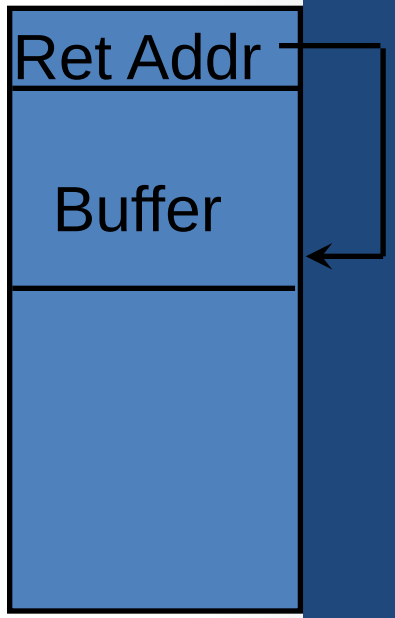
ELEC 377 – Operating Systems

Lab 5 Tutorial

Program Threats

- Buffer Overflow
  - ◊ Most common attack
  - ◊ exploit bug as security hole
  1. Write binary code into buffer, ending with a value that overwrites the return address and points into the buffer
  2. Subroutine returns into the stack instead of to calling program



Pentium Stack Layout

fd = open(“theFile”, O_RDONLY, 0744);

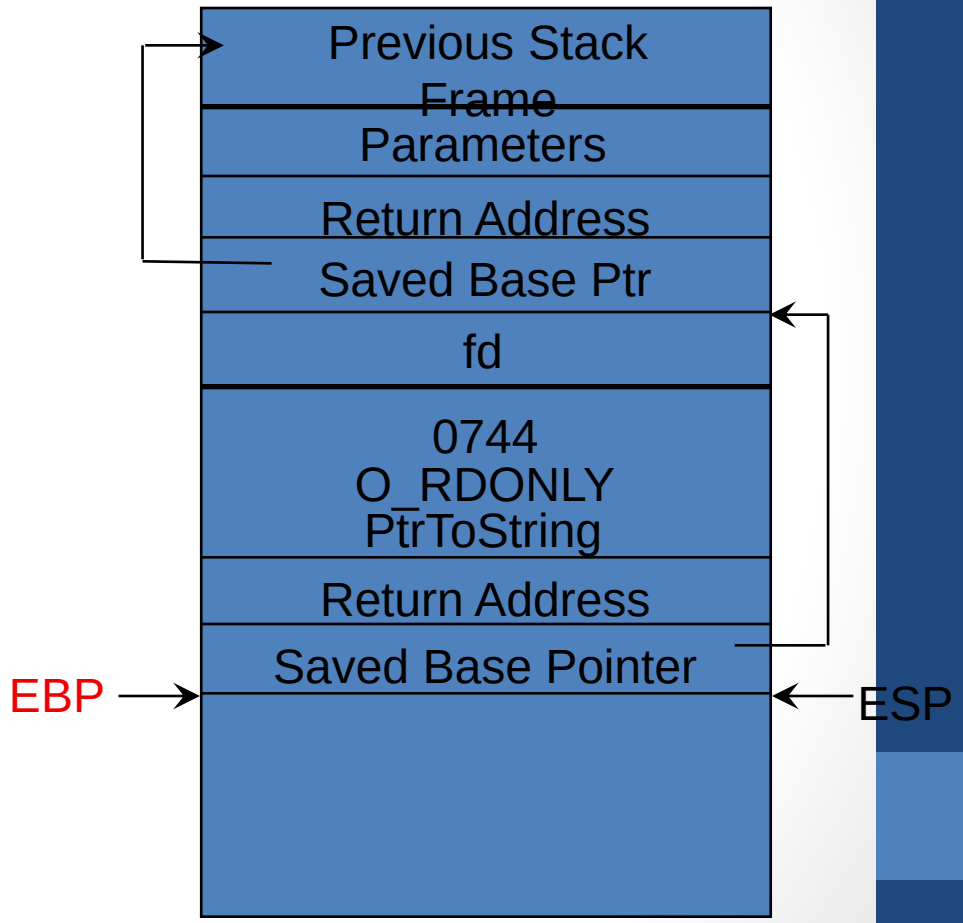
```
push 0744
push O_RDONLY
pushd PtrToString

call open

mov [ebp-fd],eax
add esp,12
```

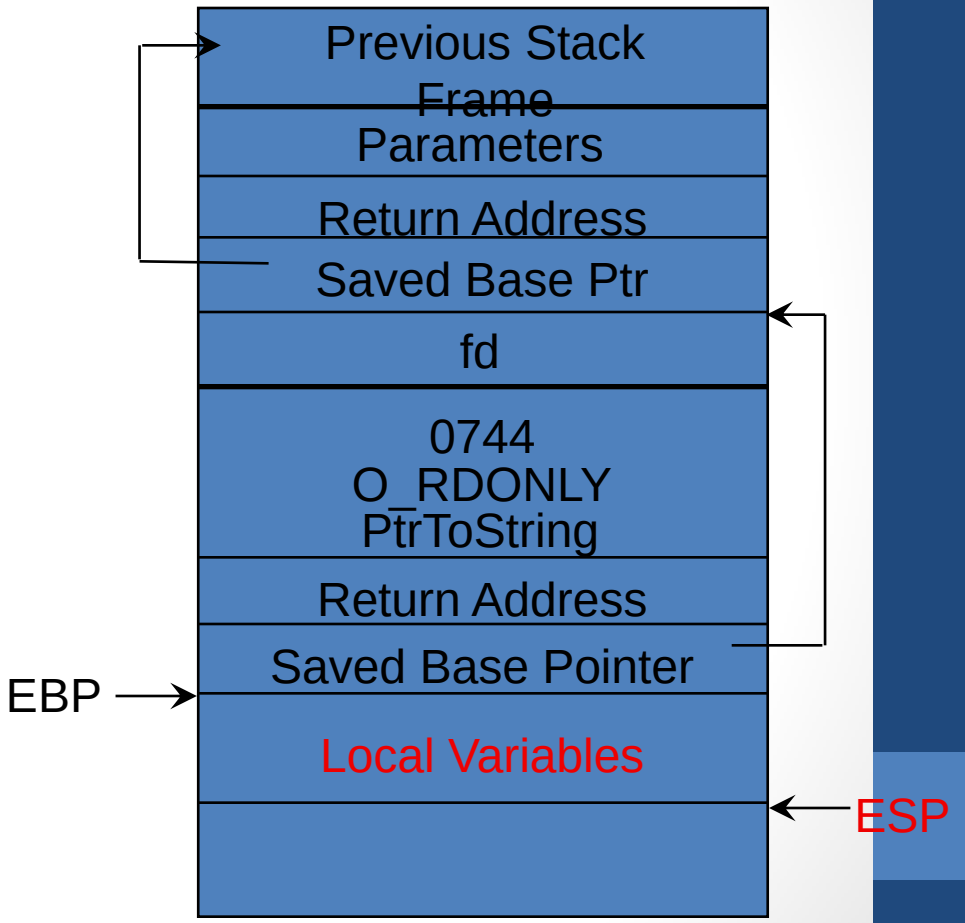
Pentium Stack Layout

push ebp
mov ebp,esp
add esp,NumLocals



leave
ret

Pentium Stack Layout



Stack Overflow Attack

```
char * GetLine(){
   char buffer[130];
   gets(buffer);
   checkChars(buffer); // only A-Z0-9
}
```

