

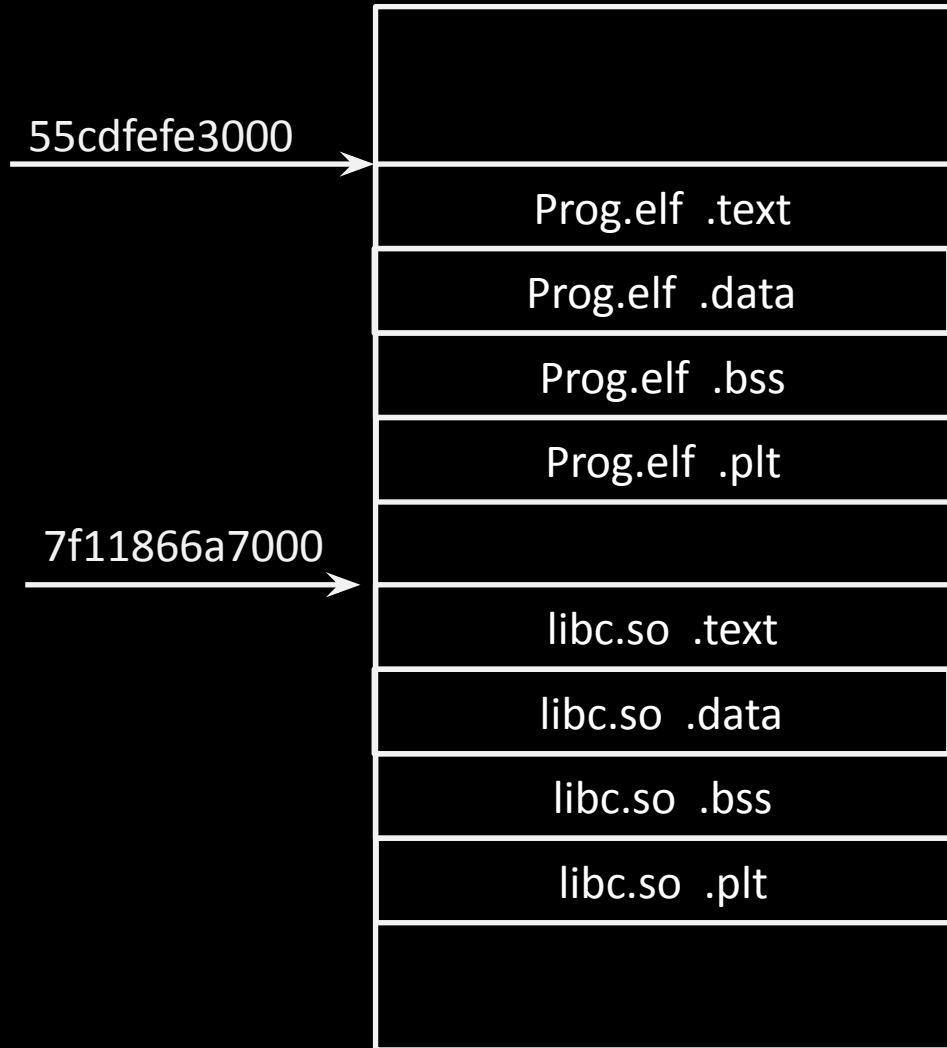
A woman with dark hair tied back, wearing a grey and white raglan shirt, is seen from behind in a library. She is looking at a shelf of books. The shelves are filled with books of various colors. The lighting is warm and soft, creating a cozy atmosphere. The text 'Атака возврата в библиотеку' and 'return to libc attack' is overlaid on the image in white.

Атака возврата в библиотеку
return to libc attack

What is libc ?



What is libc ?



