

**Федеральное государственное автономное образовательное учреждение
высшего образования**

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. Тихонова

Департамент электронной инженерии

ОТЧЕТ

О ПРАКТИЧЕСКОЙ РАБОТЕ №3

по дисциплине «Системное программирование»

«Основы дизассемблирования»

Студент гр. БИБ201

Шадрунов Алексей

Дата выполнения: 12 января 2023 г.

Преподаватель:

Смирнов Д. В.

«__» _____ 2023 г.

Москва, 2023

Содержание

1	Задание на практическую работу	3
2	Ход работы	3
2.1	Выделение памяти с помощью malloc в main	3
2.2	Обработка команды в fgets	5
2.3	Вызов API-функции CreateFile	5
2.4	Вызов API-функции WriteFile	7
2.5	Вызов API-функции ReadFile	8
2.6	Найти утечку ресурсов в программе	9
2.7	Работа программы	10
3	Выводы о проделанной работе	13
	Приложение А	14

1 Задание на практическую работу

Скомпилировать программу на языке C. Дизассемблировать полученный исполняемый файл с помощью x64dbg. Проследить за работой программы с помощью отладчика и Process Monitor и Process Hacker.

2 Ход работы

2.1 Выделение памяти с помощью malloc в main

Установим точку прерывания на месте последнего вызова функции перед завершением программы (cexit). Этот вызов соответствует функции main (Рисунок 1).

00D912C9	A1 14D0D900	mov eax,dword ptr ds:[D9D014]	
00D912CE	8B15 C8E1D900	mov edx,dword ptr ds:[<.&_initenv>]	edx:EntryPoint
00D912D4	8902	mov dword ptr ds:[edx],eax	edx:EntryPoint
00D912D6	894424 08	mov dword ptr ss:[esp+8],eax	
00D912DA	A1 18D0D900	mov eax,dword ptr ds:[D9D018]	
00D912DF	894424 04	mov dword ptr ss:[esp+4],eax	
00D912E3	A1 1CD0D900	mov eax,dword ptr ds:[D9D01C]	
00D912E8	890424	mov dword ptr ss:[esp],eax	
00D912EB	E8 7A080000	call file.D91B6A	
00D912F0	8B0D 0CD0D900	mov ecx,dword ptr ds:[D9D00C]	ecx:EntryPoint
00D912F6	A3 10D0D900	mov dword ptr ds:[D9D010],eax	
00D912FB	85C9	test ecx,ecx	ecx:EntryPoint
00D912FD	0F84 CC000000	je file.D913CF	
00D91303	8B15 08D0D900	mov edx,dword ptr ds:[D9D008]	edx:EntryPoint
00D91309	85D2	test edx,edx	edx:EntryPoint
00D9130B	74 73	je file.D91380	
00D9130D	8D65 F0	lea esp,dword ptr ss:[ebp-10]	
00D91310	59	pop ecx	ecx:EntryPoint
00D91311	5B	pop ebx	
00D91312	5E	pop esi	esi:EntryPoint
00D91313	5F	pop edi	edi:EntryPoint
00D91314	5D	pop ebp	
00D91315	8D61 FC	lea esp,dword ptr ds:[ecx-4]	
00D91318	C3	ret	
00D91319	8DB426 00000000	lea esi,dword ptr ds:[esi]	esi:EntryPoint
00D91320	A1 44D0D900	mov eax,dword ptr ds:[D9D044]	
00D91325	BB 01000000	mov ebx,1	
00D9132A	83F8 01	cmp eax,1	
00D9132D	0F85 8FFEFFFF	jne file.D911C2	
00D91333	C70424 1F000000	mov dword ptr ss:[esp],1F	
00D9133A	E8 29750000	call <JMP.&_amsq_exit>	
00D9133F	A1 44D0D900	mov eax,dword ptr ds:[D9D044]	
00D91344	83F8 01	cmp eax,1	
00D91347	0F85 9AFEFFFF	jne file.D911E7	
00D9134D	C74424 04 08F0D900	mov dword ptr ss:[esp+4],file.D9F008	
00D91355	C70424 00F0D900	mov dword ptr ss:[esp],file.D9F000	
00D9135C	E8 1F750000	call <JMP.&_initterm>	
00D91361	C705 44D0D900 02000000	mov dword ptr ds:[D9D044],2	
00D91368	85DB	test ebx,ebx	
00D9136D	0F85 7CFEFFFF	jne file.D911EF	
00D91373	871D 40D0D900	xchg dword ptr ds:[D9D040],ebx	
00D91379	E9 71FEFFFF	jmp file.D911EF	
00D9137E	66:90	nop	
00D91380	E8 EB740000	call <JMP.&_cexit>	
00D91385	A1 10D0D900	mov eax,dword ptr ds:[D9D010]	
00D9138A	8D65 F0	lea esp,dword ptr ss:[ebp-10]	