Stack Overflow Attack

```
char * GetLine(){
   char buffer[130];
   gets(buffer);
   checkChars(buffer); // only A-Z0-9
}
```

Stack Overflow Attack

```
char * GetLine(){
   char buffer[130];
   gets(buffer);
   checkChars(buffer); // only A-Z0-9
}
```

Stack Overflow Attack

```
char * GetLine(){
   char buffer[130];
   gets(buffer);
   checkChars(buffer); // only A-Z0-9
}
```

Stack Overflow Attack

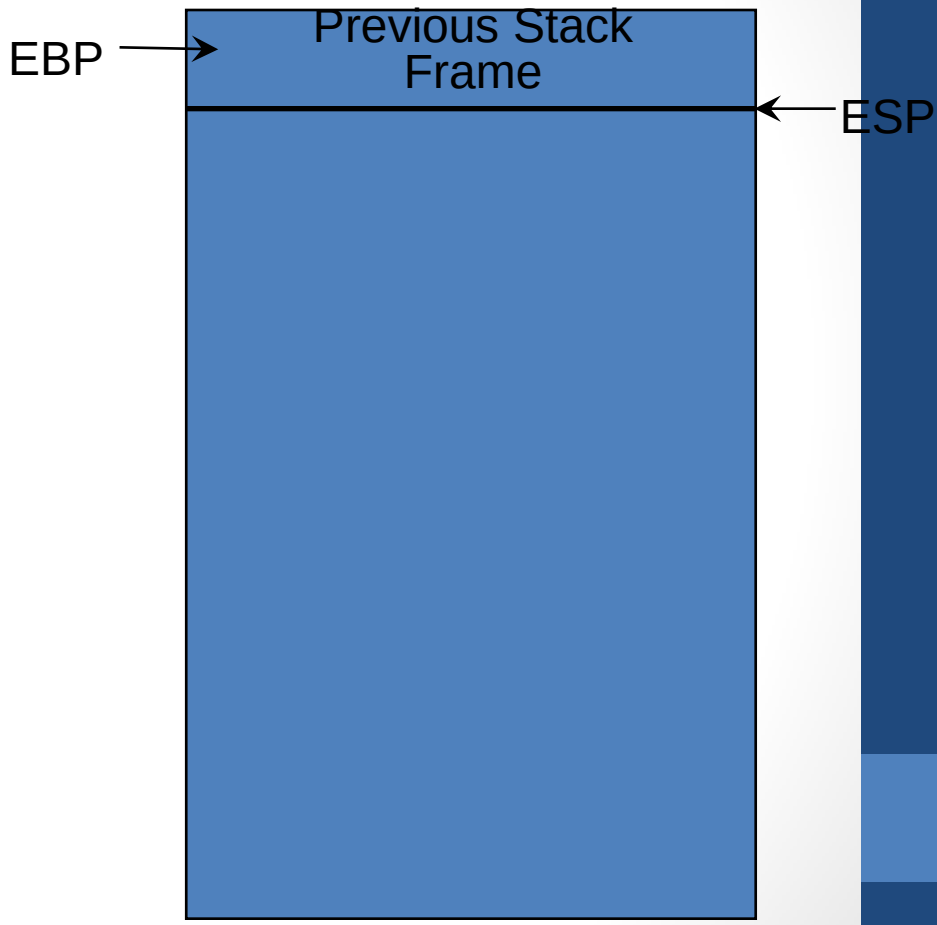
```
char * GetLine(){
   char buffer[130];
   gets(buffer);
   checkChars(buffer); // only A-Z0-9
}
```

Stack Overflow Attack

```
char * GetLine(){
   char buffer[130];
   gets(buffer);
   checkChars(buffer); // only A-Z0-9
}
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
      leave
   ret
```

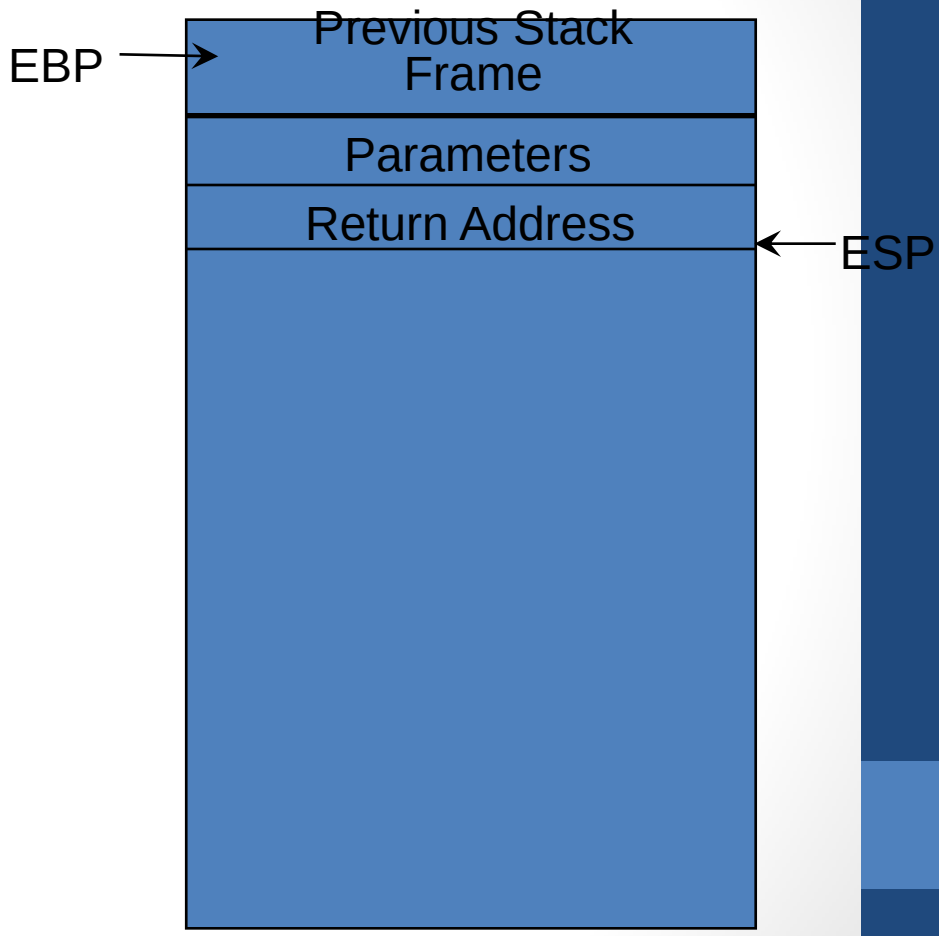
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



