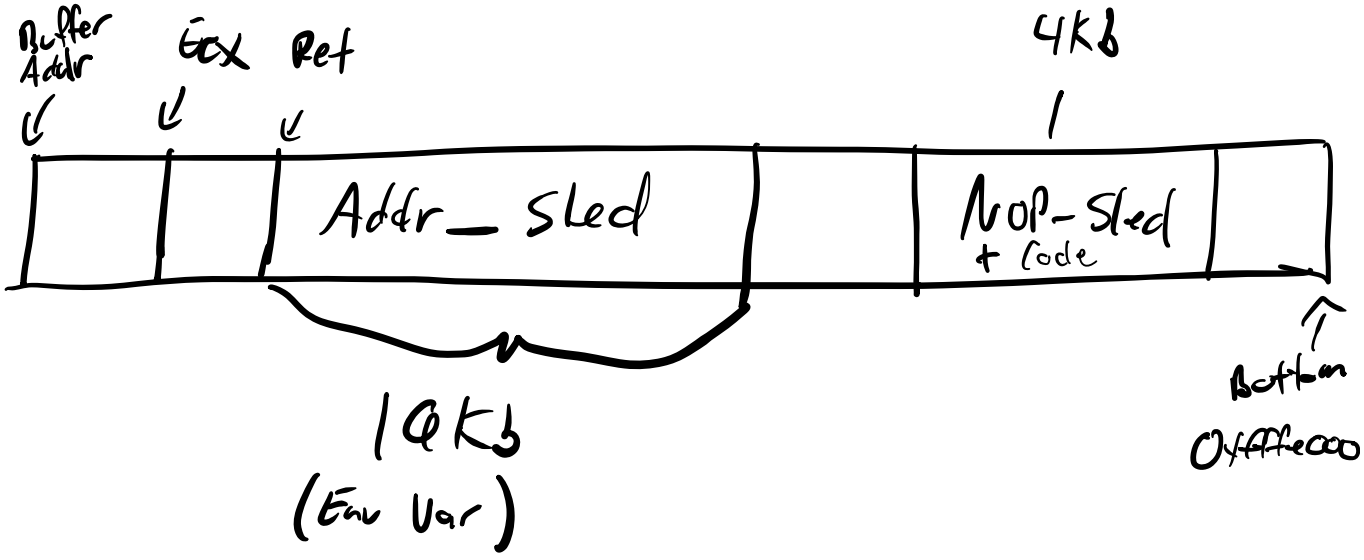
Stack 4-32 Notes

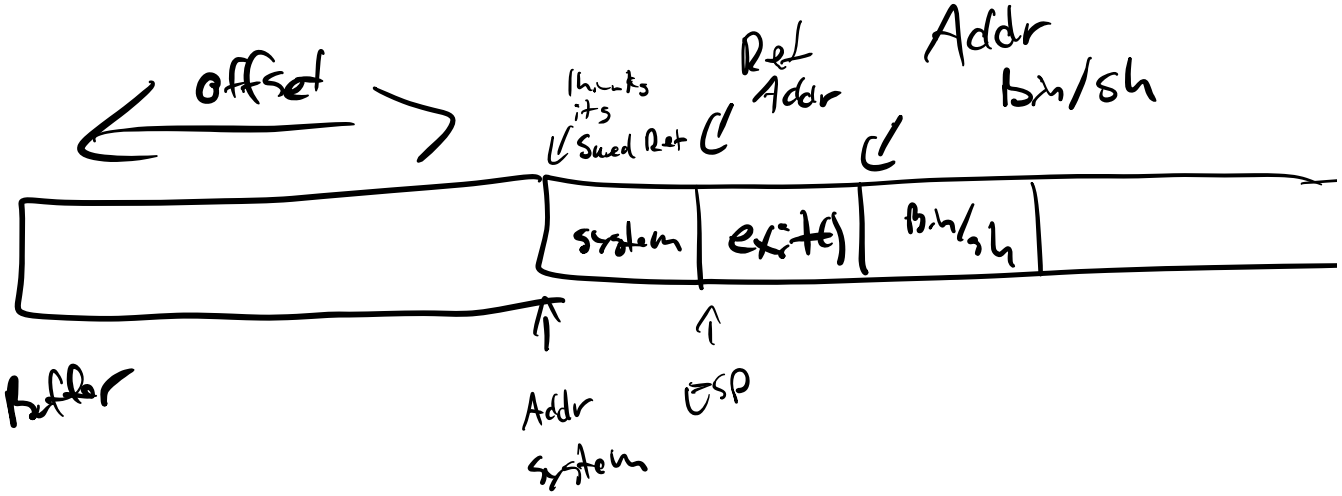
- Putting addr_sled in env variables doesn't work right because of the alignment
- Buffer is always aligned properly which is why we put the payload inside of it



