Pasamos al **ABO3**, en el **prólogo** comienza reservando **0x104** bytes para la variable **buffer** y **fn.**



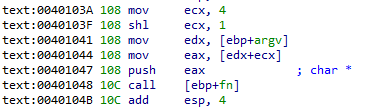
Seguimos teniendo la función vulnerable **gets,** que recibe como parámetro la variable **buf**

****

Aquí vemos una segunda variable **fn** que recibe un puntero a la función **puts**

****

Después llama a la función **fn**

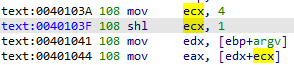
****

Primero se desplaza a la izquierda 1 bits al 4 de **ecx**. Podemos usar el **IDA** y hacer esta cuentita:



Nos da **8**, a este resultado lo dividimos por **4** y nos indicará cual es el **argumento** que se está por usar, en este caso es **argv[2]**

Finalmente mueve el contenido de **argv[2]** en **eax**



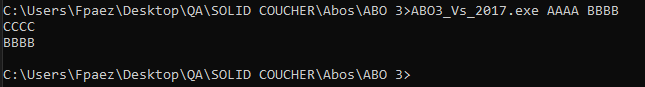
En fin, lo que le pasemos al argumento **argv[2]** lo va a mostrar con la función **puts** por pantalla.



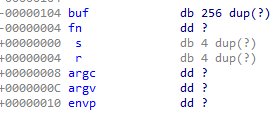
Terminando con la función **exit**, así que no vamos a poder pisar el **ret** en este caso.

Bien, ahora sabemos que el **abo** recibe dos parámetros, **argv[1]** y **argv[2]**.

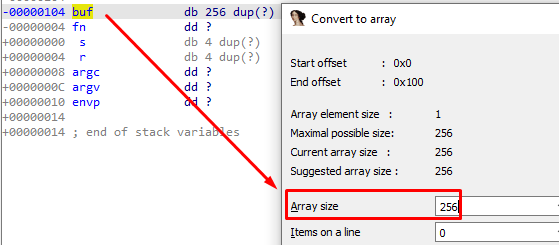
Así, que le mandamos dos cadenas, **“aaaa”** y “**bbbb”**, cuando demos el **enter**, se va a poner en pausa así introducimos el **buffer** “**cccc”** que tomará con el **gets** al presionar el **enter,** y por último imprime **argv[2]** por pantalla**.**

****

Esta es la disposición de las variables en el **stack frame**

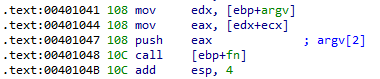


La idea es pisar el puntero en la variable **fn**, esto lo lograremos desbordando la variable **buf**.



Con **0x100 bytes** llenamos la variable **buff**, y con **4 bytes** más pisamos el puntero de la variable **fn.**

Vemos que al momento de llamar a la función **fn**, se le pushea como argumento **eax,** por lo que al llegar a este **call** en **eax** vamos a tener el puntero a **argv[2]**

****

Entonces ahora la idea sería así:

1. Desbordamos la variable **buff** para pisar el puntero de la variable **fn**
2. A este puntero lo pisamos con la dirección de algún **CALL EAX o Push eax/ret, o algún jmp eax**.



Bueno, debería haber funcionado! Pero no funcionó porque el **payload** que le pasamos por **argv[2]** tiene caracteres inválidos, por ejemplo hay un **0x0C** que me corta el **payload.**

Vamos a reinventarnos, hay una función **system** que parece estar al pedo, pero en realidad nos va a servir mucho.

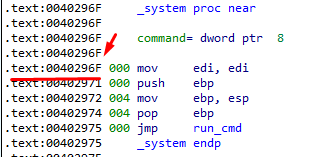


Según la Wiki esta función sirve para ejecutar subprocesos o comandos del sistema operativo.

Así que en vez del **payload**, vamos a pasarle como argumento a la función **fn** la string **“calc”**

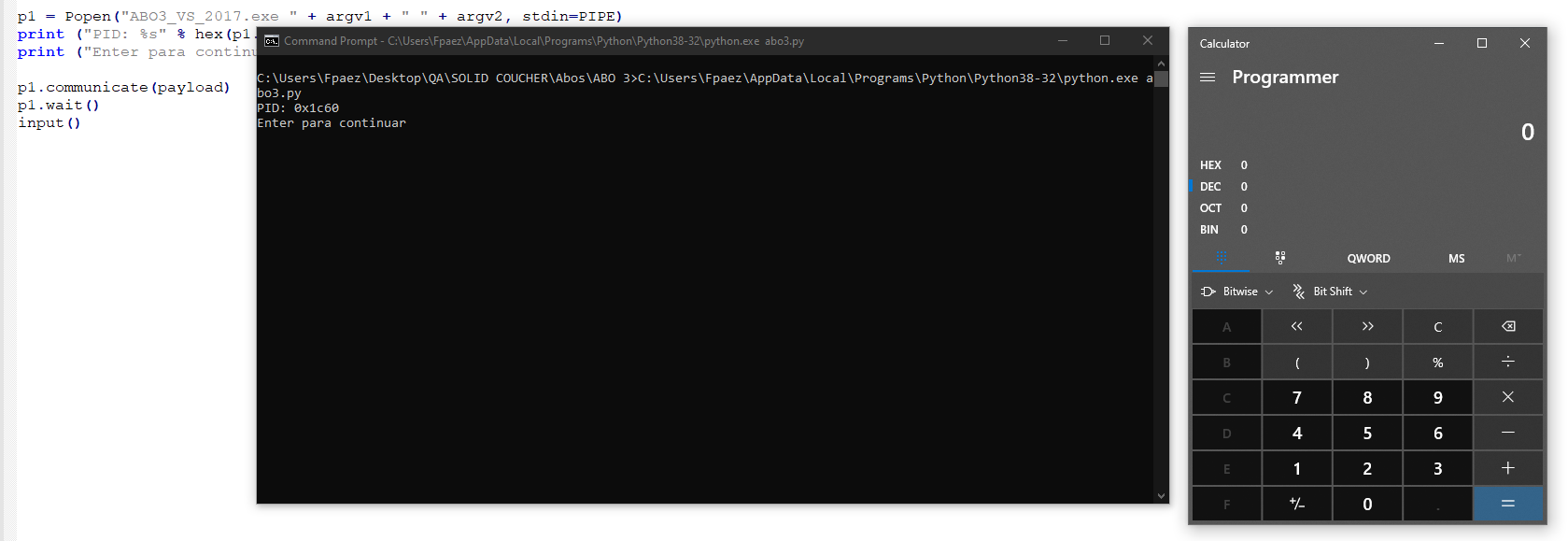
****

Debemos pisar el puntero de **fn**  con la dirección de **system**



Entonces lo que hicimos en realidad fue modificar el puntero de **fn**, que originalmente apuntaba a **puts**, y lo reemplazamos por el de **system.**

| import sys import binascii from subprocess import Popen, PIPE  buff = b"C" \* 0x100 argv1 = "AAAA" argv2 = "calc" pointer\_fn = b"\x6F\x29\x40\x00" # system  buffer = buff + pointer\_fn   payload = buffer  p1 = Popen("ABO3\_VS\_2017.exe " + argv1 + " " + argv2, stdin=PIPE) print ("PID: %s" % hex(p1.pid)) print ("Enter para continuar")  p1.communicate(payload) p1.wait() input() |
| --- |

****