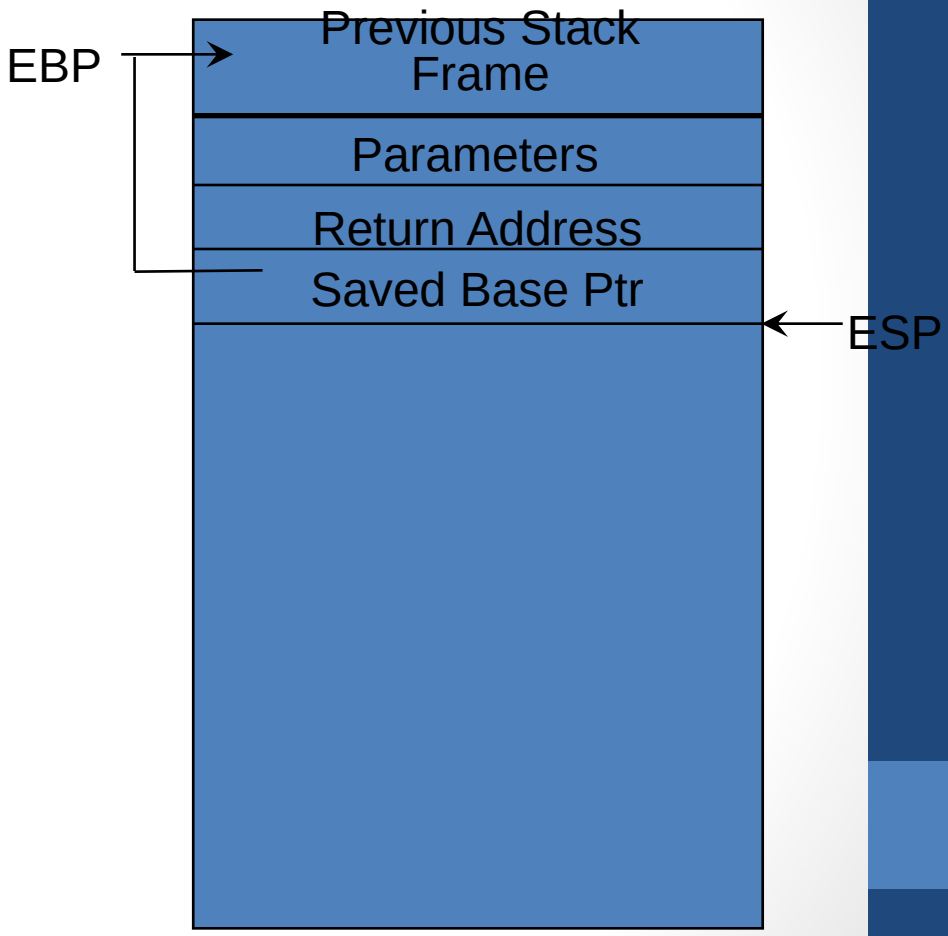
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



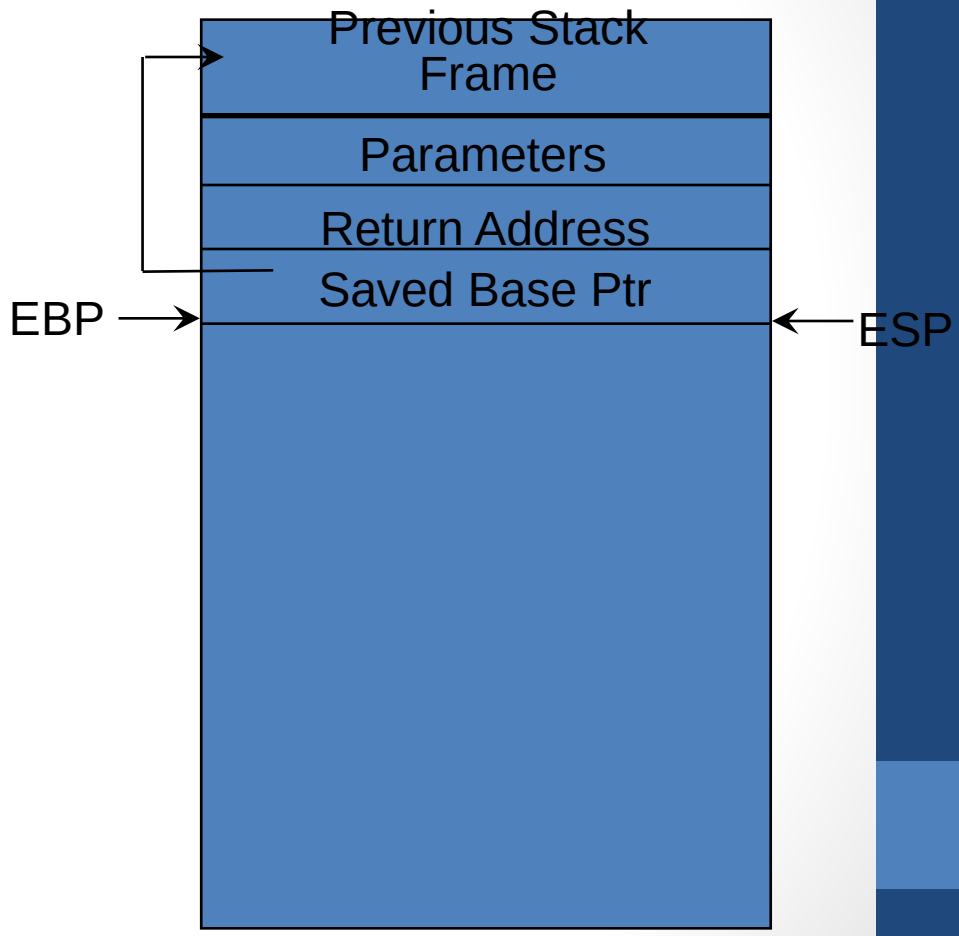
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



