ЛАБОРАТОРНАЯ РАБОТА № 2

"Методика изучения программ. Отладчики и дизассемблеры."

Цель работы. Изучение и практика методики исследования программ.Исследовать работу программы шифрования и создать собственный расшифровщик. Исследование методов защиты программ от изучения.

Краткие теоретические сведения

При исполнении программ на процессорах, команды представляются в памяти в виде чисел – машинных кодов. Можно создавать программы прямо в машинных кодах, однако более удобно использовать их символьное представление – язык Ассемблер. В настоящее время для этих целей используются языки высокого уровня: C, Pascal, C++, Delphi и др., которые стали намного ближе к естественному языку, чем Ассемблер и тем более, машинные коды. Процесс перевода исходных текстов в машинный код называется трансляцией (рис. 2.2)

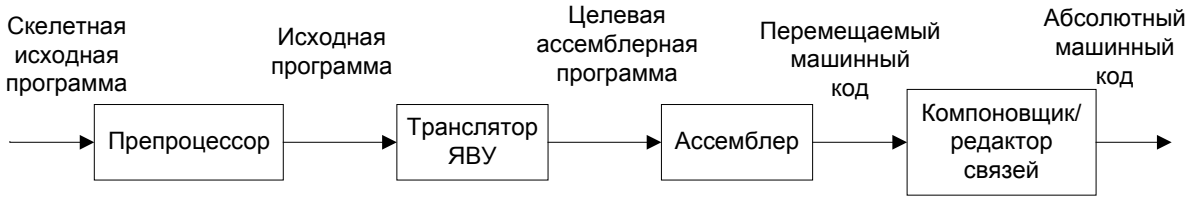

Рис 2.2 Процесс трансляции с языка высокого уровня

Процесс компиляции программы, созданной на языке Ассемблера состоит из меньшего числа промежуточных ступеней (рис 2.3).

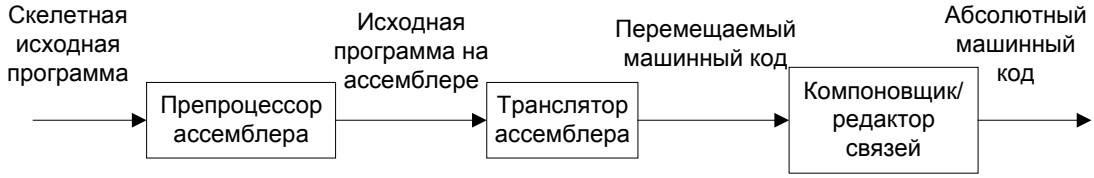

Рис 2.3 Процесс трансляции с языка Ассемблер

К исследователям программы попадают в виде бинарных исполняемых файлов, содержащих машинный код. Ставится задача восстановления алгоритма работы вируса без его исходного текста, при наличии только исполняемого файла, и есть три подхода к ее решению [71]:

1. Метод экспериментов.
2. Статический метод.
3. Динамический метод

Метод экспериментов заключается в проведении многократных экспериментов с изучаемой программой и сравнительном анализе полученных результатов. В случае вредоносного ПО данный метод может привести к деструктивным действиям.

Статический метод заключается в восстановлении алгоритма работы программы посредством анализа ее бинарного образа. Для удобства анализа машинный код переводится в команды на языке Ассемблера. Данный процесс называется синтаксическим дизассемблированием.

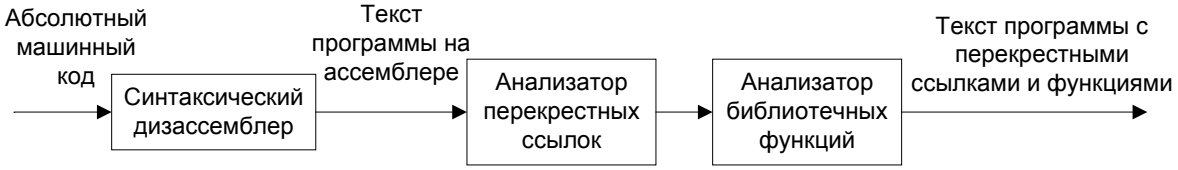

Рис 2.4 Процесс дизассемблирования контекстным дизассемблером

Контекстный дизассемблер (рис. 2.4) помимо синтаксического дизассемблирования получает перекрестные ссылки в программе, выделяет внутренние процедуры программы и имена библиотечных функций используемых ей. Процесс создания программы из исходного текста является процессом с потерями, так как теряется вся информация о названиях переменных, границах инструкций и о типах переменных для языков высокого уровня. Статический метод позволяет основательно изучить полученный текст программы и сделать в нем комментарии. Недостатком статического метода являются трудности при отслеживании содержимого регистров и различных областей памяти и переменных.