Рисунок 1 – Последний вызов перед выходом

После перехода по адресу функции мы видим два вызова (соответствуют функциям printf в исходном файле), инициализацию переменных и функции выделения памяти malloc (Рисунок 2). Выделяется память под переменную buf (куда попадают вводимые данные с консоли) и переменную command (первое слово из buf).

После вызова malloc в регистр EAX сохраняется адрес выделенной ячейки памяти. На рисунке 3 видно, что этот участок памяти заполнен случайными значениями.

00D91B6A	55	push ebp		
00D91B6B	89E5	mov ebp,esp		
00D91B6D	53	push ebx		
00D91B6E	83E4 F0	and esp,FFFFFFF0		
00D91B71	83EC 50	sub esp,50		
00D91B74	E8 47040000	call file.D91FC0		
00D91B79	C70424 79A1D900	mov dword ptr ss:[esp],file.D9A179	D9A179:"main\n"	
00D91B80	E8 98F9FFFF	call file.D91510		
00D91B85	C74424 40 00000000	mov dword ptr ss:[esp+40],0		
00D91B8D	C74424 4C 00000000	mov dword ptr ss:[esp+4C],0		
00D91B95	C74424 3C 00000000	mov dword ptr ss:[esp+3C],0		
00D91B9D	C74424 38 00000000	mov dword ptr ss:[esp+38],0	[esp+38]:EntryPoint	
00D91BA5	C74424 34 00000000	mov dword ptr ss:[esp+34],0	[esp+34]:EntryPoint	
00D91BAD	C74424 23 44616E69	mov dword ptr ss:[esp+23],696E6144		
00D91BB5	C74424 27 6C536D69	mov dword ptr ss:[esp+27],696D536C		
00D91BBD	C74424 2B 726E6F76	mov dword ptr ss:[esp+2B],766F6E72		
00D91BC5	C74424 2F 2E747874	mov dword ptr ss:[esp+2F],7478742E		
00D91BCD	C64424 33 00	mov byte ptr ss:[esp+33],0		
00D91BD2	C74424 16 48656C6C	mov dword ptr ss:[esp+16],6C6C6548		
00D91BDA	C74424 1A 6F20776F	mov dword ptr ss:[esp+1A],6F77206F		
00D91BE2	C74424 1E 726C640A	mov dword ptr ss:[esp+1E],A646C72		
00D91BEA	C64424 22 00	mov byte ptr ss:[esp+22],0		
00D91BEF	C74424 48 00000000	mov dword ptr ss:[esp+48],0		
00D91BF7	C74424 10 FFFFFFFF	mov dword ptr ss:[esp+10],FFFFFFF		
00D91BF7	C74424 44 00000000	mov dword ptr ss:[esp+44],0		
00D91C07	C70424 7FA1D900	mov dword ptr ss:[esp],file.D9A17F	D9A17F:"malloc\n"	
00D91C0E	E8 0AF9FFFF	call file.D91510		
00D91C13	C70424 81000000	mov dword ptr ss:[esp],81		
00D91C1A	E8 C96C0000	call <JMP.&malloc>		
00D91C1F	894424 3C	mov dword ptr ss:[esp+3C],eax		
00D91C23	C70424 81000000	mov dword ptr ss:[esp],81		
00D91C2A	E8 B96C0000	call <JMP.&malloc>		
00D91C2F	894424 38	mov dword ptr ss:[esp+38],eax	[esp+38]:EntryPoint	
00D91C33	C70424 87A1D900	mov dword ptr ss:[esp],file.D9A187	D9A187:"waiting for user input...\n"	
00D91C3A	E8 DEF8FFFF	call file.D91510		
00D91C3F	C70424 00000000	mov dword ptr ss:[esp],0		
00D91C46	A1 4890D900	mov eax,dword ptr ds:[D99048]		
00D91C48	FFD0	call eax		
00D91C4D	8B5424 3C	mov edx,dword ptr ss:[esp+3C]		
00D91C51	894424 08	mov dword ptr ss:[esp+8],eax		
00D91C55	C74424 04 7E000000	mov dword ptr ss:[esp+4],7E	7E: '~'	
00D91C5D	891424	mov dword ptr ss:[esp],edx		
00D91C60	E8 536C0000	call <JMP.&fgets>		
00D91C65	8B4424 3C	mov eax,dword ptr ss:[esp+3C]		
00D91C69	894424 04	mov dword ptr ss:[esp+4],eax		
00D91C6D	C70424 A2A1D900	mov dword ptr ss:[esp],file.D9A1A2	D9A1A2:"buf is %s\n"	