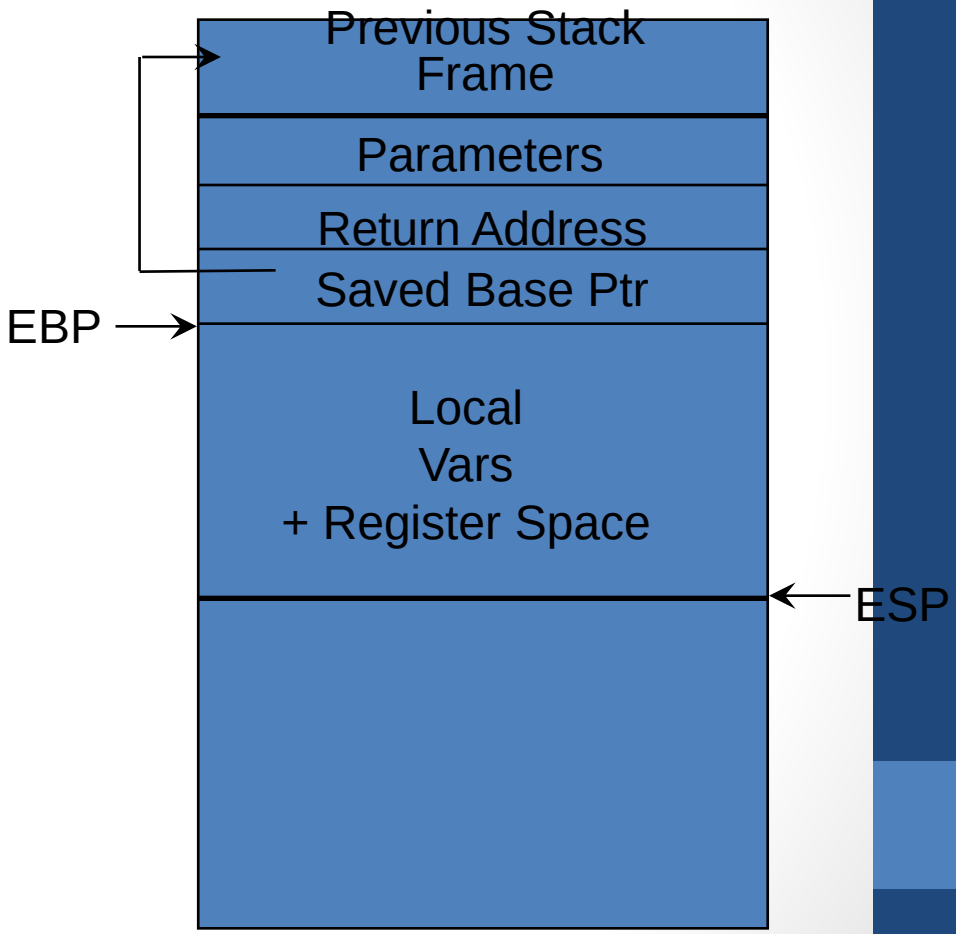
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



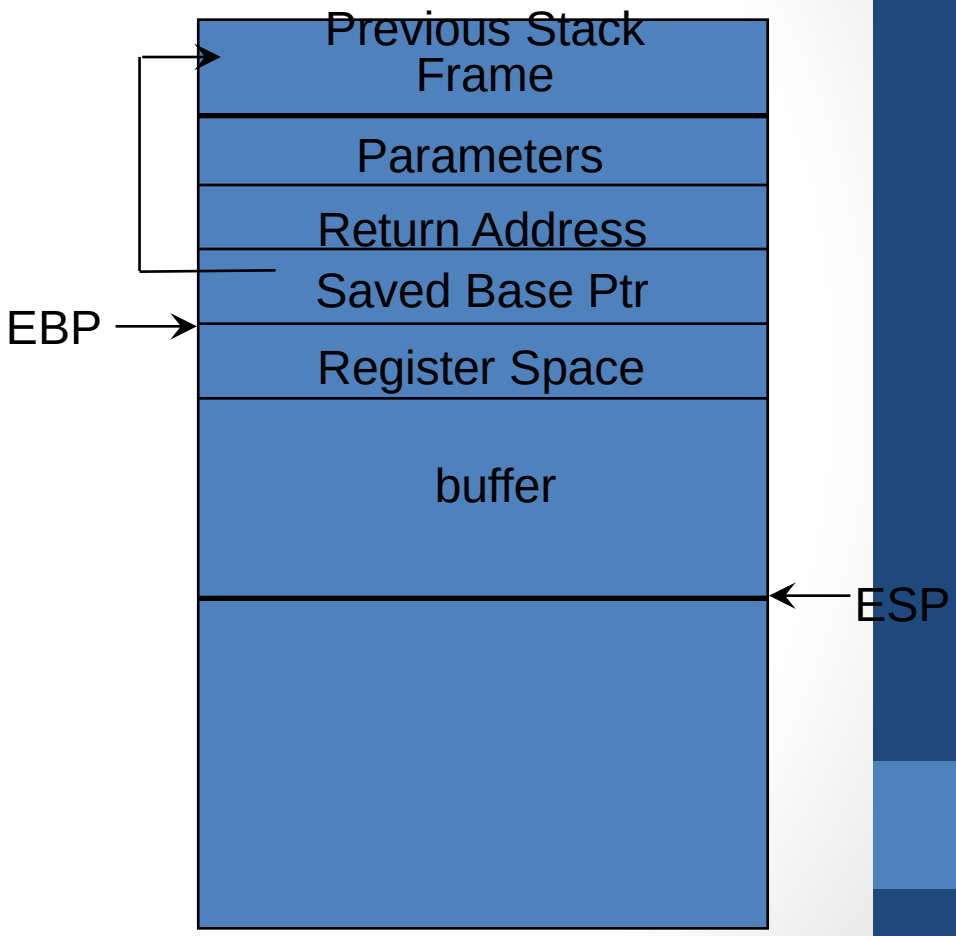
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



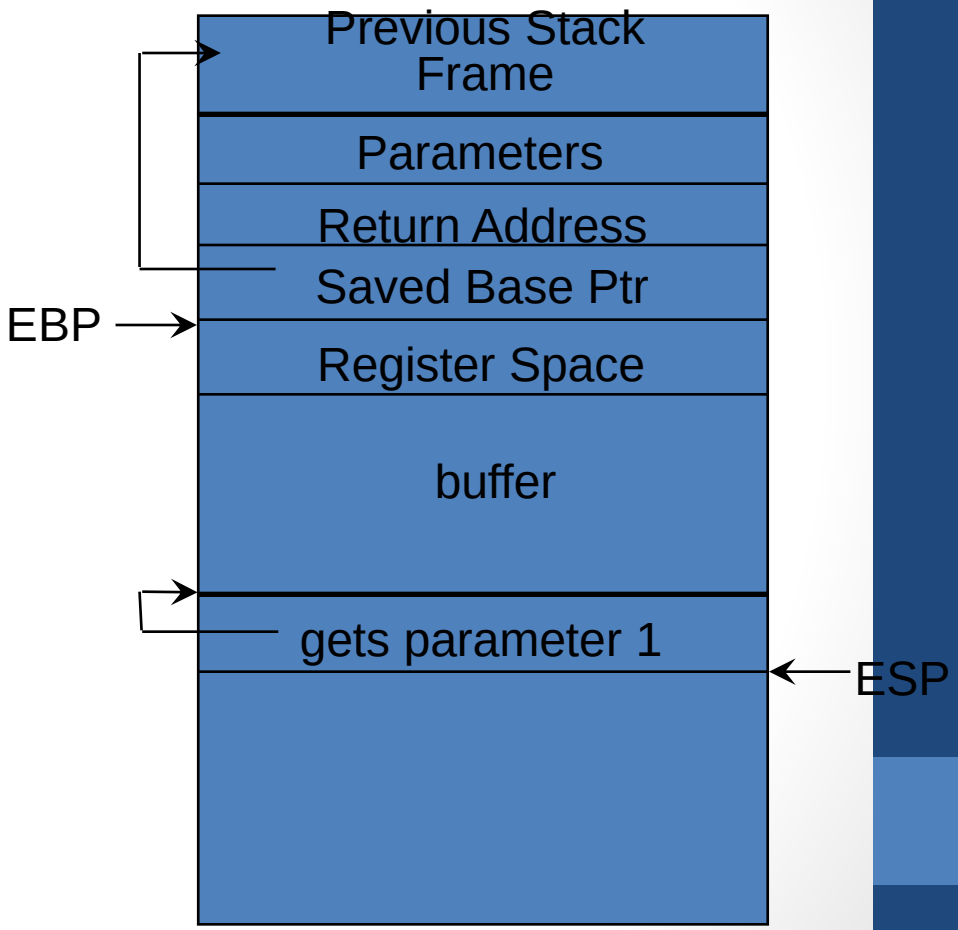
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



