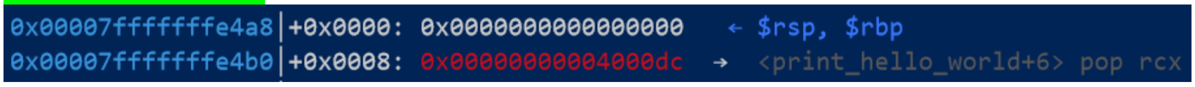
```
0x00007fffffffdfe8|+0x0000: 0x0000000000400516  →  <main+21> mov eax, 0x0        ← $rsp
```

The current stack pointer is saved into base pointer, and stack size of 1024bytes is created



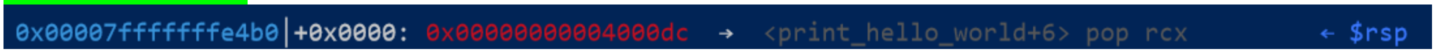
```
   0x4004e8 <proc+1>          mov    rbp, rsp
→  0x4004eb <proc+4>          sub    rsp, 0x400
```

Stack before leave is called



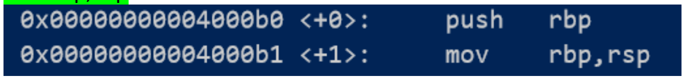
```
0x00007fffffffe4a8|+0x0000: 0x0000000000000000   ← $rsp, $rbp
0x00007fffffffe4b0|+0x0008: 0x00000000004000dc  →  <print_hello_world+6> pop rcx
```

Stack after leave is called



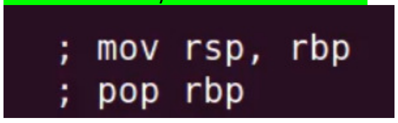
```
0x00007fffffffe4b0|+0x0000: 0x00000000004000dc  →  <print_hello_world+6> pop rcx        ← $rsp
```

In a way, it undoes
Push rbp
Mov rbp, rsp



```
0x00000000004000b0 <+0>:     push   rbp
0x00000000004000b1 <+1>:     mov    rbp,rsp
```

Leave is basically this two statements



```
  ; mov rsp, rbp
  ; pop rbp
```