Рисунок 2 – Выделение памяти с помощью malloc

EIP

007C1C1A

007C1C1F

007C1C23

007C1C2A

007C1C2F

007C1C33

007C1C3A

007C1C3F

007C1C46

007C1C48

007C1C4D

007C1C51

007C1C55

007C1C5D

007C1C60

007C1C65

007C1C69

007C1C6D

007C1C74

007C1C79

007C1C81

007C1C83

007C1C88

007C1C8C

007C1C8F

007C1C94

007C1C98

007C1C9A

E8 C96C0000

894424 3C

C70424 81000000

E8 B96C0000

894424 38

C70424 87A17C00

E8 DEF8FFFF

C70424 00000000

A1 48907C00

FFD0

8B5424 3C

894424 08

C74424 04 7E000000

891424

E8 536C0000

8B4424 3C

894424 04

C70424 A2A17C00

E8 A4F8FFFF

C74424 48 00000000

EB 05

834424 48 01

8B4424 3C

890424

E8 7C6C0000

394424 48

73 33

8B5424 3C

call <JMP.&malloc>

mov dword ptr ss:[esp+3C],eax

mov dword ptr ss:[esp],81

call <JMP.&malloc>

mov dword ptr ss:[esp+38],eax

mov dword ptr ss:[esp],file.7CA187

call file.7C1510

mov dword ptr ss:[esp],0

mov eax,dword ptr ds:[7C9048]

call eax

mov edx,dword ptr ss:[esp+3C]

mov dword ptr ss:[esp+8],eax

mov dword ptr ss:[esp+4],7E

mov dword ptr ss:[esp],edx

call <JMP.&fgets>

mov eax,dword ptr ss:[esp+3C]

mov dword ptr ss:[esp+4],eax

mov dword ptr ss:[esp],file.7CA1A2

call file.7C1510

mov dword ptr ss:[esp+48],0

jmp file.7C1C88

add dword ptr ss:[esp+48],1

mov eax,dword ptr ss:[esp+3C]

mov dword ptr ss:[esp],eax

call <JMP.&strlen>

cmp dword ptr ss:[esp+48],eax

jae file.7C1CC0

mov edx,dword ptr ss:[esp+3C]

dword ptr ss:[esp+3C]=[00FFFB5C]=0

eax=01890E20

.text:007C1C1F file.exe:\$1C1F #101F

Dump 1

Dump 2

Dump 3

Dump 4

Dump 5

Watch 1

[x=] Lo

Address	Hex	ASCII
01890E20	0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA
01890E30	0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA
01890E40	0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA
01890E50	0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA
01890E60	0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA
01890E70	0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA

00FFFB20

00FFFB24

00FFFB28

00FFFB2C

00FFFB30

00FFFB34

00FFFB38

00FFFB3C

00FFFB40

00FFFB44

00000081
00000002
00FFFC24
753FCC0C
FFFFFFFF
6548FFFE
206F6C6C
6C726F77
44000A64
6C696E61

msv

Рисунок 3 – Memory dump

2.2 Обработка команды в fgets

В консоль введём слово open для открытия файла. На рисунке 4 видно, что введённое значение было записано в ячейку памяти.

