Лабораторная работа №3.

Изучение методов применения пакетных файлов для устранения последствий инфицирования носителей информации.

Цель работы: сформировать практические навыки поиска, устранения и автоматизации чистки последствий работы вируса.

Задачи: изучить принцип работы вируса семейства Win32/AutoRun.PSW.Delf.I (по сигнатурам ESET). Изучить способы нейтрализации последствий работы вируса. Автоматизировать процесс устранения вируса и его последствий.

Введение

Вирусы семейства PSW.Delf.I (по сигнатурам ESET) и их модификации получили широкое распространение по всему миру в 2010 году, и их периодические всплески тревожат пользователей и в настоящее время. Такое тотальное заражение получилось вследствие их весьма простого, но в то же время назойливого способа размножения. Вирус, находясь в ОП, отслеживал любое монтируемое устройство в системе и незамедлительно копировал себя в корень монтируемого устройства, создавая *autorun.inf* файл с запуском самого себя. Кроме автозапуска, он также подменял все существующие на монтируемом устройстве директории на ярлыки (*.lnk) с указанием на запуск себя, кроме этого создавал копию своего исполняемого файла (*.exe) в корне монтируемого устройства. Все ярлыки имели названия, идентичные названиям корневых директорий на монтируемом устройстве. Самим директориям присваивались атрибуты “скрытый” и “системный”, что позволяло скрыть их при стандартных настройках операционной системы.

Антивирусные компании добавили сигнатуры вируса в свои базы, и сегодня практически каждый антивирус способен идентифицировать это семейство вирусов и удалить исполняемые файлы с монтируемого устройства. Однако не решена проблема дальнейшего лечения, а именно: восстановление отображения директорий.

Данная лабораторная работа будет посвящена изучению способов автоматического устранения последствий работы вируса семейства Win32/AutoRun.PSW.Delf.I.

Ход работы: необходимо написать программу для автоматического изменения атрибутов директорий зараженного внешнего носителя.

Всего в современных операционных системах корпорации Microsoft чуть меньше 80 команд. Несколько команд, полезных для автоматизации обработки файлов. Детали вы всегда сможете уточнить по команде **help** или в справочнике.

set — установка переменной;
del — удаление одного или нескольких файлов;

for — оператор цикла;
echo — оператор вывода сообщения;
dir — вывод каталогов в директории;
attrib — изменение атрибутов;

Текущий каталог. Абсолютные и относительные пути

При работе с файловыми командами исключительную важность приобретает понятие текущего каталога. Дело в том, что при указании файла в качестве параметра команды мы всегда используем один из двух возможных способов указания на них: либо абсолютный путь, либо относительный путь. В полном пути мы указываем все, начиная с диска (или сетевого имени компьютера), например **d:\misha\box\beer.txt**. Какой бы каталог ни оказался текущим в момент ввода команды, полный путь будет соответствовать одному и тому же файлу. Для относительного пути текущий каталог служит отправной точкой. Простейший случай относительного пути — имя файла. В контексте выполнения команды оно означает файл с таким именем, расположенный в текущем каталоге.
Для записи относительного пути к текущему каталогу существует условная запись **.** (точка). Для записи относительного пути к каталогу, в котором содержится текущий каталог, существует условная запись **..** (две точки). Команда, показанная на следующем листинге, копирует все файлы из текущего каталога в каталог **neighbour**, расположенный рядом с ним.

```
copy *.* .\neighbour
```

Комментирование командного файла и его выдачи. Команды echo и rem

Командный файл, по существу, представляет собой программу, написанную на языке командного процессора операционной системы. Текст программы полагается снабжать комментариями, чтобы, вернувшись к нему некоторое время спустя, не вспоминать мучительно, для чего эта программа нужна, и как она устроена.
В системе команд MS-DOS для оформления комментариев предусмотрена команда **rem**. Это фиктивная команда, которая не предполагает выполнения каких бы то ни было действий, но позволяет написать в строке после своего имени произвольный текст. Причем командный процессор не воспринимает его как синтаксическую ошибку. Пример оформления командного файла комментариями показан на следующем листинге.

```
rem **************************************************
rem Формирование файлов справки по командам copy и move
rem **************************************************

rem Формируем файлы справки
help copy > copy.help
```

```
help move > move.help

rem Создаем каталог для хранения файлов справки
md msdos-help

rem Перемещаем файлы справки в подготовленный каталог
move *.help msdos-help
```

При выполнении приведенного выше командного файла все команды будут выводиться на экран по мере их выполнения, что не всегда удобно. Выдачу команд можно отключить с помощью команды **@echo off**. Символ «собака» перед командой **echo** означает, что и сама эта команда должна выполняться в «молчаливом» режиме. С таким же успехом мы могли бы не пользоваться командной **echo off**, а поместить «собаку» перед каждой командой.

