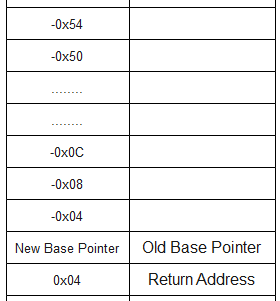
Al abrir el **STACK1\_VS\_2017**.**exe** en el **IDA** lo primero que vemos es el **Epílogo** de la función.



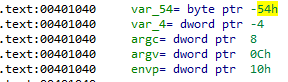
Aquí se pushea el **viejo base pointer** de alguna rutina anterior, y luego se fija el **nuevo base pointer** de esta función. A continuación se reserva el espacio de memoria que se va a necesitar, son **0x54 (84) bytes**.

Sabemos que al entrar a una función se pushea la **dirección de retorno,** por lo que el stack nos va a quedar así:

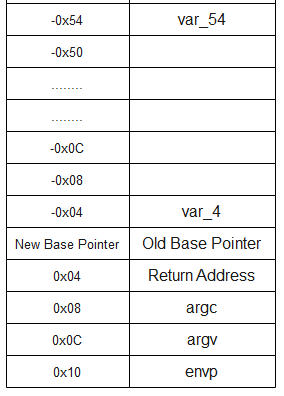


**EBP** que ahora es el **nuevo base pointer** queda fijo, y nos servirá para apuntar a las **variables locales**, y desplazarnos por el **stack frame** para lo que sea necesario.

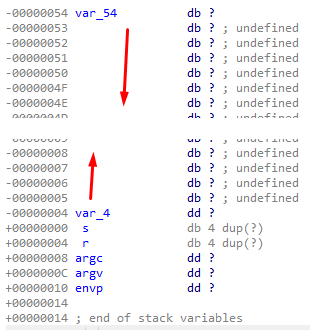
El **IDA** al entrar a una función nos muestra al comienzo la declaración de variables y también los argumentos.



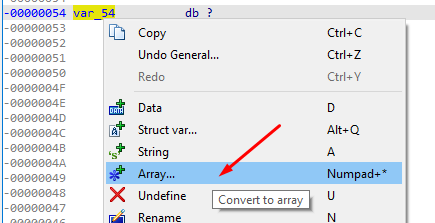
Ahí nos muestra el offset en el stack donde estarán ubicados, ponemos estos datos en la tablita y nos queda así:

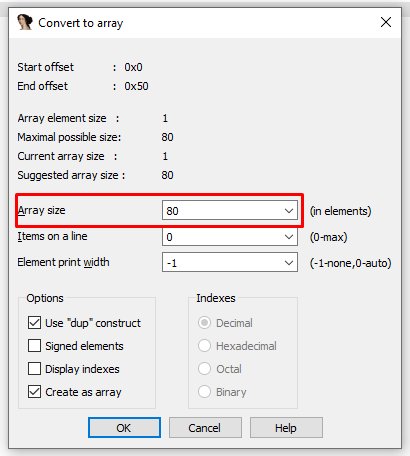


Esto mismo ya lo hace el **IDA** automáticamente :P si hacemos doble clic en cualquier **variable local** nos enviará a la ventana del stack.



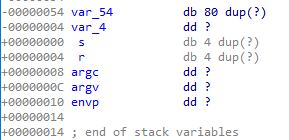
Aquí podemos ver que el tamaño de la variable **var\_54** es de **0x50 (80) bytes,** podemos convertirlo en un **array,** y así lo vemos mejor :P





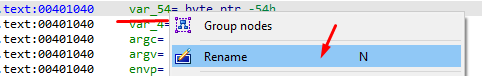
**IDA** automáticamente calcula el tamaño de esta variable, que es de **80** bytes.

y ahora si ya lo vemos mas lindo al stack

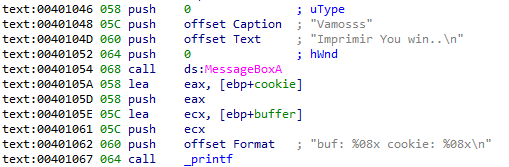


A **var\_54** lo renombramos a **buffer**, y a **var\_4** lo llamaremos **coockie,** para hacer esto nos paramos sobre la variable y presionamos la tecla **N**, o elegimos **rename** desde el menú

que se despliega con clic derecho.



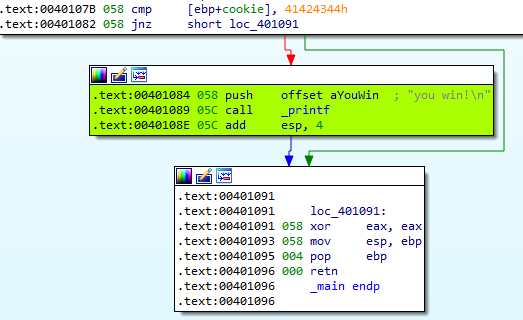
Seguimos con el análisis, vemos que aparece un  **MessageBox** al que deberemos aceptar para que se imprima en la consola un mensaje con las direcciones de las variables **buffer**  y **coockie.**

****

Luego pushea un puntero al comienzo de la variable **buffer** que se la pasa como argumento a la función **gets.**

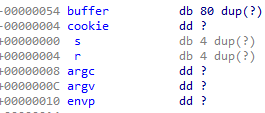


Finalmente se fija si la variable **coockie** es igual a **0x41424444,** en caso de que sea igual entonces mostrara el mensaje de **“you win!”**



Caso contrario el programa terminará sin hacer más nada.

Muy bien, recordemos como teníamos la pila



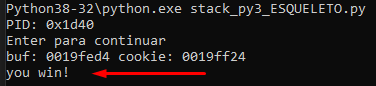
La función **gets** no comprueba la longitud del búfer. Podemos aprovechar esto y desbordar la variable **buffer** para pisar la variable **cookie** con el contenido **0x41424344.**

Entonces enviaremos 80 bytes más 0x41424344.

Mi script en python para explotar este ejercicio es el siguiente:

| import sys from subprocess import Popen, PIPE  buffer = b"A" \* 80 cookie = b"\x44\x43\x42\x41"  payload = buffer + cookie  p1 = Popen("STACK1\_VS\_2017.exe", stdin=PIPE) print ("PID: %s" % hex(p1.pid)) print ("Enter para continuar")  p1.communicate(payload) p1.wait() input() |
| --- |

Una vez que acepte el messagebox veremos el mensaje de **“you win!”**

****

El siguiente ejercicio es [02 Stack Nro 2](https://docs.google.com/document/d/1mhYTbhJ0FW-7oGWoZ5MxBMGb3u5lXNwzp00fF0KxKDM/edit?usp=sharing)