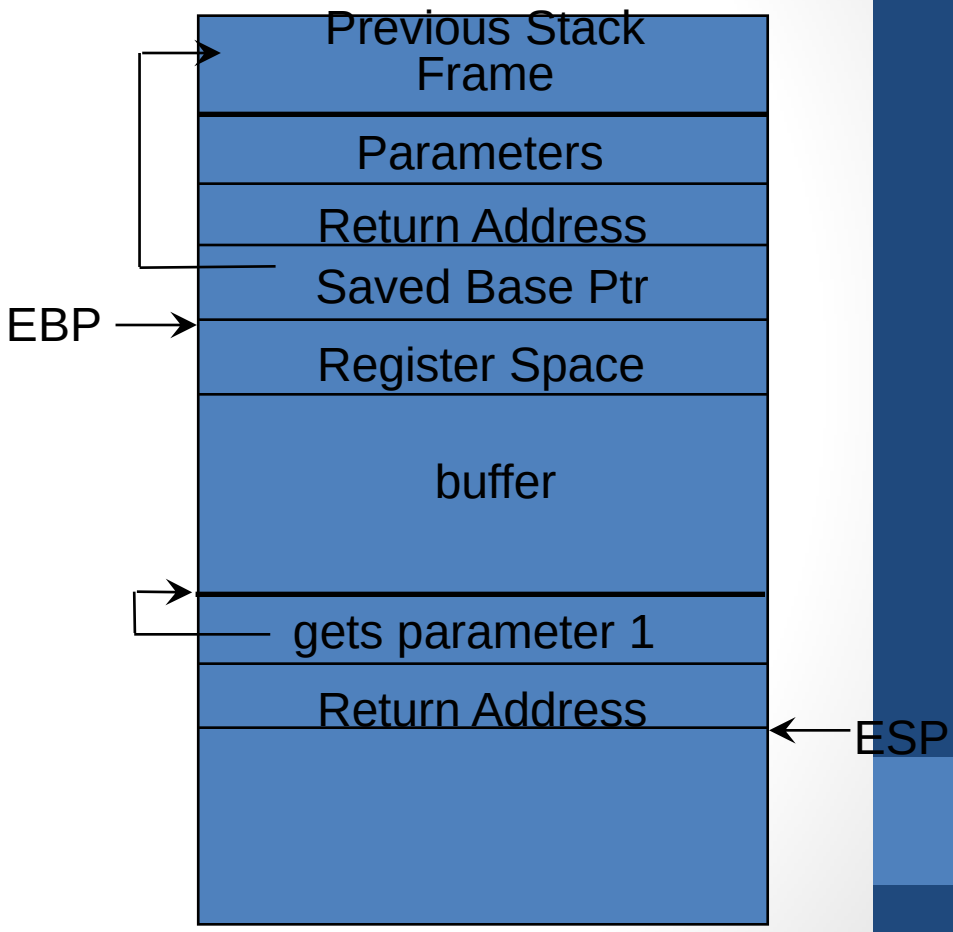
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



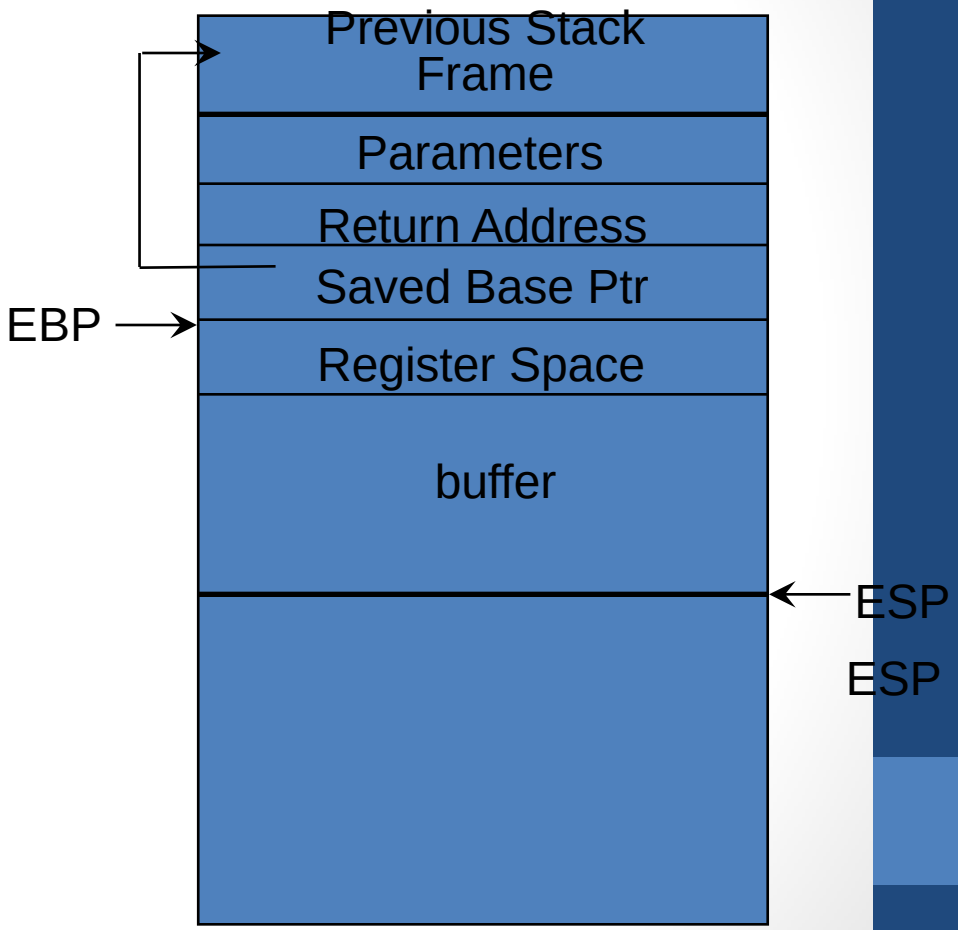
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



