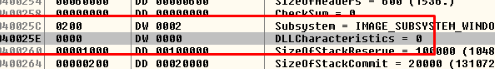
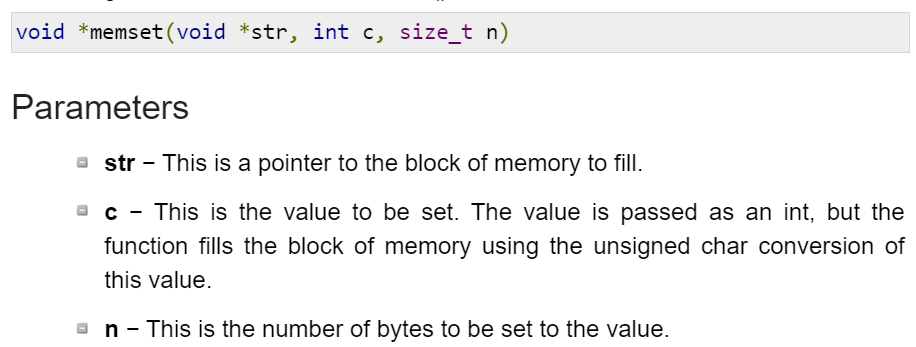
Abrimos este ejercicio, que no tiene ASLR

Ya que le hemos cambiado esta propiedad a cero



No tengo el codigo fuente, asi que vamos ir viendolo en IDA

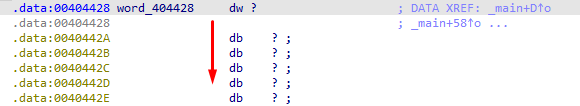
Al comienzo del **main** tenemos un llamado a la función **memset**



Lo que hace es rellenar un buffer apuntado por **\*str** y lo rellena con **int c** hasta **size\_t n.**

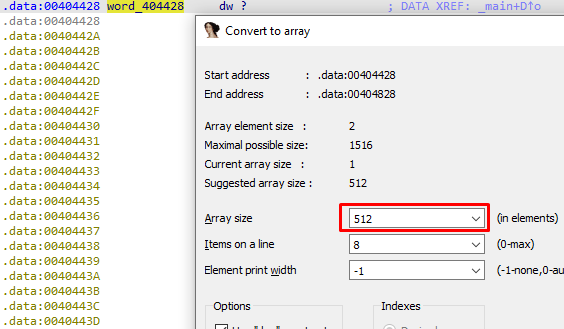
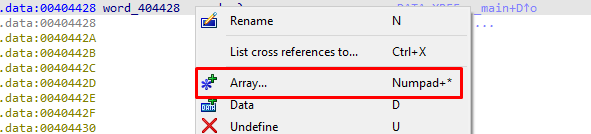


Ahi vemos que **404428** es una **variable global**, lo sabemos por si hacemos doble clic sobre ella nos lleva aquí



Y como se ve esta declarada en la **seccion data,** que es el lugar donde estaran las **variables globales**.

Podemos ver el tamaño del mismo convirtiendolo en un **array,** y en este caso **IDA** lo detecta como si tuviera **512 bytes de tamaño.**



Prestemos atencion! Ahi vemos que esta declarado como **word ¡!!** Por eso son **512** que si lo multiplicamos por 2 nos daria **1024 bytes ¡!!**

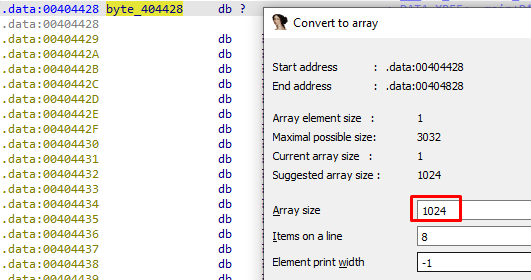
Asi que me posiciono arriba de la variable



Y presiono varias vece la tecla “**d”** hasta que tengamos **“db”**



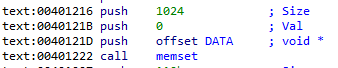
Y ahora lo volvemos a convertir en un **array**



Vamos a ponerle un nombre a esta variable, nos posicionamos sobre ella y presionamos la tecla “**N”**