Во многих случаях требуется, чтобы командный файл выводил на экран (или в файл) те или иные сообщения. В одних случаях это могут быть сообщения об ошибках, в других информационные сообщения, объясняющие пользователю командного файла, что происходит в данный момент. Для вывода сообщений применяется та же самая команда echo. В качестве параметра ей передают текст выводимого сообщения. Листинг усовершенствованного командного файла приведен ниже.

```
@echo off

rem **************************************************
rem Формирование файлов справки по командам copy и move
rem **************************************************

@echo Формируем файлы справки. Одну секундочку...

rem Формируем файлы справки
help copy > copy.help
help move > move.help
rem Создаем каталог для хранения файлов справки
md msdos-help
rem Перемещаем файлы справки в подготовленный каталог
move *.help msdos-help
echo Готово!
```

Переменные. Команда set

Переменной называется поименованное значение. В учебниках по программированию переменную обычно сравнивают с конвертом, на котором написано имя. Внутрь конверта можно положить нечто, например,

определенную сумму денег — это ее значение. Как и в случае с конвертом, значение переменной можно изменить.
Для объявления переменной и одновременно для присвоения ей значения применяется команда **set**. Пример записи этой команды показан на следующем листинге.

```
rem Компилятор хелп-файлов в формате CHM
set help_compiler=c:\HTML Help Workshop\hcc.exe
```

Для извлечения значения переменной ее имя помещают между двумя знаками процента, как показано ниже.

```
rem Компилятор хелп-файлов в формате CHM
set help_compiler=c:\HTML Help Workshop\hcc.exe

rem Проект хелп-файла модуля "Склад"
set store_hpj=help\sources\store\store.hpj

rem Проект хелп-файла модуля "Продажи"
set sales_hpj=help\sources\sales\sales.hpj

rem Компилируем хелп-файлы
%help_compiler% %store_hpj%
%help_compiler% %sales_hpj%
```

При написании командных файлов часто применяют следующий прием: несколько значений переменных указывают рядом (или перемежая их какими-либо символами или строками), так, чтобы получить некоторое новое осмысленное значение. Пример приведен на следующем листинге.

```
rem Путь к компилятору хелп-файлов
set help_compiler="c:\Program Files\HTML Help Workshop\hhc.exe"

rem Путь к каталогу, в котором находятся проекты хелп-файлов
set project_path=e:\work\projects\help-projects

rem Вызываем компилятор для обработки конкретного проекта,
rem имя которого передаем в первом параметре
%help_compiler% %project_path%\%1.hpj
```

Массовая обработка файлов. Команда for

Команда for позволяет организовать выполнение повторяющихся однотипных действий. Можно использовать ее для того, чтобы вывести на экран числа от одного до десяти, как показано на следующем листинге.

for /l %%i in (1,1,10) do echo %%i

Переменная **i** является счетчиком цикла. В силу своеобразия синтаксиса команды **for**, имя счетчика цикла должно состоять из одной буквы. Причем, если мы пишем командный файл, то перед именем счетчика цикла надо поставить сдвоенный знак процента, если же мы просто набираем команду в командной строке, то одиночный.

Логика работы этой команды такова. После слова **in** указан диапазон изменения счетчика цикла. В данном варианте команды это тройка чисел: начальное значение счетчика, шаг счета, предельное значение счетчика. При выполнении команды командный процессор сначала присвоит переменной **i** значение **1**, а потом на каждом шаге цикла будет увеличивать его на **1**, пока оно не превысит **10**. Очевидно, таких шагов получится десять. Если бы в качестве шага счета мы указали число **2**, то цикл выполнился бы пять раз. На каждом шаге цикла выполняется тело цикла, написанное после слова **do**. В приведенном примере это команда echo, которая выводит на экран текущее значение счетчика цикла.

Изменение атрибутов

ATTRIB [+R | -R] [+A | -A] [+S | -S] [+H | -H] [+I | -I]
[диск:][путь][имя_файла] [/S [/D] [/L]]

\+ Устанавливает атрибут.
\- Снимает атрибут.
R Атрибут "Только чтение".
A Атрибут "Архивный".
S Атрибут "Системный".
H Атрибут "Скрытый".
I Атрибут "Неиндексированное содержимое".
X Атрибут файла "Без очистки".
V Атрибут целостности.
[диск:][путь][имя файла]
Указывает файл или набор файлов для обработки.
/S Обрабатывает файлы с указанными именами в текущем каталоге
и во всех его подкаталогах.
/D Обрабатывает файлы и каталоги.
/L Работает с атрибутами самой символьной ссылки,
а не ее целевого объекта.

Задание.

Разработать программу, исправляющую последствия работы вируса. Примерный алгоритм действия вируса:

1) копирование тела вируса в корень съемного диска,

2) создание на устройстве autorun.inf файла, в котором указан автозапуск вируса,

3) применение ко всем директориям в корне диска атрибутов скрытый и системный,

4) создание ярлыков (*.lnk) с названием аналогичных директорий и указывающих на запуск исполняемого файла вируса и открытие соответствующей папки.

Программа должна обрабатывать только корневые папки, на случай монтируемого устройства с большим количеством файлов. Кроме этого, пользователь должен видеть ход выполнения работы программы (например, список проанализированных папок, либо прогресс-бар, показывающий общий ход выполнения).

Отчет должен содержать:

1) Текстовое описание последовательности действий по удалению вируса
2) Скриншоты действий по удалению вируса из системы
3) Исходный код программы, нейтрализующей последствия работы вируса
4) Скриншоты работы программы