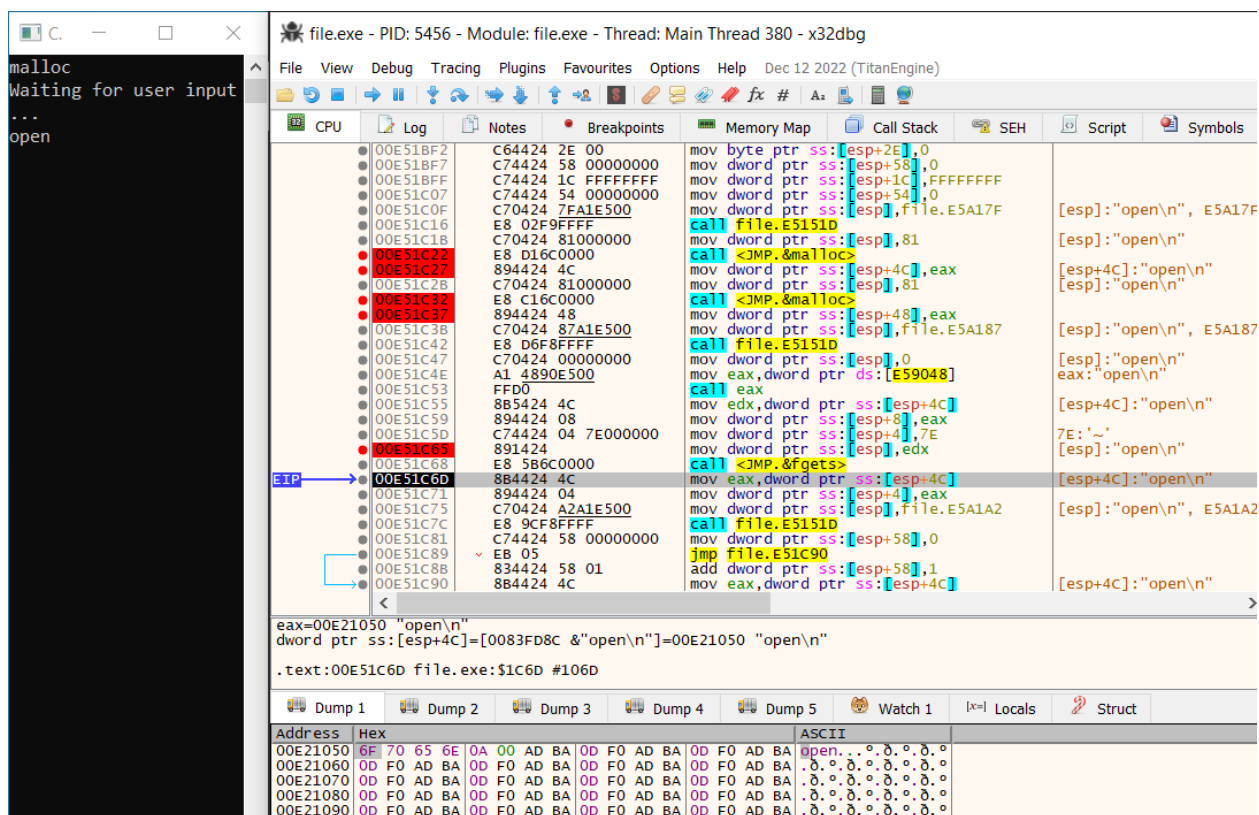


Рисунок 4 – Содержимое памяти

2.3 Вызов API-функции CreateFile

Далее вызывается функция `CreateFileA`. На рисунке 5 приведён нужный фрагмент библиотеки `kernel32.dll`.

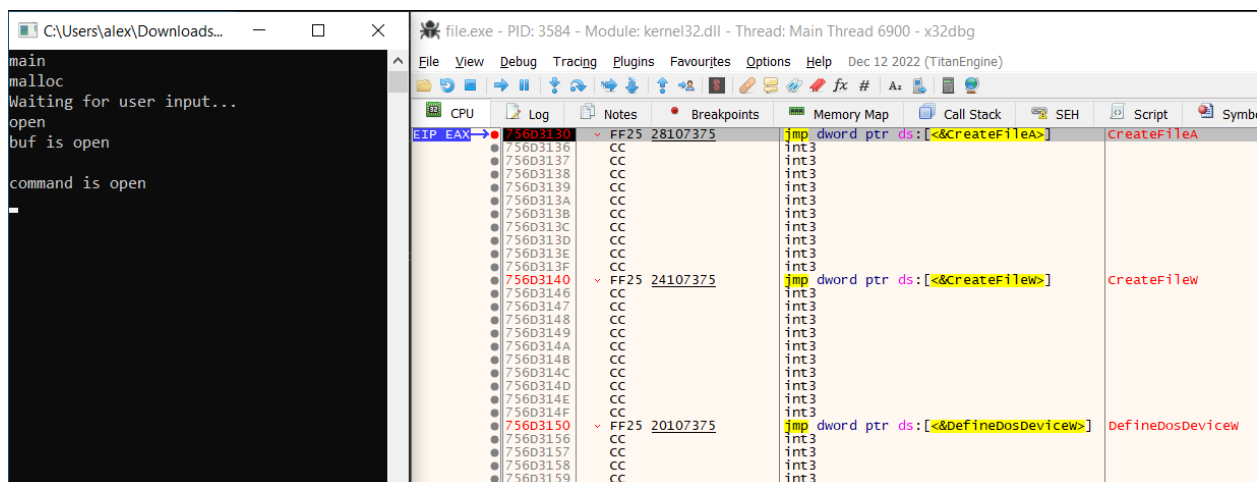


Рисунок 5 – Остановка на функции `CreateFileA`

На рисунке 6 видно, что в стеке сохранены аргументы функции, например,

название файла (AlekseyShadrinov.txt).