Динамический метод исследования - это изучение работы программы в отладчике. Микропроцессоры семейства Intel и совместимые содержат в себе специальные аппаратные средства для отладки программ. Данные средства позволяют останавливать программы в любом месте при наступлении определенных событий: обращение к определенному адресу оперативной памяти, выполнение команды по определенному адресу, обращение к портам ввода-вывода и др. При останове управление получает специальная программ – отладчик, которая позволяет просматривать значения регистров, стека и ячеек памяти исследуемой программы. Отладочные средства позволяют отслеживать работу программы в динамике, а именно, как программа меняет регистры, стек, содержимое памяти и обращается к портам и прерываниям.

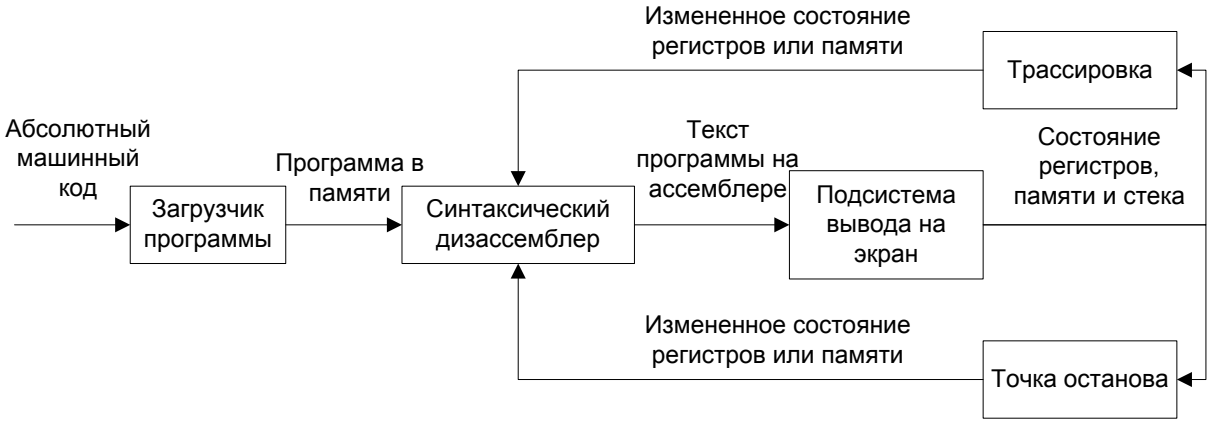

Рис 2.5 Процесс отладки программы

Выделяют два способа отладки. Трассировка – это покомандное выполнение программы, при котором есть возможность наблюдать за состоянием

процессора и памяти после выполнения каждой команды. Точка останова позволяет прервать программу в любом месте ее выполнения, при этом может выполниться от одной команды и более. Недостаток динамического метода в том, что вирус может выполнить все свои действия по заражению или деструкции на ЭВМ.

Дизассемблер Hview находится в исполняемом файле hiew.exe или hview.exe.

Запуск в командной строке: hiew.exe имя исполняемого файла

F1	Информация об исполняемом файле
F4	Режим: Text, Hex, Decode (дизассемблер)
F7	Поиск строки
F8	Выбор кодировки символов
F10	Выход из программы
F5	Перейти по адресу в режиме Decode
F3	Редактировать (Edit) программу в Decode
F2	Добавить команду ассемблера в Decode-Edit
F9	Принять изменения и выйти из Decode-Edit
ESC	Выйти из Decode-Edit, отменив изменения

Отладчик Turbo Debugger входит в состав пакета TASM 5. Исполняемый файл td.exe находится в каталоге bin пакета TASM 5. Имеет меню с выбором действий. Программу для отладки можно выбирать после запуска программы или указывать в командной строке

Запуск в командной строке: td.exe имя исполняемого файла

F2	Установить точку останова
F4	Выполнить до курсора
F7	Исполнять по командно с входом в процедуры
F8	Исполнять по командно без входа в процедуры
F9	Выполнить программу

Дизассемблер IDA PRO имеет для разных операционных систем разные исполняемые файлы: idax.exe для MS DOS, idaw.exe для Windows, idag.exe - графический оконный интерфейс. Имеет меню с выбором действий. Исполняемый файл для исследования выбирается с помощью стандартного диалога открытия файла.

Дизассемблер IDA PRO.