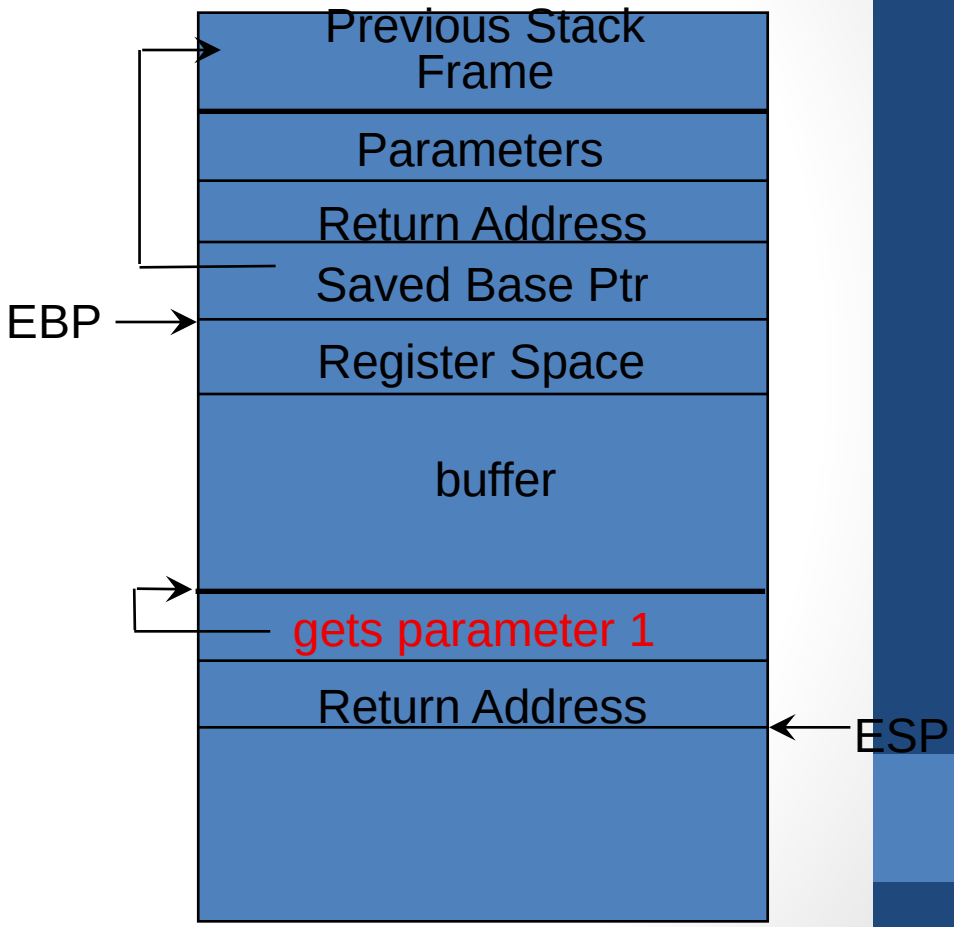
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



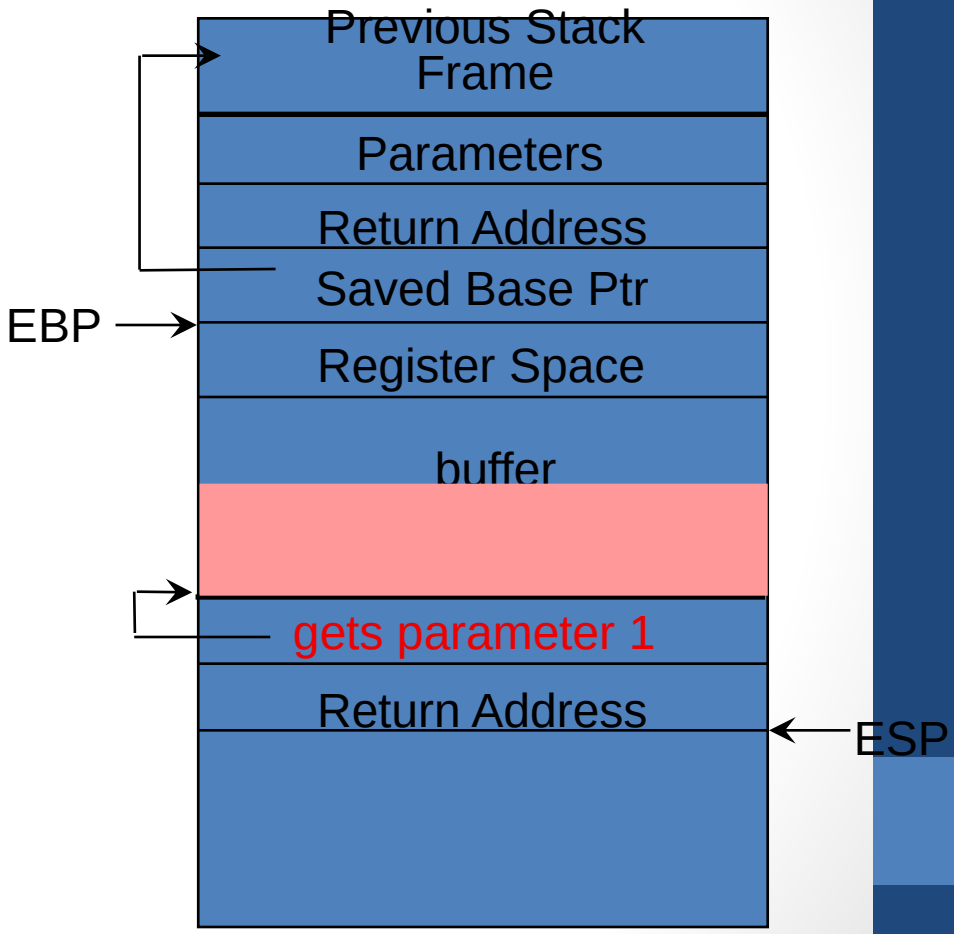
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