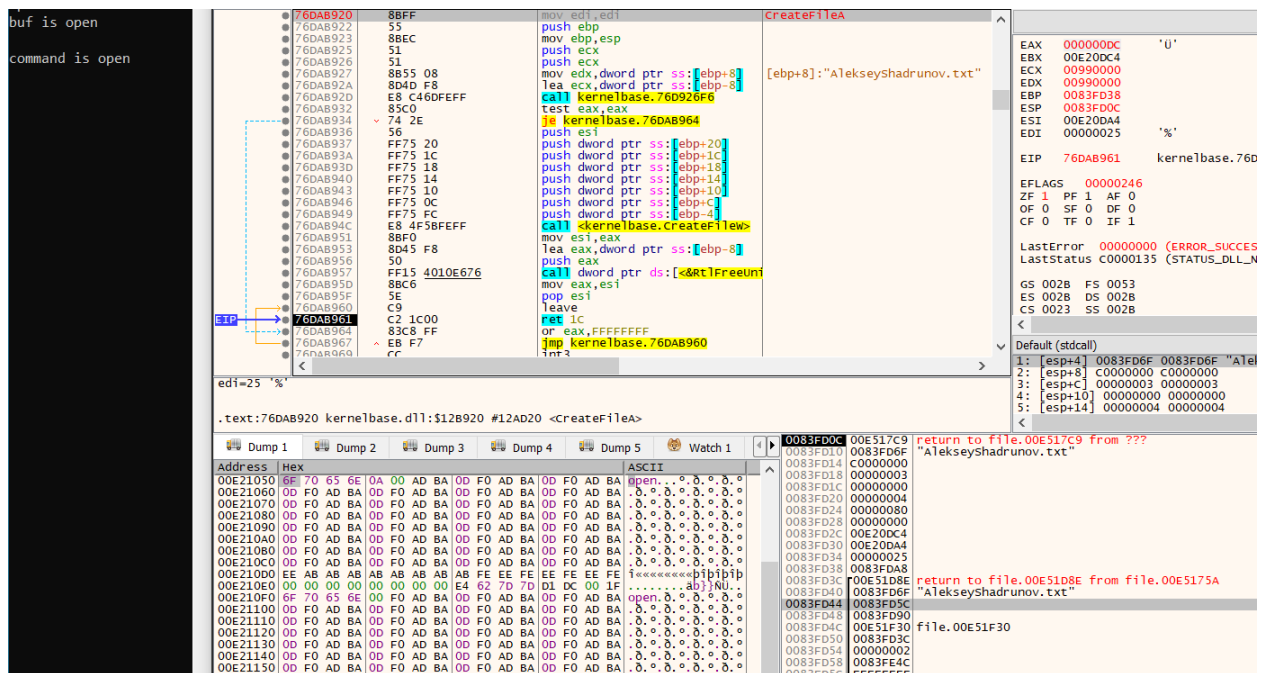


Рисунок 6 – Стек

После вызова API-функции CreateFileA можно увидеть, что в EAX записано значение 0xE0. Это файловый дескриптор или хэндл открытого файла (Рисунок 7).

