

This source code

1. Print 'hello world' greeting 2 times to the console
2. Pops a shell '/bin/sh'

```
section .data
    greeting: db "hello world",0xa ; Greeting message for user
    mlen equ $-greeting           ; For use with write() later

    binSH: db "/bin/sh" ; Command for execve()

section .text
    global _start

_start:
    mov al,0x2 ; Maximum loop value
    jmp GreetUser ; Jumps to GreetUser subroutine

GreetUser:
    push eax ; Saves the current loop value to the stack

    mov al,0x4 ; Syscall # for write
    mov bl,0x1 ; #1 means STDOUT
    mov ecx,greeting ; Pointer to string containing hello world
    mov dl,mlen ; Length of the message to be displayed
    int 0x80 ; Calls kernel

    pop eax ; Pop the current loop value from the stack to eax
    dec eax ; Decrements the current loop value by 1
    jnz GreetUser ; If zero flag isn't set aka loop value isn't 0, GreetUser

PopShell:
    ; Zeroes the eax to edx register
    xor eax,eax
    xor ebx,ebx
    xor ecx,ecx
    xor edx,edx

    ; execve("/bin/sh",0,0)
    mov al,0xb ; Syscall # for execve
    mov ebx,binSH ; "/bin/sh"
    int 0x80 ; Calls kernel

Exit:
    ; Having an exit() is necessary, else program will segfault
    ; Zeroes eax,ebx register
    xor eax,eax
    xor ebx,ebx

    ; exit(0)
    mov al,0x1 ; Syscall # for exit
    int 0x80 ; Calls kernel
```

Filter shellcode from objdump

```
tao@kali:~/myshellcode$ objdump -D -M intel control | grep "[0-9a-f]:" | grep -v "Dis*" | cut -d '$\t' -f2 | tr -d ' \n' | sed 's/./\x&/g'
\x00\x02\xeb\x00\x50\xb0\x04\xb3\x01\xb9\x00\xa0\x04\x08\xb2\x0c\xcd\x80\x58\x48\x85\xc0\x75\xec\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\xb0\xbb
\x31\xdb\xb0\x01\xcd\x80\x68\x65\x6c\x6c\x6f\x20\x77\x6f\x72\x6c\x64\xa2\x2f\x62\x69\x6e\x2f\x73\x68tao@kali:~/myshellcode$
```