What is libc ?

```
// You program
```

```
main
```

```
    call printf
```

```
    ret
```

```
plt:
```

```
    jmp printf
```

```
// libc.so
```

```
printf
```

```
    ...
```

```
    ret
```

```
Puts
```

```
    ...
```

```
    ret
```

```
System
```

```
    ...
```

```
    ret
```

ret2libc

1. We know version of libc.so
2. We know address of libc.so
3. We know any function address at libc.so

ret2libc

```
int main (){  
    char buf [16];  
    gets(buf);  
}
```



ret2libc

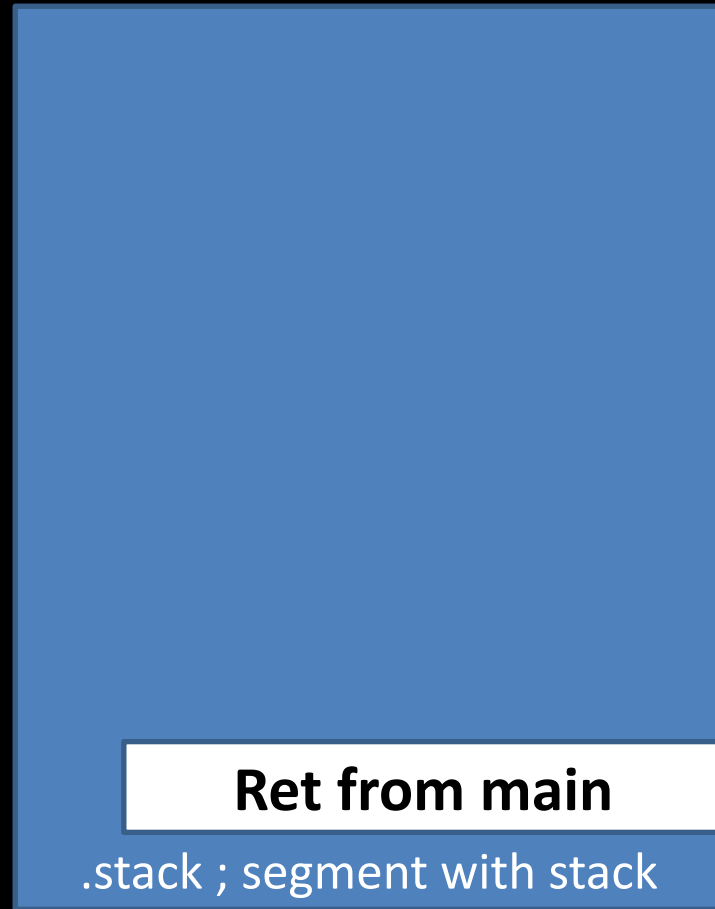
Main:

1. push ebp □eip
2. mov ebp, esp
3. sub esp, 16
4. lea edx, buf
5. push edx
6. call gets
7. add esp, 4
8. leave
9. Ret

.....

....

10. system :
11. ...
12. ret
13. Printf:
14. ...
15. ret



ret2libc

Main:

1. push ebp
2. mov ebp, esp □ eip
3. sub esp, 16
4. lea edx, buf
5. push edx
6. call gets
7. add esp, 4
8. leave
9. Ret

.....

....

10. system :
11. ...
12. ret
13. Printf:
14. ...
15. ret



ret2libc

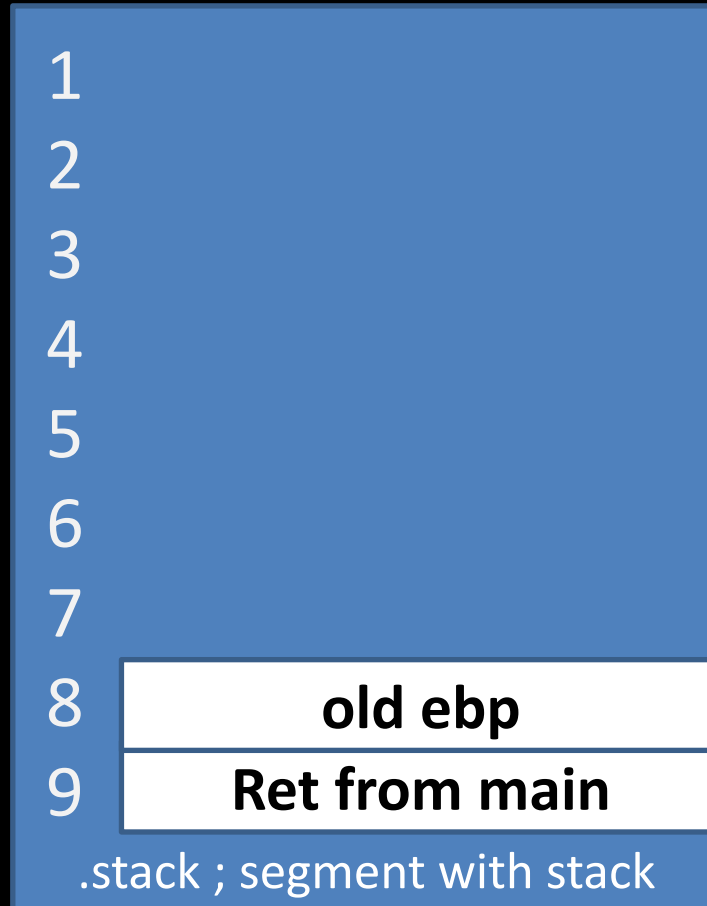
Main:

