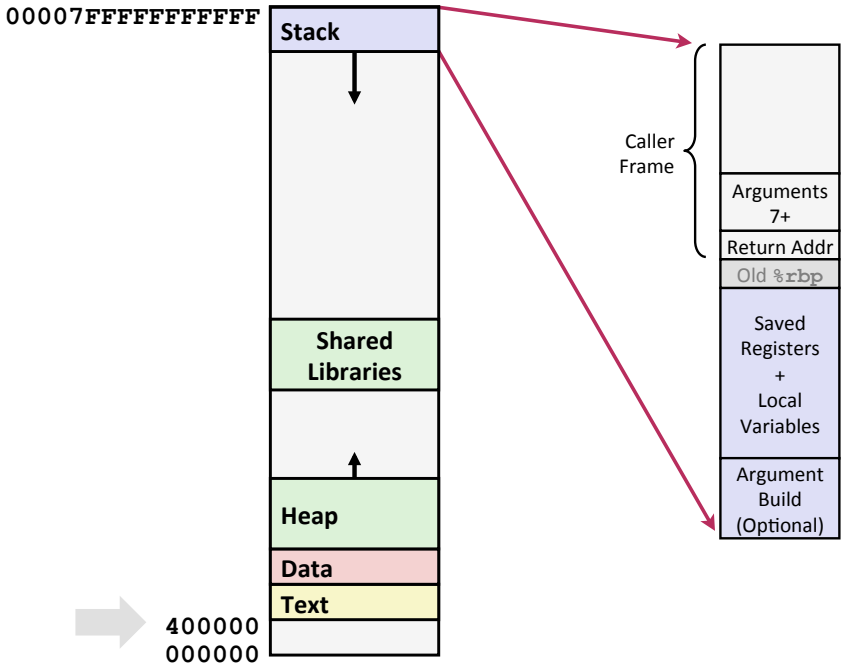
x86-64 Linux Memory Layout



String Library Code

```
/* Get string from stdin */
char* gets(char* dest) {
    int c = getchar();
    char* p = dest;
    while (c != EOF && c != '\n') {
        *p++ = c;
        c = getchar();
    }
    *p = '\0';
    return dest;
}
```

See also: `strcpy`, `strcat`, `scanf`, `fscanf`, `sscanf`, …

Vulnerable Buffer Code

```
/* Echo Line */
void echo() {
    char buf[4];  /* Way too small! */
    gets(buf);
    puts(buf);
}
```

```
void call_echo() {
    echo();
}
```

```
unix>./buftest
Type a string:0123456789012345678901230123456789012345678901230123
```

```
unix>./buftest
Type a string:012345678901234567890123
Segmentation Fault
```


Buffer Overflow Assembly

echo:

```
00000000004006cf <echo>:
 4006cf:  48 83 ec 18           sub    $0x18,%rsp
 4006d3:  48 89 e7              mov    %rsp,%rdi
 4006d6:  e8 a5 ff ff ff        callq  400680 <gets>
 4006db:  48 89 e7              mov    %rsp,%rdi
 4006de:  e8 3d fe ff ff        callq  400520 <puts@plt>
 4006e3:  48 83 c4 18           add    $0x18,%rsp
 4006e7:  c3                    retq
```

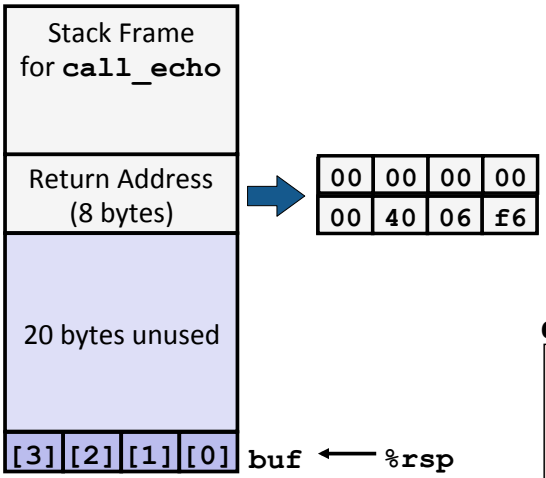
call_echo:

```
 4006e8:   48 83 ec 08           sub    $0x8,%rsp
  4006ec:  b8 00 00 00 00        mov    $0x0,%eax
  4006f1:  e8 d9 ff ff ff        callq  4006cf <echo>
  4006f6:  48 83 c4 08           add    $0x8,%rsp
  4006fa:  c3                    retq
```

Buffer Overflow Stack

```
/* Echo Line */
void echo() {
    char buf[4];
    gets(buf);
    puts(buf);
}
```

Before call to gets



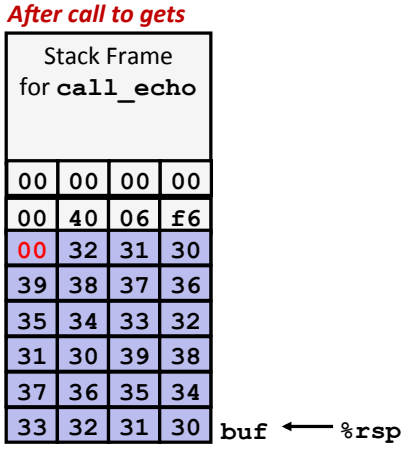

```
echo:
  subq  $24, %rsp
  movq  %rsp, %rdi
  call  gets
  . . .
```

call_echo:

```
  . . .
4006f1:  callq  4006cf <echo>
4006f6:  add    $0x8,%rsp
  . . .
```


Buffer Overflow Examples

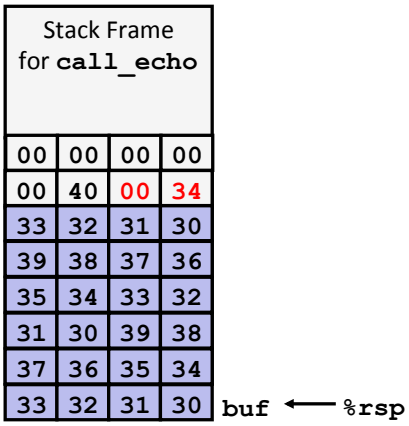
After call to gets




```
unix>./buftest
Type a string:0123456789012345678901
0123456789012345678901
```

Overflowed, but did not corrupt state

After call to gets




```
unix>./buftest
Type a string:012345678901234567890123
Segmentation Fault
```

Overflowed and corrupted return pointer

Code Injection Attacks

```
void P(){
  Q();
  ...
}
```

return address A

```
int Q() {
  char buf[64];
  gets(buf);
  ...
  return ...;
}
```