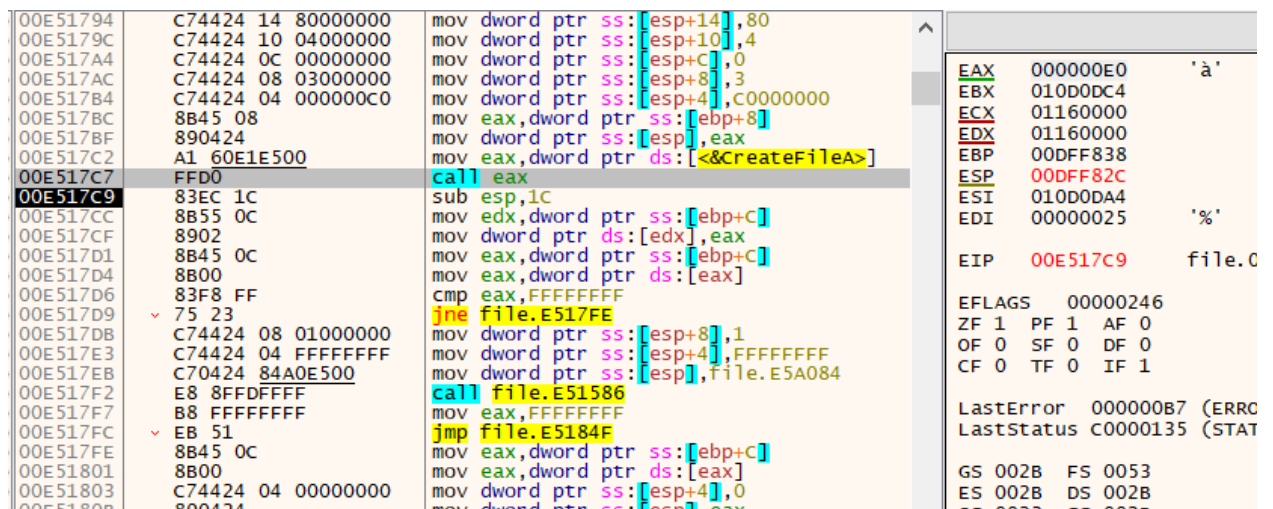


Рисунок 7 – Хэндл открытого файла

Этот же хэндл можно пронаблюдать в Process Hacker (Рисунок 8).