1. push ebp
2. mov ebp, esp
3. sub esp, 16 eip
4. lea edx, buf
5. push edx
6. call gets
7. add esp, 4
8. leave
9. Ret

.....

....

10. system :
11. ...
12. ret
13. Printf:
14. ...
15. ret



ebp=8

ret2libc

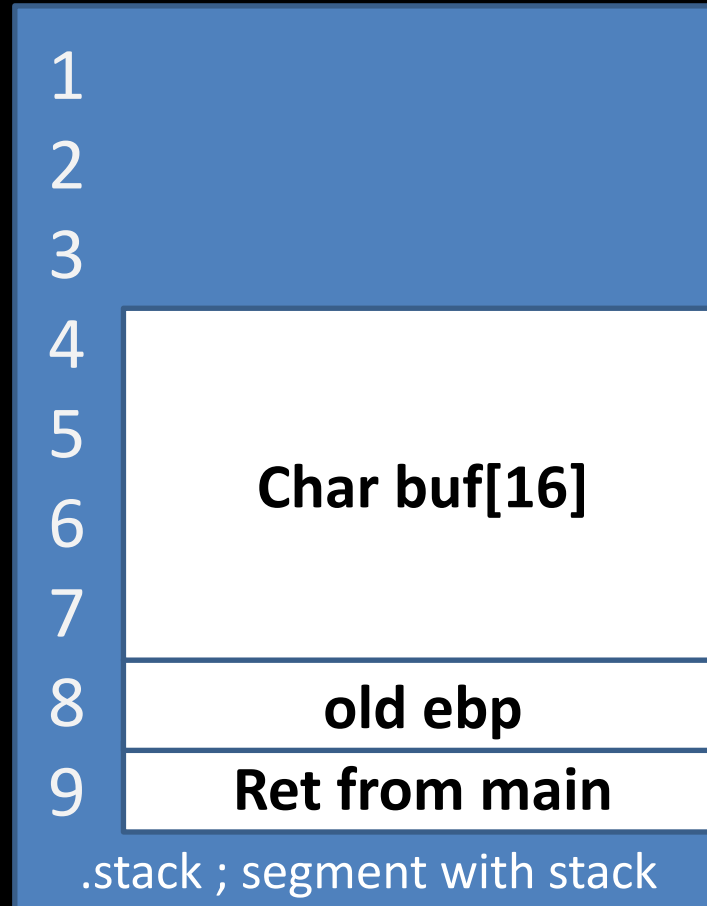
Main:

1. push ebp
2. mov ebp, esp
3. sub esp, 16
4. lea edx, buf □ eip
5. push edx
6. call gets
7. add esp, 4
8. leave
9. Ret

.....

....

