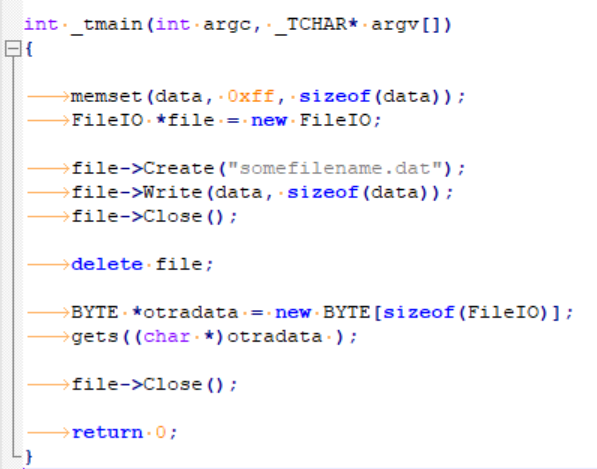
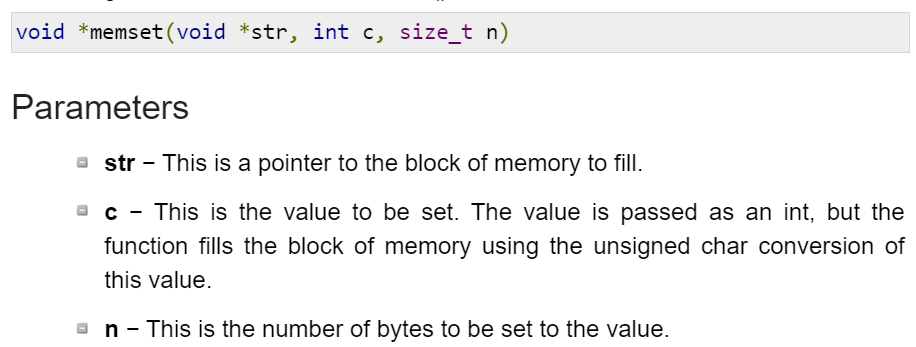
Tenemos el codigo de este ABO

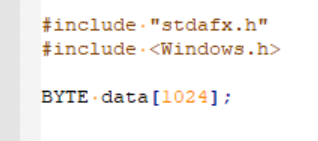


Vamos a poner primero nombres a las variables en el IDA.

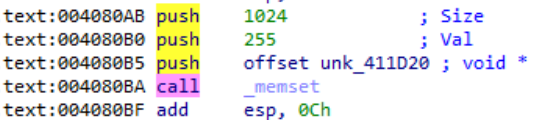
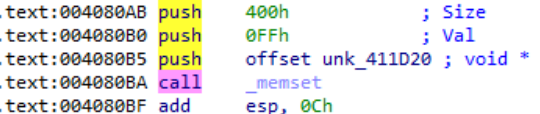
Analicemos primero la funcion **main**, como se ve el el **source** comienza con una **funcion memset**



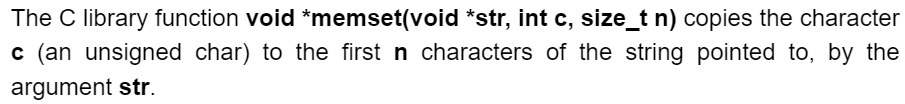
Tiene tres argumentos, el primero que recibe es la variable **data,** que como se ve en el **source** es una **variable global**, ya que esta definida afuera de las funciones. La variable **data** tiene un tamaño de **1024 bytes.**



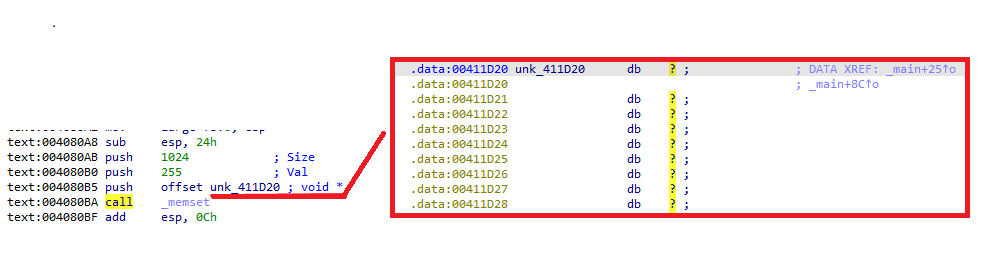
En **IDA** ubicamos esta función aquí. Haciendo clic derecho sobre las constantes podemos pasarlo de hexadecimal a decimal a los valores.