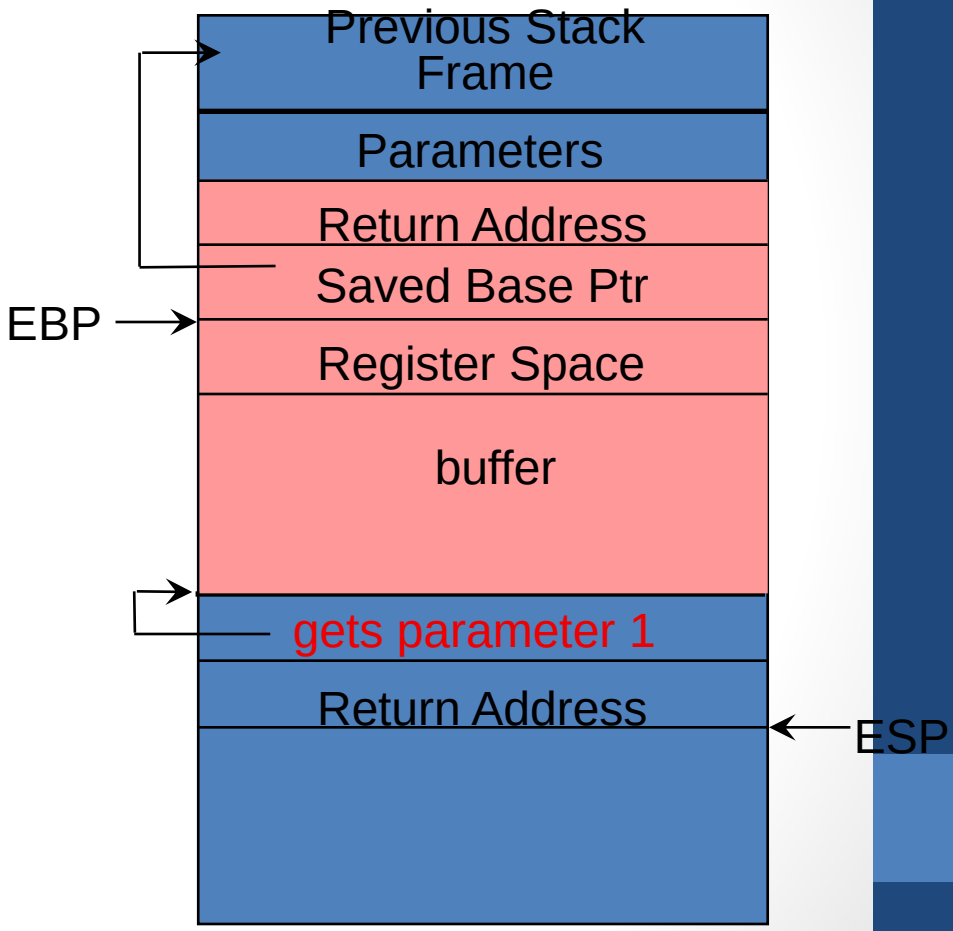
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



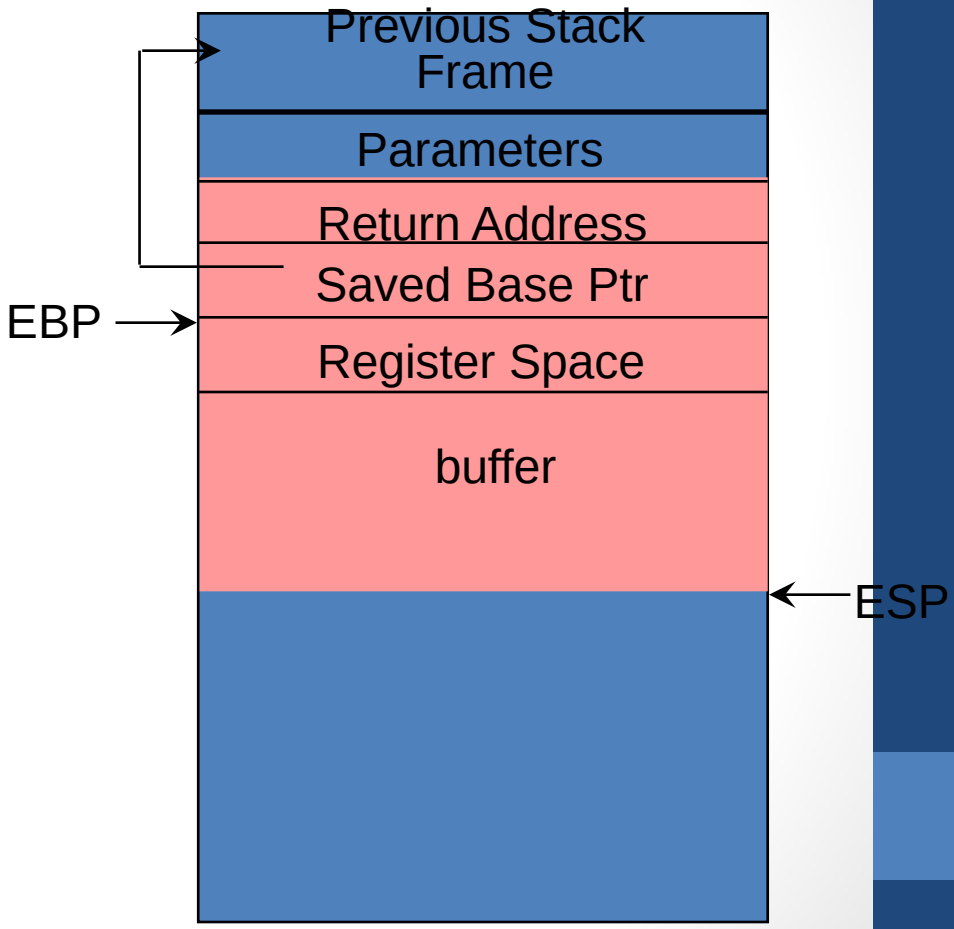
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



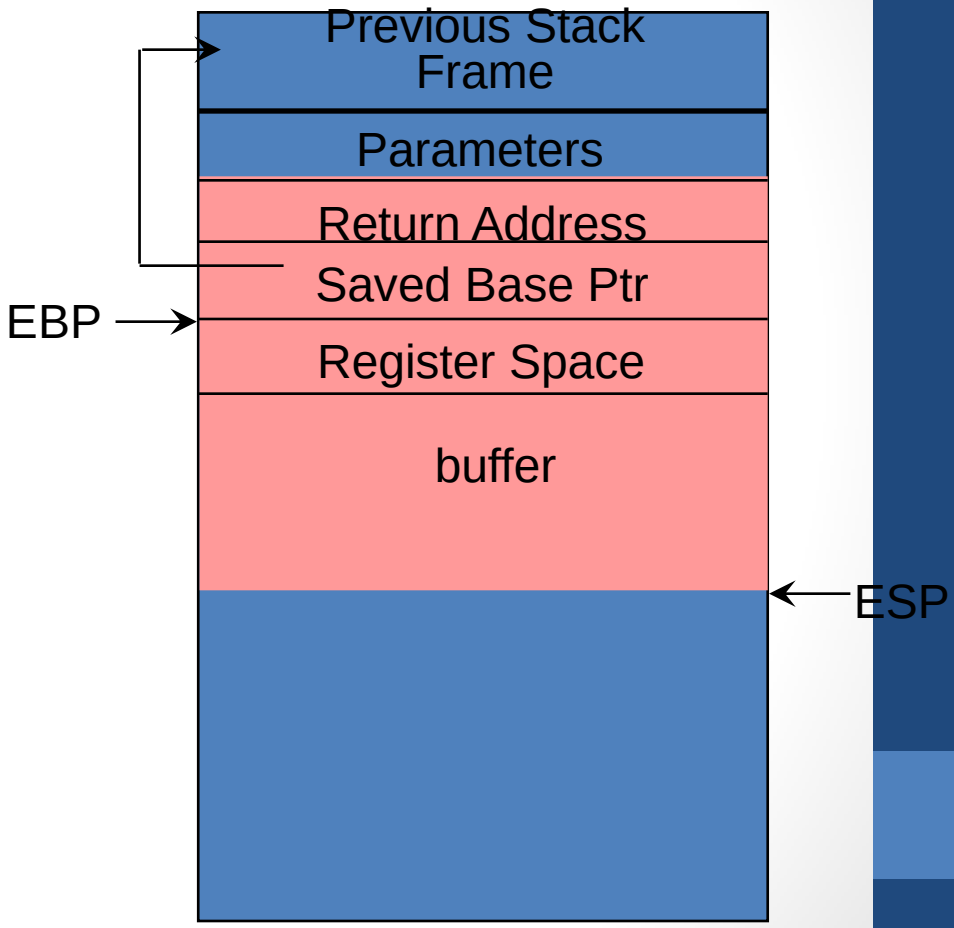
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