10. system :
11. ...
12. ret
13. Printf:
14. ...
15. ret



ebp=8

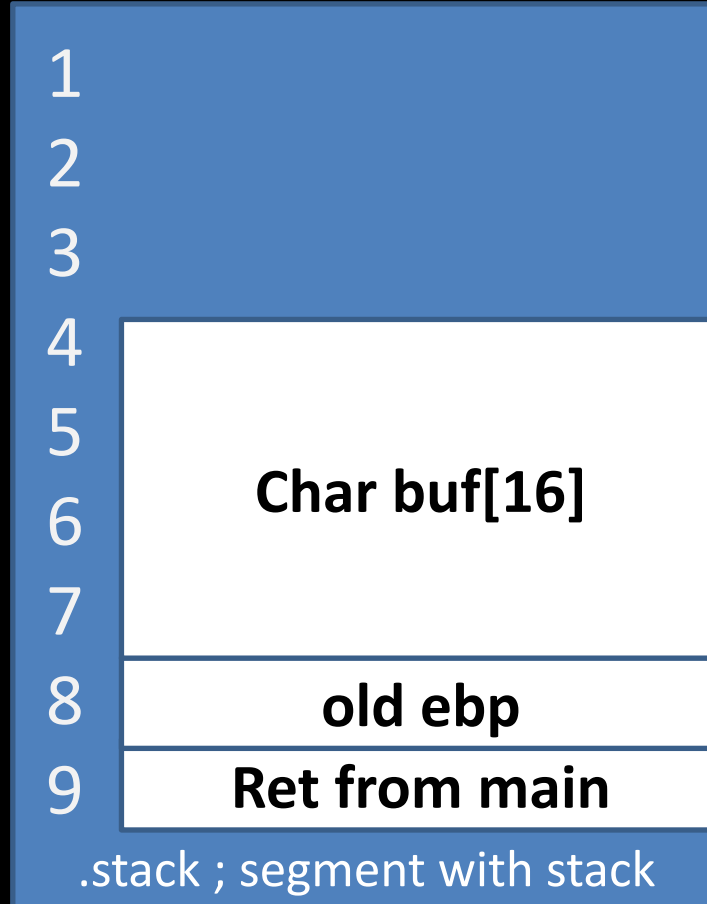
ret2libc

Main:

1. push ebp
2. mov ebp, esp
3. sub esp, 16
4. lea edx, buf
5. push edx □ eip
6. call gets
7. add esp, 4
8. leave
9. Ret

.....

-
10. system :
 11. ...
 12. ret
 13. Printf:
 14. ...
 15. ret



| |
|-------|
| ebp=8 |
| edx=4 |

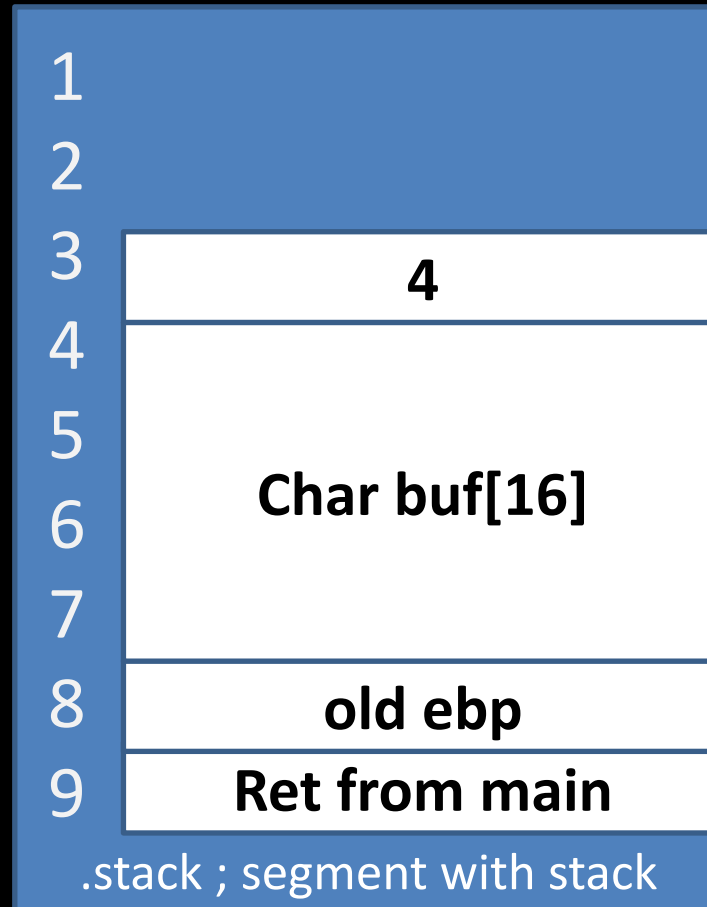
ret2libc

Main:

1. push ebp
2. mov ebp, esp
3. sub esp, 16
4. lea edx, buf
5. push edx
6. call gets □ eip
7. add esp, 4
8. leave
9. Ret

.....