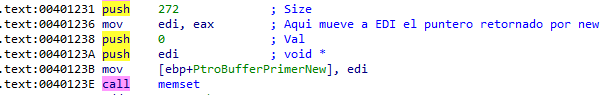
Bueno volvamos al codigo en **main,** el primer argumento es el **offset de DATA,** osea el puntero, el segundo sera con lo que se quiere rellenar a **DATA,** en este caso se llenara con muchos **0,** tantos ceros como indique el **size** en el tercer argumento, que son **1024.**



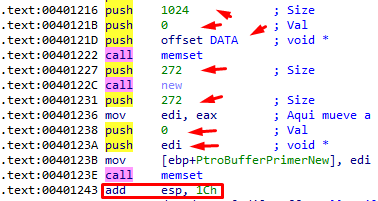
Seguimos y encontramos un **operador new**, que reservara **272 bytes** de memoria



Despues hace **memset** nuevamente, el primer argumento sera el **puntero retornado por new,** el segundo es un CERO, de esta forma se llenara este nuevo buffer con **272 bytes** de CEROS. Finalmente mueve el **puntero del primer new** a **var\_4,** lo renombrare como **PtroBufferPrimerNew**



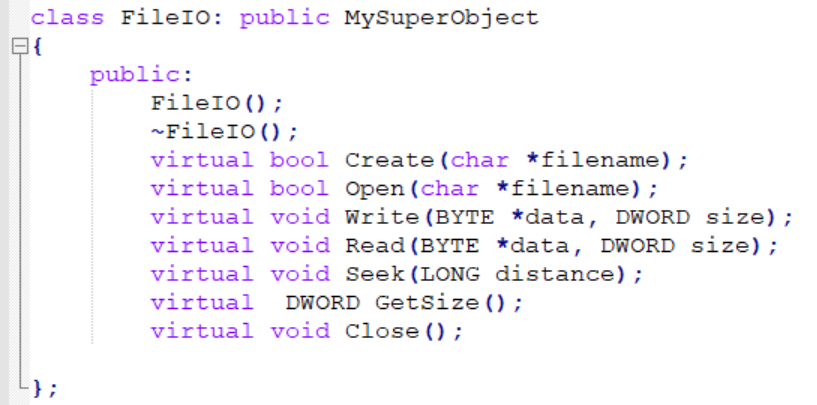
Aqui hace un **ADD ESP,1C**. No es mas nada para liberar memoria de todos los argumentos que se han pasado con **PUSH.** Si dividimos **0x1C / 4**  nos dara **7,**  que es el total de argumentos que se usaron hasta este punto.



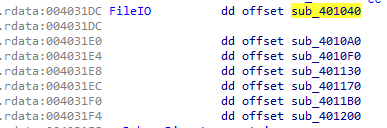
Hay un **puntero a FileIO**



Como sabemos ya por el ejercicio anterior se trata de la **clase FileIO,** Que tiene 7 funciones declaradas.

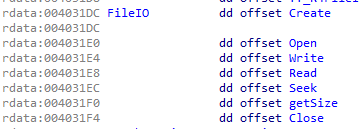


Si hacemos doble clic sobre este puntero a **FileIO** podemos ver estas 7 funciones pero sin el nombre asignado.

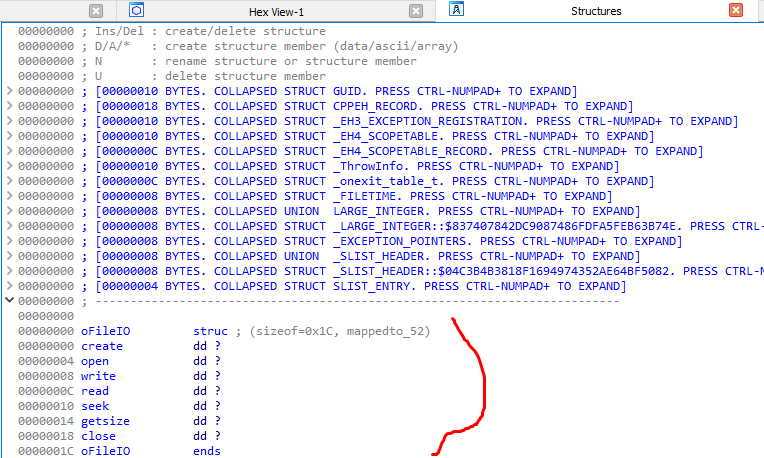


Podemos ir a cada uno de ellos y ver de cual se trata… Yo ya lo hice, asi que voy a ponerle el nombre directamente…