- Env Vars allow us to grow the stack and make guessing easier

Stack 5 Continued

- Stack is not executable (NX Bit)
- Calls for code reuse
  - ↳ Ret2 libC attack



Compile 32 bit

```
gcc -m32 -o simple simple.c -g
```

* useful for seeing calling conventions in assembly *

To search in gef:

```
search-pattern <text> <Addr>-<Addr>
```

0. Corefile to get a corefile when

the program crashes

p.corefile.fault_addr

↳ returns the addr that caused the program to crash

offset = cyclic.find( )

Stack 6

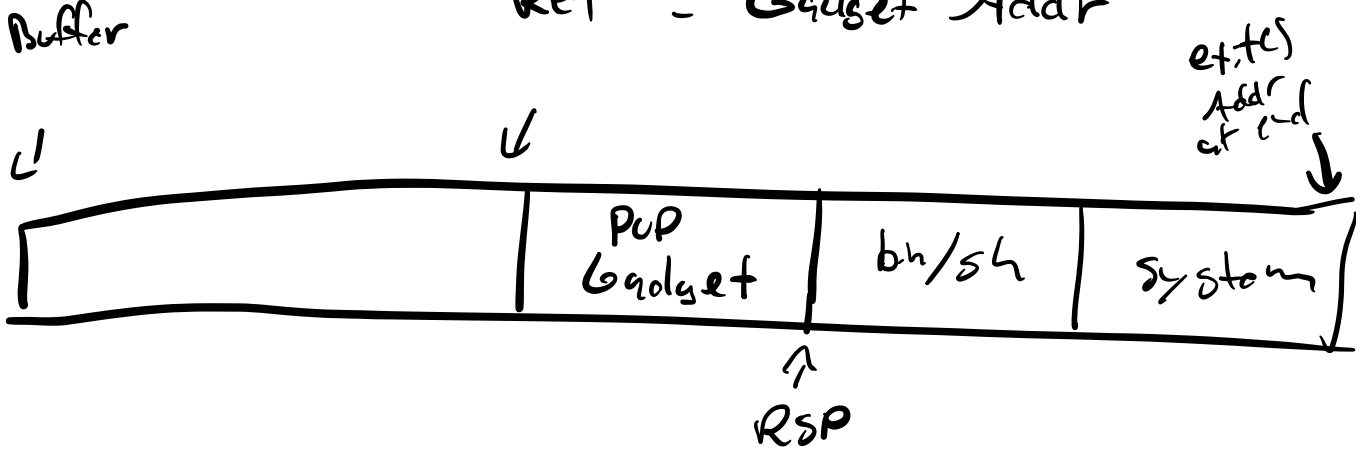
- 64 bit system calls use registers instead of the stack
- Need ROP gadgets to exploit

the binary

- Rdi register is used for system
- Need to find a gadget that can load a value into Rdi
- POP Rdi instruction
  ret
- Use ROP Gadget to find it

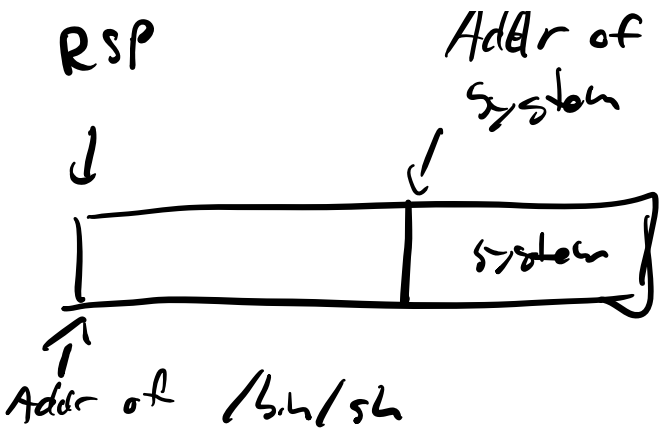
- addr_POP = ROP gadget Addr

Ret = Gadget Addr



System

Rdi : addr of /bin/sh



Pop Rdi

ret