-
10. system :
 11. ...
 12. ret
 13. Printf:
 14. ...
 15. ret



| |
|-------|
| ebp=8 |
| edx=4 |

ret2libc

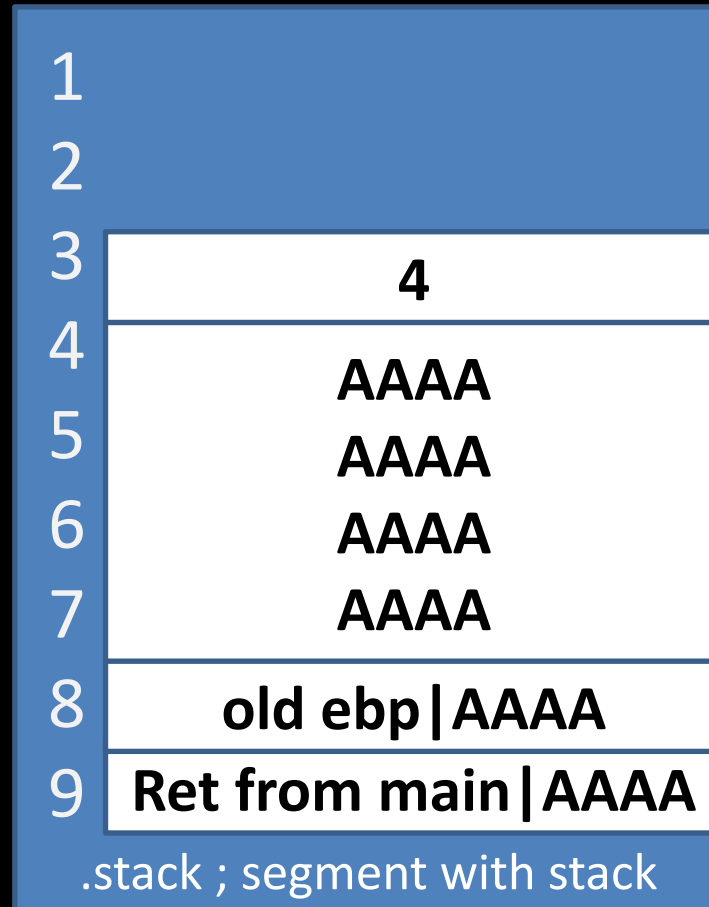
Main:

1. push ebp
2. mov ebp, esp
3. sub esp, 16
4. lea edx, buf
5. push edx
6. call gets
7. add esp, 4 □ eip
8. leave
9. Ret

.....

....

10. system :
11. ...
12. ret
13. Printf:
14. ...
15. ret



| |
|-------|
| ebp=8 |
| edx=4 |

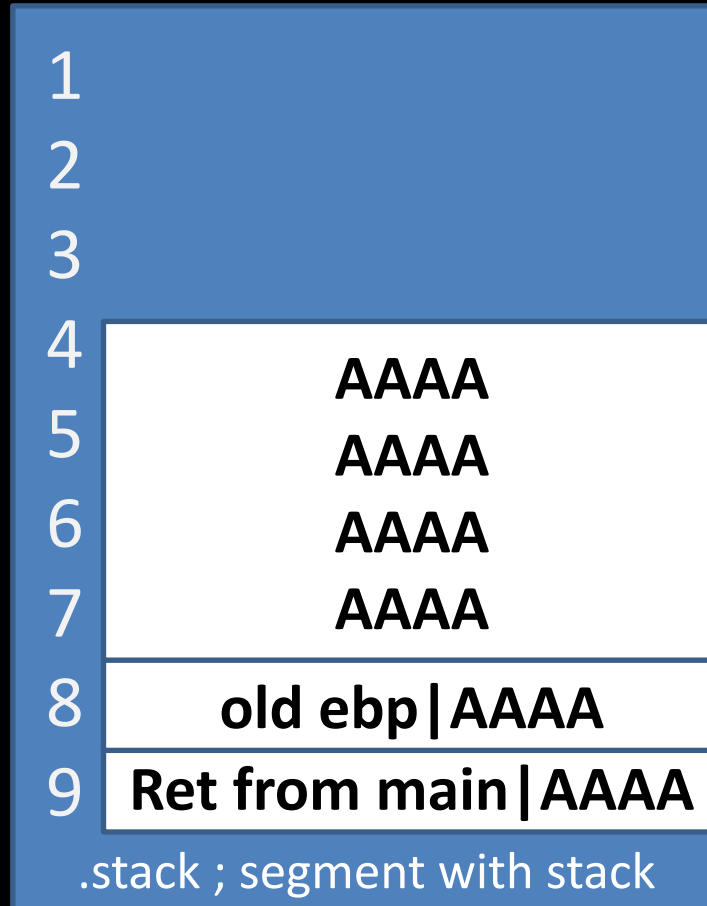
ret2libc

Main:

1. push ebp
2. mov ebp, esp
3. sub esp, 16
4. lea edx, buf
5. push edx
6. call gets
7. add esp, 4
8. leave □ eip
9. Ret

.....

-
10. system :
 11. ...
 12. ret
 13. Printf:
 14. ...
 15. ret



| |
|-------|
| ebp=8 |
| edx=4 |

ret2libc

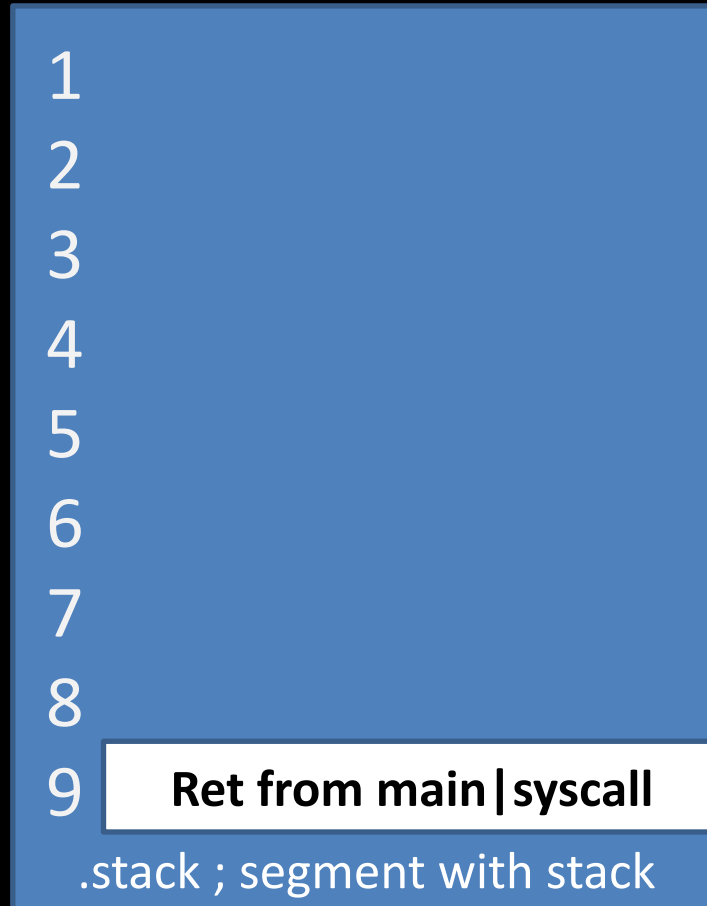
Main:

1. push ebp
2. mov ebp, esp
3. sub esp, 16
4. lea edx, buf
5. push edx
6. call gets
7. add esp, 4
8. leave
9. Ret □ eip

.....

....

10. system :
11. ...
12. ret
13. Printf:
14. ...
15. ret



| |
|-----------------|
| ebp=AAAA |
| edx=4 |

system =?