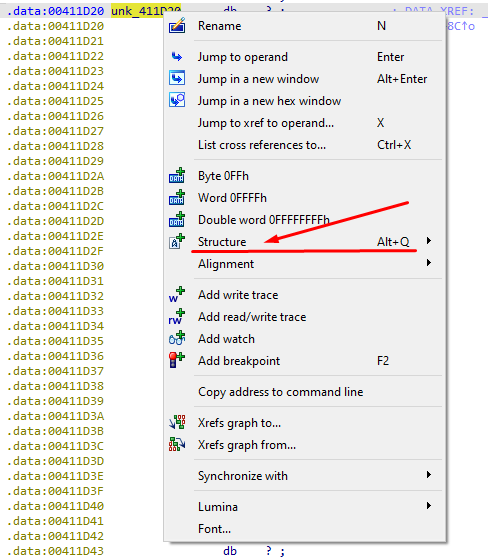
Ahí vemos que el **tercer argumento** es el **size,** en el **source** el equivalente en **ida** seria el **sizeof(data) = push 1024**.



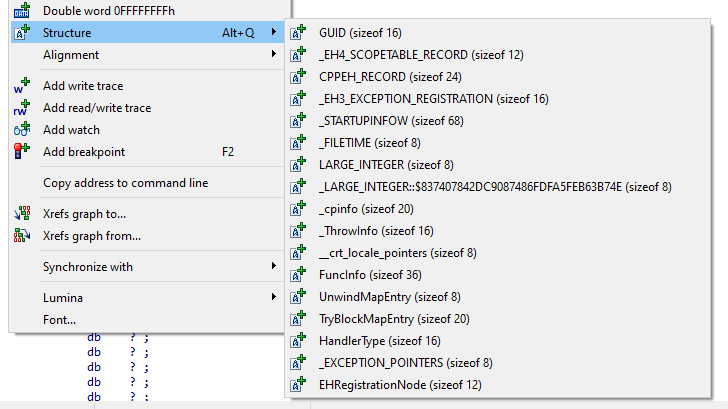
El **primer argumento** es la **variable global data** que tiene un tamaño de **1024 bytes.** En **IDA** si hacemos doble clic sobre esta variable podemos ver que es un **array de bytes**.



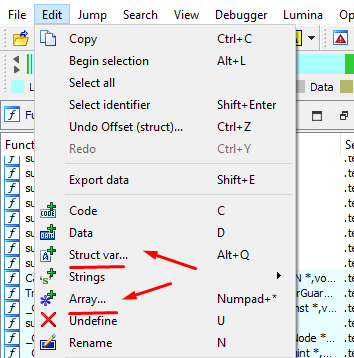
Al ser una **variable global** esta definida en la **seccion data**, y en el **menu de ida** no aparece la opcion para **convertirlo en un array de bytes**. En su lugar lo podemos convertir en una **estructura de bytes :P**



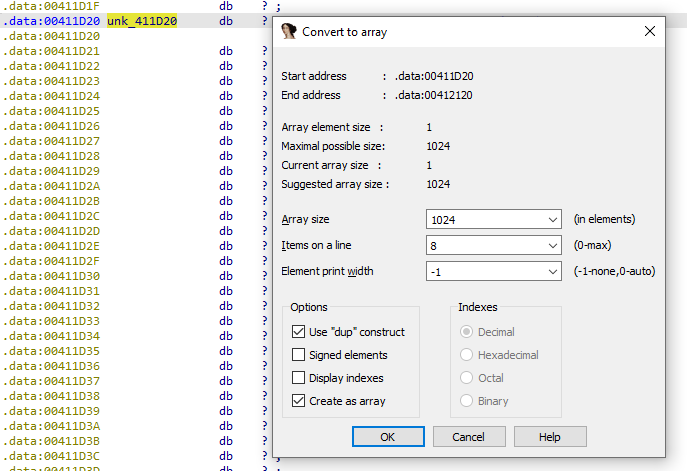
El tema es que si elegimos la opcion **structure** solo aparecen las estructuras ya definidas.