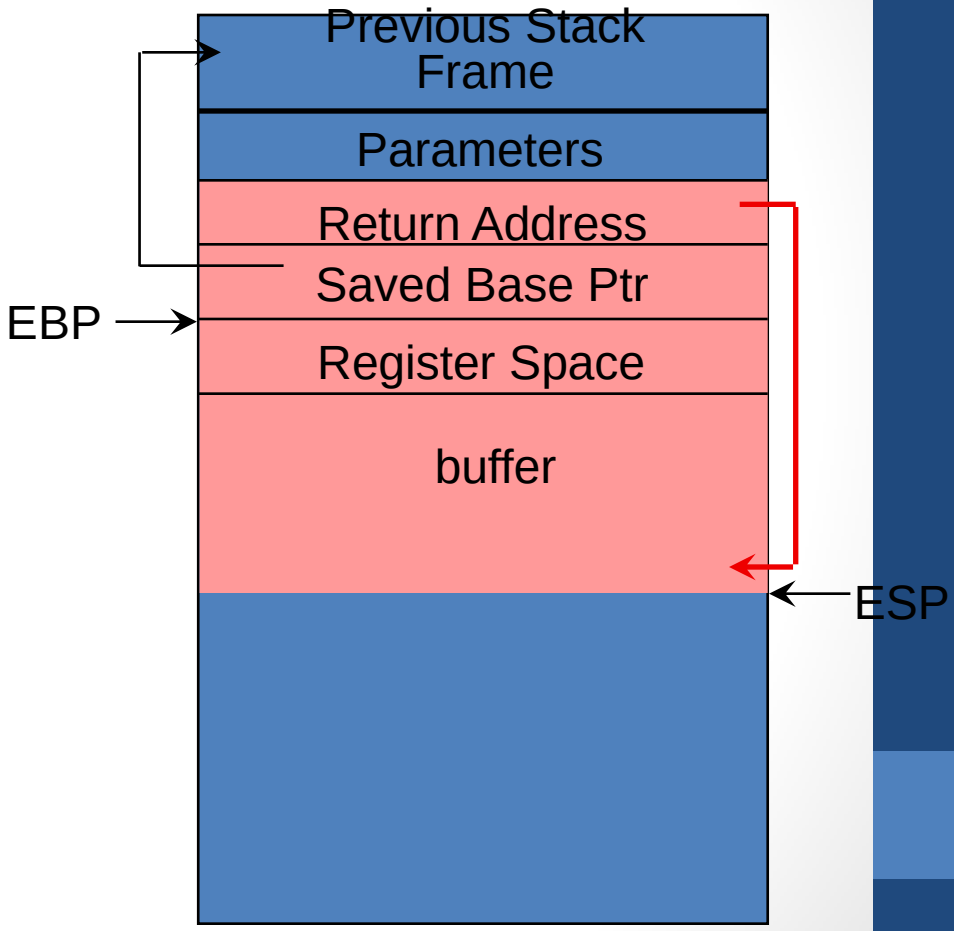
Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



Stack Overflow Attack

```
getLine:
   push  ebp
   mov   ebp,esp
   sub   esp,152
   lea   eax,-152(ebp)
   pushl eax
   call  gets
   add   esp,4
   lea   eax,-152(ebp)
   pushl eax
   call  checkChars
   add   esp,4
     leave
   ret
```



What does the code look like?

- System Call

```
result = execve(
                    const char * path,
                            char *const argv[],
                            char *const envp[]
                            );
```

What does the code look like?

- System Call

```
result = execve(
            const char * path, cmd to execute
                    char *const argv[],args for cmd
                    char *const envp[]
        );
```

What does the code look like?

- System Call

```
result = execve(
            const char * path, cmd to execute
            char *const argv[],args for cmd
            char *const envp[] environment
          );
```

What does the code look like?

- System Call

```
result = execve(
                        const char * path,
                        char *const argv[],
                        char *const envp[]
                      );
```

- Calling Conventions:
  - ◊ uses registers instead of the stack
  - ◊ eax = system call (execve => 0x0b)
  - ◊ ebx = 1st parameter (path)
  - ◊ ecx = 2nd parameter (argv)
  - ◊ edx = 3rd parameter (envp)
- both argv and envp are arrays of pointers, terminated with a null pointer.

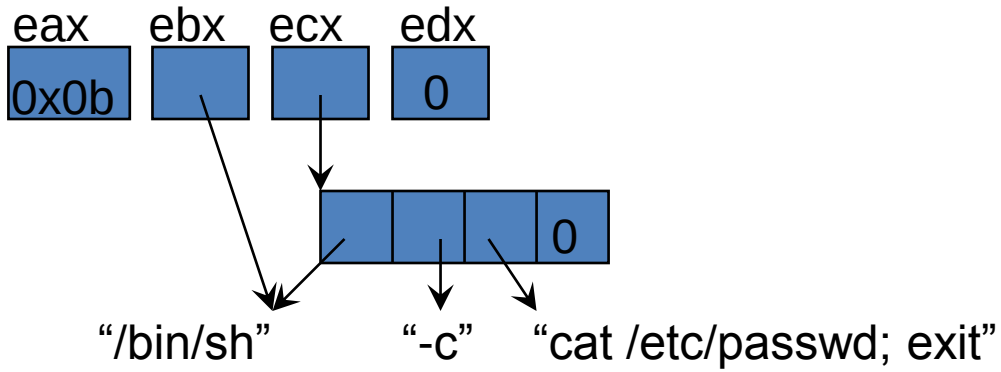
What does the code look like?

- System Call

```
result = execve(
                        const char * path,
                        char *const argv[],
                        char *const envp[]
                      );
```

- argv contains:
  - ◊ 0 => path to command *(same as first arg to execve)*
  - ◊ 1...n-2 => flags
  - ◊ n-1 => NULL
- Our case:
  - ◊ 0 => /bin/sh
  - ◊ 1 => -c
  - ◊ 2 => cat /etc/passwd; exit
  - ◊ 3 => NULL

Data Structure for Exploit



Final Points

- Web site gives the structure of the shellcode
  - ◊ still need modifications
  - ◊ deal with null and 0x0A (newline) bytes

- Start by compromising self code, then move on to server.
- Testing - show the file recovered is the password file