Me quedo asi



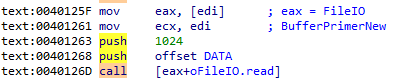
Voy hacer una estructura FileIO, asi puedo renombrar despues en el codigo y que se vea mas lindo.



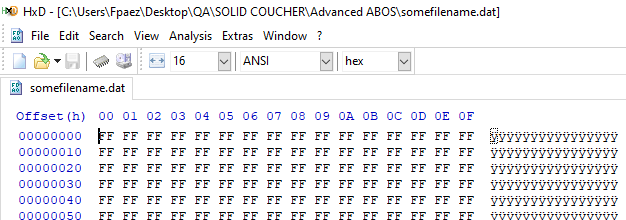
Despues vemos que crea un archivo con **Create**



Despues llama a read, aqui lee **1024 bytes de DATA**



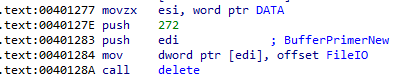
Si me fijo en ese archivo veo que esta todo lleno de **0xFFFFFFFF**, por lo que llenara el buffer **DATA** todo con **FFFFFFFF**



Despues cierra el archivo

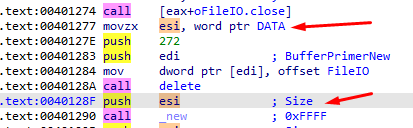


Y libera el **BufferPrimerNew**



Luego tenemos una llamada a la funcion **New**, y se le pasa por argumento el **size 0xFFFF,** osea un size de **65535 bytes** de espacion que va a tener este nuevo **buffer.**

Como antes se hizo un **delete,** al hacer ahora un **new** el espacio de memoria que se reservara sera **probablemente** en la misma direccion que el **buffer** que fue eliminado con **delete.**



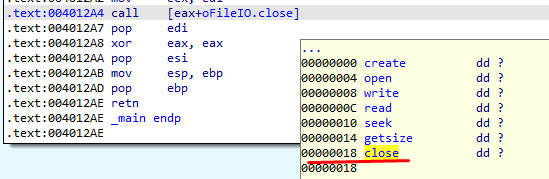
A continuacion tenemos el **gets\_s** que recibe este **segundo buffer** creado por **new** y que tendra un **tamaño nomas de 0xFFFF (65535).**



Entonces una vez que metamos algo con el **gets\_s,** los primeros **4 bytes** se usaran como **puntero,** podremos manejar el flujo del programa como queramos.

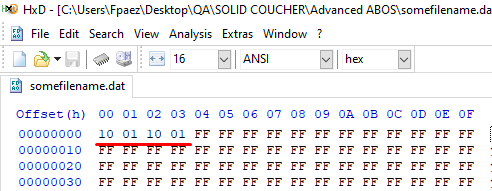


Vemos que termina haciendo una llamada a **[eax + 18]**



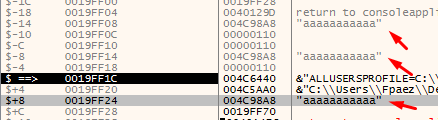
La direccion que deberia devolvernos el segundo **new** en este caso no es igual al del primero -.-… Esto ocurre porque los dos **new** tienen diferentes **sizes,** cuando deberian de ser iguales para que tengan la misma direccion…

Sabemos que este size lo saca del archivo que se creo, vamos a modificarlo para que tenga el mismo size.



Ahora van a ser iguales los dos sizes, y va a calcular la misma direccion en ambos news.

Mi pila al momento de llegar al call lo tengo asi con estos punteros al buffer



Hay que elegir entre todos los punteros que hay en el stack que son muchos, y descontarle 0x18, ese es el valor del stack que debemos usar en el script…