Entonces debemos de **crear una estructura de 1024 bytes!** A esta etructura la llamaremos **data.** Esto lo podemos hacer desde el **menu edit**.



Bueno, aqui si se ve la opcion de **crear un array**, pero tambien tenemos la opcion de **crear una estructura.** Ya que podemos lo hagamos con la opcion de **array.**



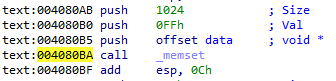
Una vez que le damos a **ok** nos va a quedar asi



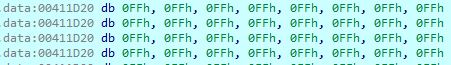
Le podemos **renombrar a data.**



Ahora la funcion **memset** nos va a quedar asi



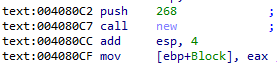
Al ejecutar el **memset** este **array data** se llenara con **1024 bytes** de **0xFF**



Despues tenemos la siguiente linea

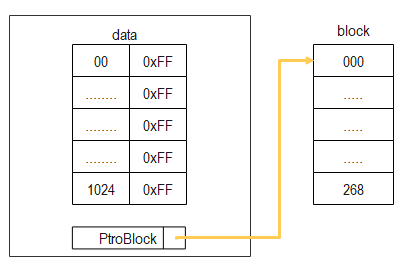


En **IDA,** vemos esto

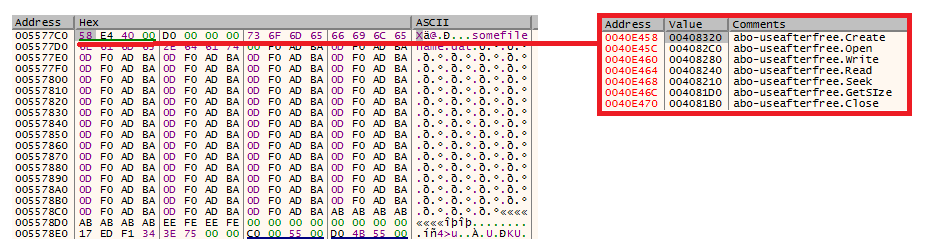
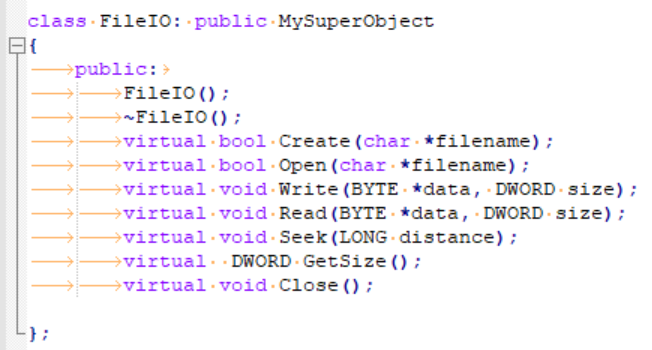


Esta funcion **new** no es mas que la funcion **malloc**, por lo que **alloca** **268 bytes** y retorna en **eax** la direccion en donde se allocaron, guardando esta en la variable **block**.

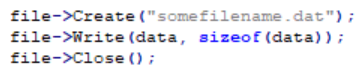
Renombrare a **block** como **PtroBlock. Block** Sera de tipo **FileIO.**



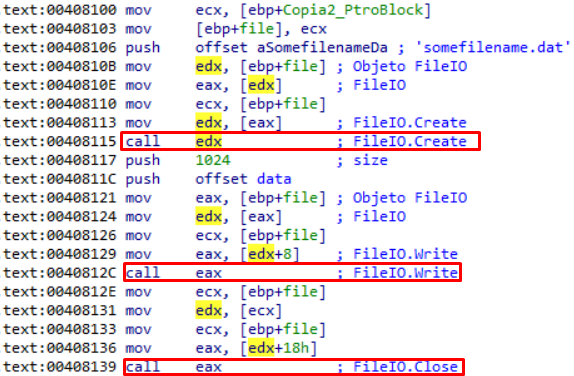
En el **primer dword** del **buffer block** se seteara una direccion del objeto **FileIO** que apuntara al comienza de las funciones publicas declaradas en dicha clase.