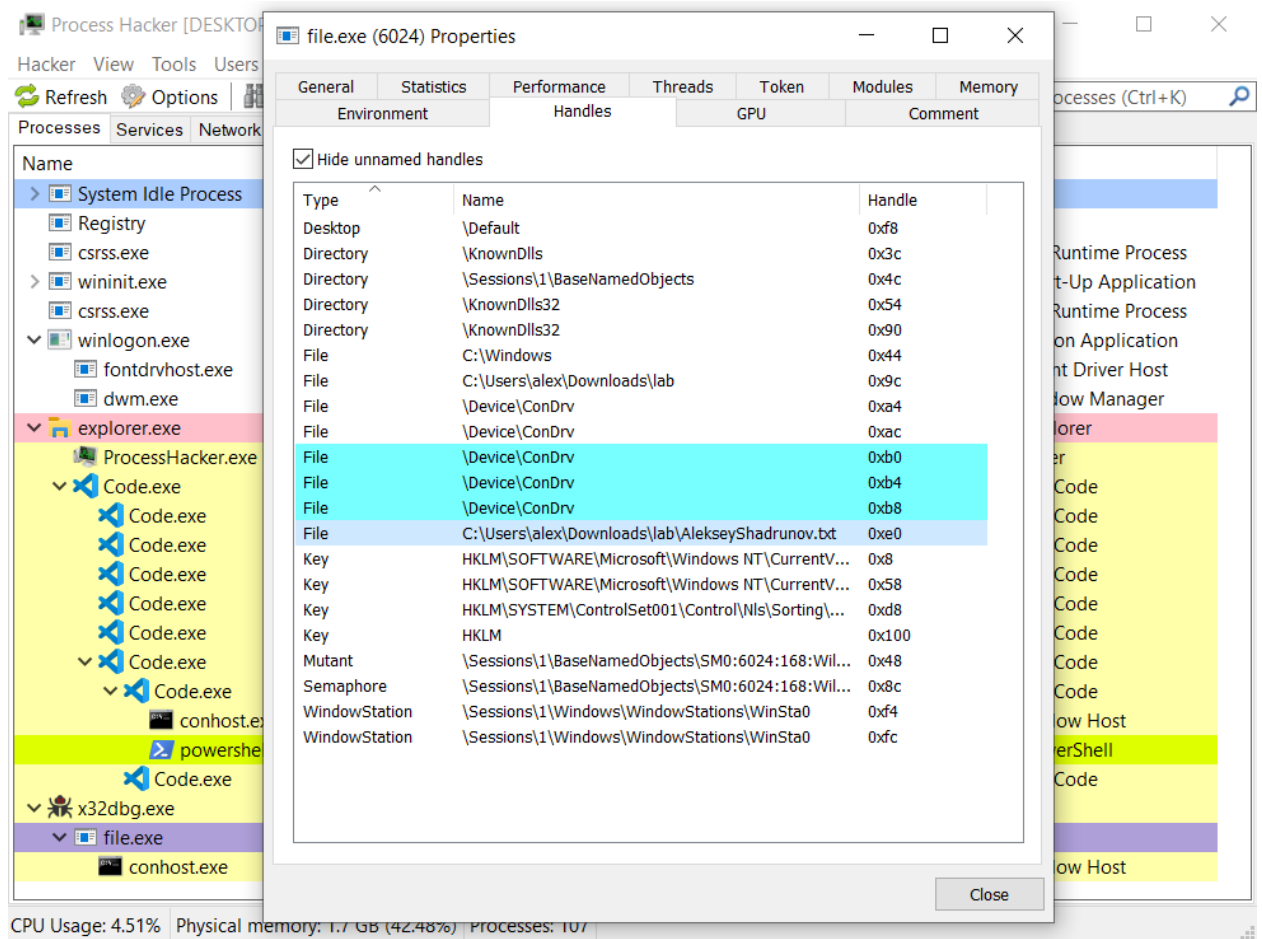


Рисунок 8 – Хэндл в Process Hacker

2.4 Вызов API-функции WriteFile

Вводим в консоль команду write для записи строки в файл. На рисунке 9 показано состояние системы перед вызовом функции WriteFile. Все аргументы записаны в стек: хэндл файла (104), буфер со строкой, количество байт для записи (0xC, то есть 13) и ещё два аргумента.

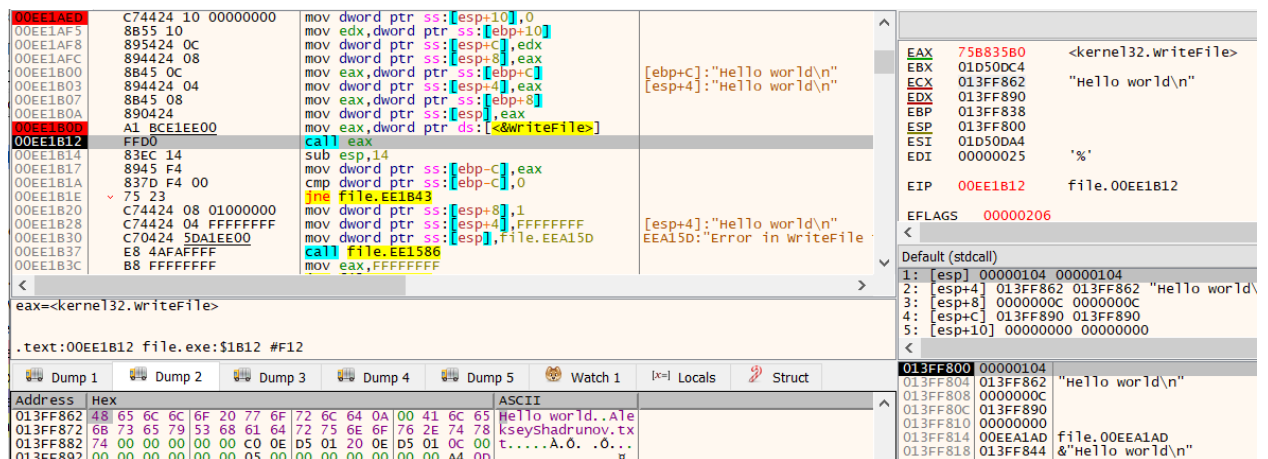


Рисунок 9 – Аргументы в стеке

На рисунке 10 видно тело функции WriteFile, а также строка для записи, которая находится в дампе памяти. На рисунке 11 — Возвращаемое значение 1 (в регистре EAX), что соответствует успешному выполнению функции.

766507CF	CC	int3			
766507D0	6A 18	push 18	writeFile		
766507D2	68 20A67076	push kernelbase.7670A620			
766507D7	E8 E48F0200	call kernelbase.766797C0			
766507DC	33C9	xor ecx,ecx	ecx:"Hello world\n"		
766507DE	894D E0	mov dword ptr ss:[ebp-20],ecx	[ebp-20]:&"Hello world\n"		
766507E1	894D E4	mov dword ptr ss:[ebp-1C],ecx			
766507E4	8B75 14	mov esi,dword ptr ss:[ebp+14]			
766507E7	85F6	test esi,esi			
766507E9	74 02	je kernelbase.766507ED	ecx:"Hello world\n"		
766507EB	890E	mov dword ptr ds:[esi],ecx			
766507ED	8B7D 08	mov edi,dword ptr ss:[ebp+8]			
766507F0	83FF F4	cmp edi,FFFFFFFF			
766507F3	0F84 0F540300	je kernelbase.76685C08			
766507F9	83FF F5	cmp edi,FFFFFFF5			
766507FC	0F84 F5530300	je kernelbase.76685BF7			
76650802	83FF F6	cmp edi,FFFFFFF6			
76650805	0F84 0B530300	je kernelbase.76685BE6			
7665080B	8B5D 18	mov ebx,dword ptr ss:[ebp+18]			