system =printf-CONST

system =libc.so:system

ret2libc

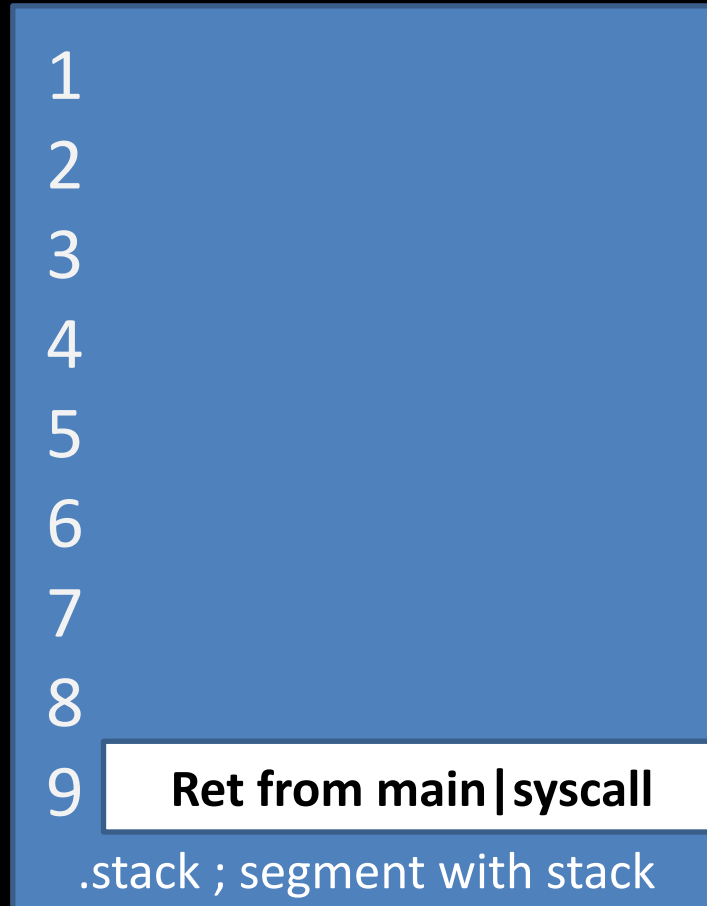
Main:

1. push ebp
2. mov ebp, esp
3. sub esp, 16
4. lea edx, buf
5. push edx
6. call gets
7. add esp, 4
8. leave
9. Ret □ eip

.....

....

10. system :
11. ...
12. ret
13. Printf:
14. ...
15. ret



| |
|-----------------|
| ebp=AAAA |
| edx=4 |

ret2libc

9. ret

.....

.... eip

10. system :

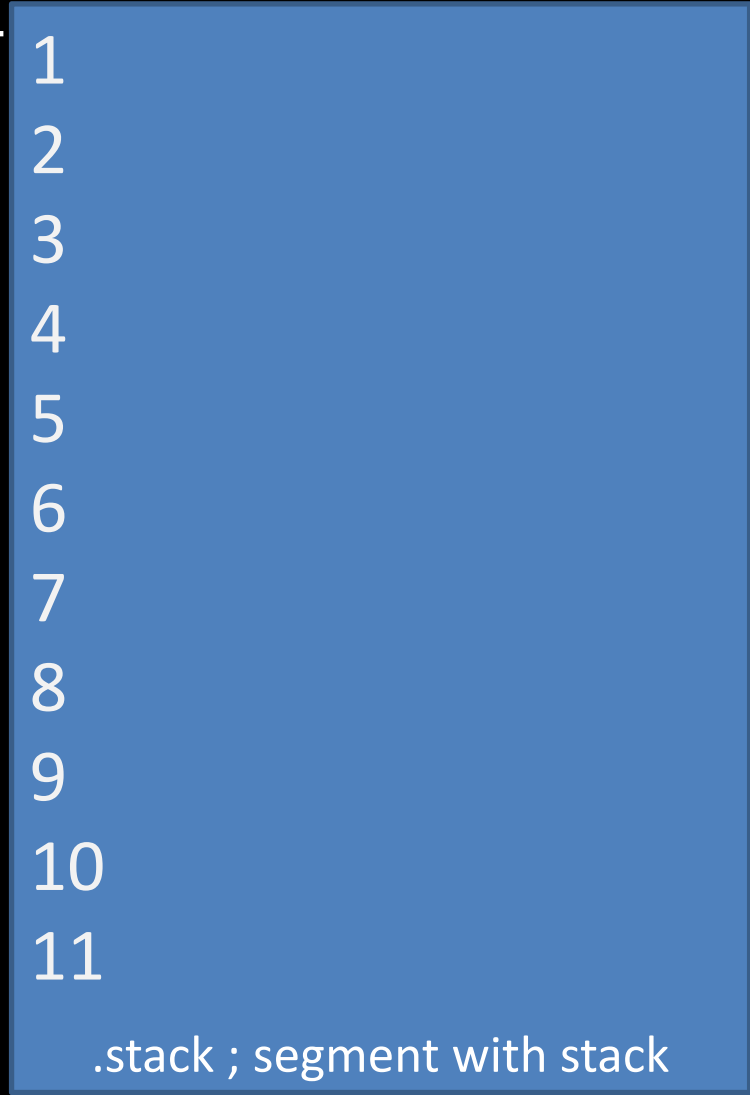
11. ...