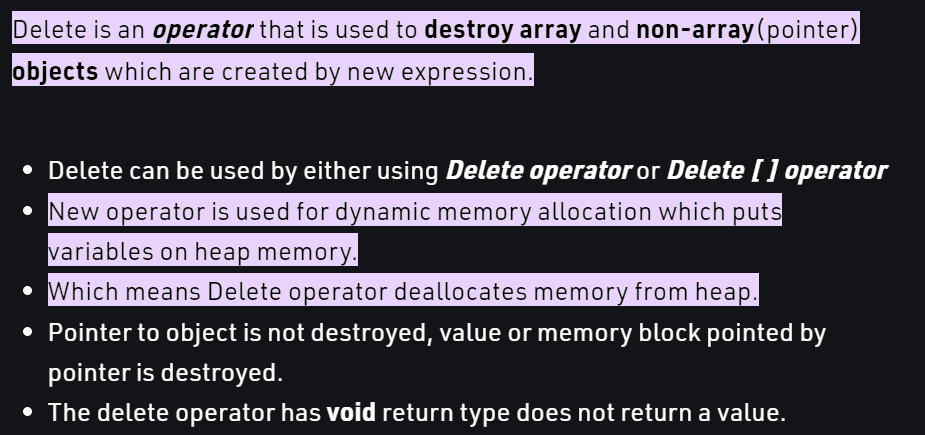
Luego accede a esa tablita cada vez que necesita ejecutar alguna funcion, por ejemplo en el codigo ahora tenemos estas 3 funciones ejecutadas una seguida de la otra.



En ida tenemos esta parte aquí



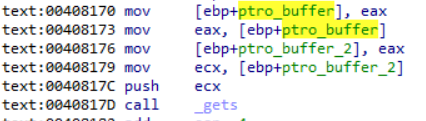
Despues se libera el puntero 5 del objeto file



Continua en el codigo con la siguiente linea



Esto nos devolvera un puntero que se va a guardar aquí

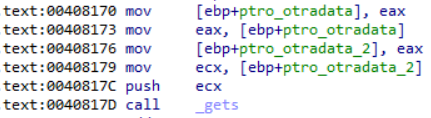


Vemos que al final **ptro\_buffer\_2** termina valiendo lo mismo que **ptro\_buffer**, y usa este buffer como argumento en la llamada a **gets.**

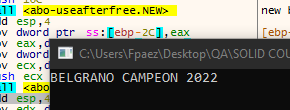


Para seguir con los mismos nombres que el codigo lo voy a renombrar a **otradata.**

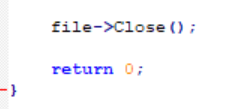
Lo dejare asi



Luego aquí deberemos escribir algo en la consola de windows y dar enter, esto se almacenara en la variable **otradata**

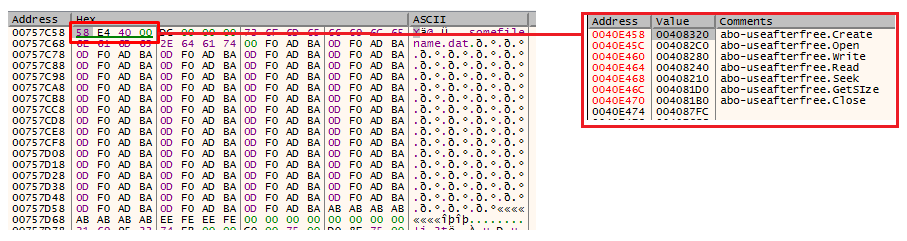


Finalmente se llama a la funcion close

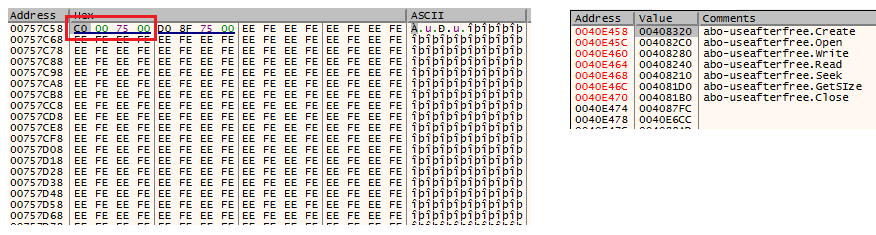


Bueno, aqui ya veo **algo sospechoso**, como vimos antes el puntero al bloque de memoria apuntado por **file** se destruyo con **delete,** y aquí se vuelve a llamar para invocar a **close.**

Antes destruirse con delete estaba asi:

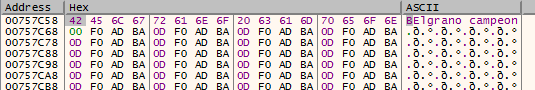


Despues del delete quedo asi:



Como se ve el puntero al bloque de funciones del objeto se libero.