EAX	75B835B0	<kernel32.writeFile>
EBX	00F40DC4	
ECX	013FF702	"Hello world\n"
EDX	013FF730	
EBP	013FF6D8	
ESP	013FF698	
ESI	00F40DA4	'%'
EDI	00000025	
EIP	766507D2	kernelbase.766507D2
EFLAGS	00000206	

1:	[esp+4]	00EE1B14	file.00EE1B14
2:	[esp+8]	00000108	00000108
3:	[esp+C]	013FF702	013FF702 "Hello world\n"
4:	[esp+10]	0000000C	0000000C
5:	[esp+14]	013FF730	013FF730

Address	Hex	ASCII
013FF702	48 65 6C 6C 6F 20 77 6F 72 6C 64 0A 00 41 6C 65	Hello world..Ale
013FF712	68 73 65 79 53 68 61 64 72 75 6E 6F 76 2E 74 78	kseyshadrnunov.tx
013FF722	74 00 00 00 00 00 C0 0E F4 00 20 0E F4 00 0C 00	t....A.o..o...
013FF732	00 00 00 00 00 00 05 00 00 00 00 00 00 00 A4 0DM.
013FF742	F4 00 C4 0D F4 00 88 F7 3F 01 F0 12 EE 00 01 00	o.A.o..p.d.i...

Рисунок 10 – Дамп памяти со строкой

00EE1AF5	8B55 10	mov edx,dword ptr ss:[ebp+10]			
00EE1AF8	895424 0C	mov dword ptr ss:[esp+C],edx			
00EE1AFC	894424 08	mov dword ptr ss:[esp+8],eax			
00EE1B00	8B45 0C	mov eax,dword ptr ss:[ebp+C]			
00EE1B03	894424 04	mov dword ptr ss:[esp+4],eax			
00EE1B07	8B45 08	mov eax,dword ptr ss:[ebp+8]			
00EE1B0A	890424	mov dword ptr ss:[esp],eax			
00EE1B0D	A1 BCE1EE00	mov eax,dword ptr ds:[<writeFile>]			
00EE1B12	FFD0	call eax			
00EE1B14	83EC 14	sub esp,14			
00EE1B17	8945 F4	mov dword ptr ss:[ebp-C],eax			
00EE1B1A	837D F4 00	cmp dword ptr ss:[ebp-C],0			
00EE1B1E	75 23	jne file.EE1B43			
00EE1B20	C74424 08 01000000	mov dword ptr ss:[esp+8],1			
00EE1B28	C74424 04 FFFFFFFF	mov dword ptr ss:[esp+4],FFFFFFFF			
00EE1B30	C70424 5DA1EE00	mov dword ptr ss:[esp],file.EEA15D			
00EE1B37	E8 4AFAFFFF	call file.EE1586			
00EE1B3C	B8 FFFFFFFF	mov eax,FFFFFFFF			
00EE1B41	EB 05	jmp file.EE1B48			

EAX	00000001
EBX	00BF0DC4
ECX	7ED967D5
EDX	009FF928
EBP	009FF988
ESP	009FF964
ESI	00BF0DA4
EDI	00000025
EIP	00EE1B14
EFLAGS	00000200

1:	[esp+4]	009FF964	file.009FF964
----	---------	----------	---------------

Рисунок 11 – Возвращаемое значение 1

2.5 Вызов API-функции ReadFile

Далее считываем из файла с помощью функции ReadFile. Аргументы функции: хэндл (0x100), адрес выходного буфера (0xDC0F60), количество байт для чтения (0xC) и ещё два опциональных аргумента (Рисунок 12). Аргументы записываются в стек перед вызовом функции.

После вызова функции по адресу выходного буфера содержится строка из файла (Рисунок 13)

3 Выводы о проделанной работе

В ходе работы я реализовал программу, подсчитывающую сумму остатков от деления на 3 элементов массива длиной 10. Программа написана на ассемблере. В программе используется синтаксис Intel, а также реализован вывод в консоль без подключения библиотек.

Приложение А