G	Прыжок по адресу
C	Преобразовать в код
D	Преобразовать в данные
A	Преобразовать в текст
U	Отменить любое преобразование
N	Переименовать метку
O	Операнд - смещение
*	Задать массив

;	Задать комментарий к строке
ENTER	Прыжок на метку под курсором
ESC	Возврат на один шаг назад по прыжкам
CTRL+E	Прыжок на точку входа в программу
CTRL+X	Показать ссылки на данную строку
CTRL+P	Прыжок на метку по имени
CTRL+U	Прыжок на начало функции

Одним из самых распространенных способов защиты от дизассемблирования является шифровка участков кода программы. Одним из самых простых методов является статическая шифровка, которая часто встречается в конвертных защитах. Суть в следующем. Создается шифрующая программа, которая перед сборкой загрузочного файла шифрует часть программы. В загрузочных файл добавляется фрагмент, который с фиксированных адресов расшифровывает зашифрованный фрагмент. В результате во время работы программы необходимые действия программой выполняются, но во время изучения дизассемблером мы видим другой код. Такие программы также называют самомодифицирующимися. Для изучения такой программы можно использовать отладчик, но при этом трудно получить листинг программы. Другой выход - использовать дизассемблер IDA, в котором с помощью встроенных скриптов можно расшифровать зашифрованный участок.

crypt.com - статическая шифровка, часто использующаяся в "конвертных защитах".

Далее приведены исходные коды для компонент защиты, все они компилируются в COM программы и для экономии места первые три и последняя строки опущены

Первая программа Lab21.asm является как бы префиксом защищаемой программы, в ее задачи входит расшифровка в памяти защищаемой программы.

```
start:                  ;
mov si,offset _end      ;в SI смещение метки _end, ее
                        адрес
lodsw                   ;в AX слово по адресу DS:SI,
                        SI=SI+2
xchg ax,cx              ;помещаем в CX значение AX
push si                 ;сохраняем в стек значение SI
decrypt:                ;метка decrypt
xor byte ptr            ;сложение по модулю 2
[si],0AAh
inc si                  ;увеличиваем SI на 1
loop decrypt            ;цикл с метки decrypt CX раз
jmp si                  ;прыгаем по адресу SI, в lab23
_end:                   ;метка _end
filesize dw 0           ;размер шифруемого файла
```

Программа Lab22.asm – защищаемая программа, вместо нее может быть подставлена любая другая

```
start:
mov ah,9
mov dx,offset text
int 21h
ret
text db
'Hello',0dh,0ah,'$'
_end:
```

Программа Lab23.asm это заключительная часть защиты, в ее функции входит копирование расшифрованной программы в начало сегмента и передача ей управления

```
start:
call _next
_next:
pop cx
pop si
mov di,offset start
push di
sub cx,si
rep movsb
ret
```

Lab24.asm – это программа, которая выполняет сборку в памяти всех фрагментов, шифрование защищаемой программы и запись всего этого обратно в файл.

```
start:
mov ax,3d02h
mov dx,offset file1
int 21h
xchg ax,bx
mov ah,3Fh
mov cx,100h
mov dx,pointer
int 21h
add pointer,ax
mov ax,3d02h
mov dx,offset filename
int 21h
xchg ax,bx
push bx
mov ah,3Fh
mov cx,1000h
mov dx,pointer
int 21h
mov di,pointer
mov [di-2],ax
```

```
add pointer,ax
xchg ax,cx
mov si,dx
crypt:
xor byte ptr [si],0AAh
inc si
loop crypt
mov ax,3d02h
mov dx,offset file2
int 21h
xchg ax,bx
mov ah,3Fh
mov cx,100h
mov dx,pointer
int 21h
add pointer,ax
mov ax,4200h
pop bx
xor cx,cx
xor dx,dx
int 21h
mov ah,40h
mov dx,offset buffer
mov cx,pointer
sub cx,dx
int 21h
ret
filename db 'lab22.com',0
file1 db 'lab21.com',0
file2 db 'lab23.com',0
pointer dw offset buffer
buffer:
```

После компиляции данных программ четвертая производит операции по сборке непосредственно самой защиты.

Расшифровывается 3-мя строками в IDA: используется 2 функции (одна чтение, другая записи).

1. A = Byte ([0x1000],[0x100]). – в ‘A’ записывается байт из сегмента 1000 и смещением 100h.
2. PatchByte ([0x1000, 0x100], 0x27) - запись в байт по 1000: 100h значение 27h

Сам скрипт производящий расшифрование:

```
auto a:
For (a = 0x100; a<0x120; a + +)
PatchByte ([0x1000, a], Byte([0x1000, a])^0x75)
```

Порядок выполнения работы

1. Наберите примеры и скомпилируйте их.
2. Исследуйте примеры дизассемблером Hview и IDA Pro.
3. Запустите lab24, который соберет защиту, исследуйте его работу под отладчиком Turbo Debugger, а также работу защищенной программы.
4. Исследуйте защищенную программу в Hview и IDA Pro, запустите типовой скрипт и расшифруйте фрагмент.
5. Добавьте в программу lab24 механизм выбора файла для шифрования (lab25), измените ключ шифрования и зашифруйте с ее помощью пример lab14.com, исследуйте его в IDA Pro и расшифруйте фрагмент.

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. Какое шифрующее преобразование используется в лабораторной?
2. Какая последовательность работы с файлами.
3. Какие функции работают с файлами.
4. Какие команды можно применять в шифровании?
5. Объясните работу предложенного фрагмента из лабораторной.