Luego cuando se vuelve hacer un **new** y el **gets** queda asi:

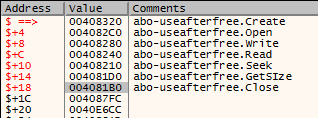


En IDA tenemos asi

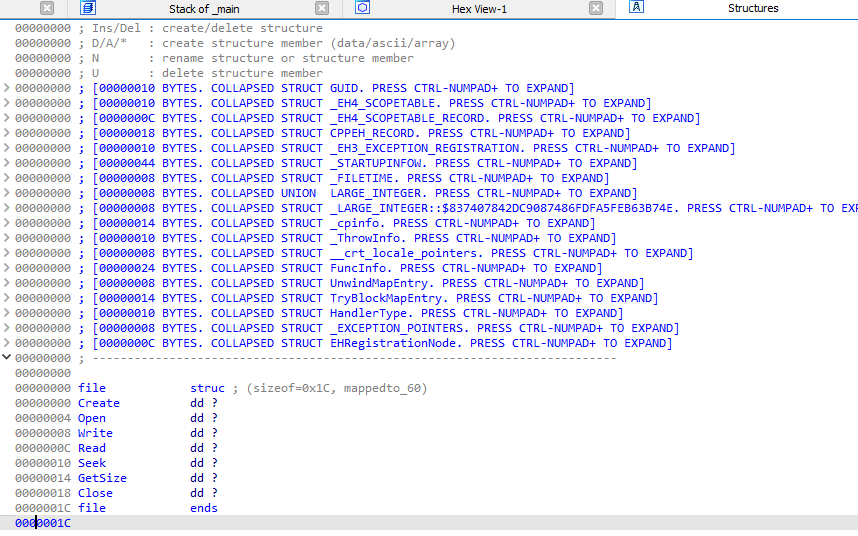


Voy hacer algo aqui, a crear una estructura del objeto file, asi lo vemos mejor en IDA.

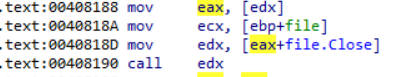
Sabemos que las funciones del objeto estan asi:



Nuestra estructura quedara como esa tablita, y la llamare file tambien



Una vez creada podemos asignarle con Ctrl+T al objeto file para saber las funciones a la que apunta.

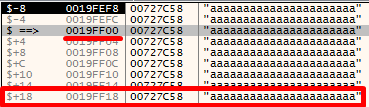


Okey, entonces los primero **4 bytes** que tome con el **gets** sera un **puntero que podemos controlar**, y lo tendremos en **eax**, por lo que si no pasamos un puntero correcto, ese **call edx va a fallar.**

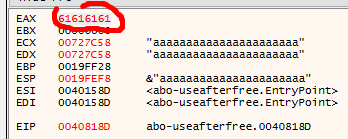
Bien, en este espacio de memoria que se asigno con el operador **new** tenemos **268 bytes de espacio**, a los primero **4 bytes** lo usaremos como **un puntero a donde queramos saltar**.

Despues de este puntero, podemos poner nuestra **shellcode de la calculadora.**

Si nos fijamos en la pila, vemos que que hay muchos punteros al espacio de memoria donde se escribe con el **gets.** Este **stack no esta randomizado**, por lo que generalmente sus direcciones siempre seran los mismos, y podemos usarlos.



Nuestros primeros 4 bytes deben ser el puntero 0x19FF00, lo que pongamos ahí lo tendremos en EAX,



yo elegi este valor porque no se rompe cuando se lo tracea a dichos bytes… Bueno, a este valor que esta en EAX se le va a sumar 0x18 y el contenido de este resultado sera 0x727C58 que es donde tenemos nuestra shellcode.

Si probamos veremos que se ejecuta sin problemas la calculadora…