12. ret

13. Printf:

14. ...

15. ret



| |
|-----------------|
| ebp=AAAA |
| edx=4 |

ret2libc

9. ret

.....

.... eip

10. system:

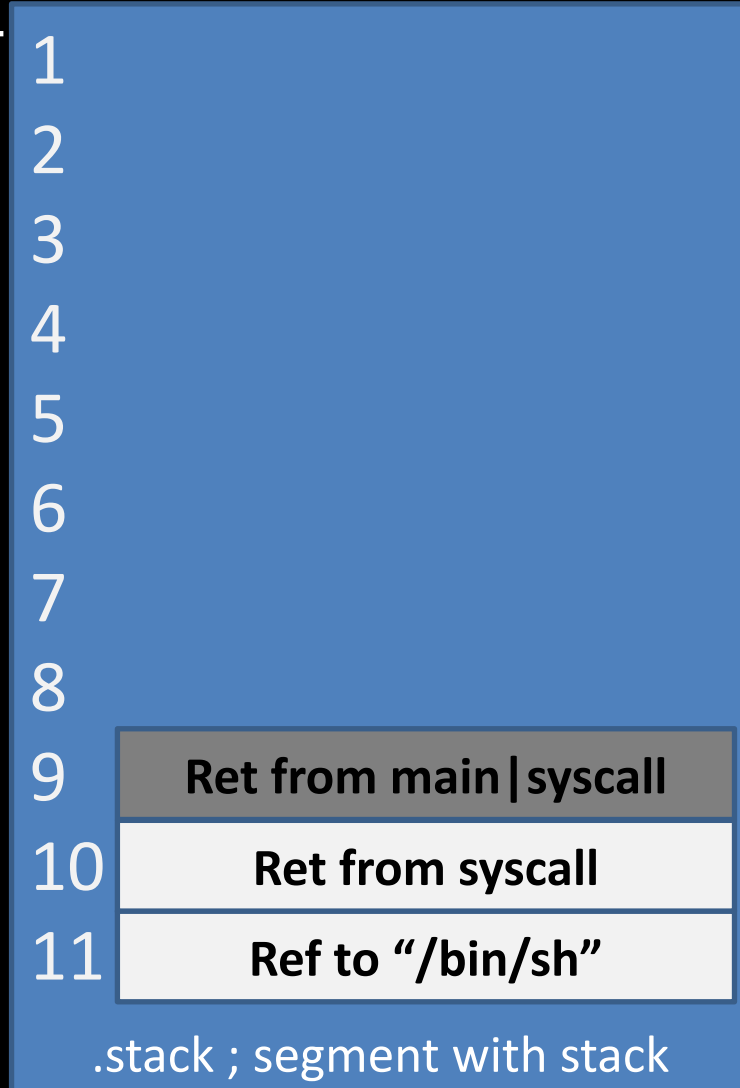
11. ...

12. ret

13. Printf:

14. ...

15. ret



| |
|----------|
| ebp=AAAA |
| edx=4 |

Where is `/bin/sh` ?

Where is /bin/sh ?

ropchain

What about randomization

`/proc/sys/kernel/randomize_va_space`

0 – No randomization. Everything is static.

1 – Conservative randomization. Shared libraries, stack, `mmap()`, VDSO and heap are randomized.

2 – Full randomization. In addition to elements listed in the previous point, memory managed through `brk()` is also randomized.

Static compile



Static compile (-s)



Static compile (-s)

1. Works in any linux with any libc installed
2. ELF contains entire libraries
3. Very big binary
4. You can find many functions and gadgets - dangerous

GDB commands

`gdb:`

`maint info sections` – show sections

`shell ps aux | grep test` – show process pid

`cat /proc/[PID]/maps` – show sections of process

`find [START ADDRESS], [END ADDRESS], “[STRING]”`

`shell:`

`ldd test`

Now

```
#include <stdio.h>
```

EXPLOIT this

```
int main(){  
    char buf[16];  
    gets(buf);  
    puts(buf);  
    return 0;  
}
```