Лабораторная работа №1

Тема: Способы обнаружения вирусов

Цель работы: ознакомление с различными способами обнаружения компьютерных вирусов и иного вредоносного программного обеспечения действующих в операционной системе (ОС) Windows и их удаление.

Время выполнения – 2 часа.

1. Постановка задачи

Отработать вопросы, связанные с различными способами обнаружения вирусов:

1. Выявление действия вируса в ОС Windows с помощью анализа протекающих процессов.
2. Выявление действия вируса в ОС Windows с помощью анализа кода подозрительных программ.
3. Обнаружение вирусов с помощью антивирусных программ и их удаление.

2. Отработка вопросов лабораторной работы.

Выявление действия вируса в ОС Windows с помощью анализа протекающих процессов

Определение процесса вируса в ОС Windows

Действие вируса можно обнаружить путем анализа протекающих в компьютере процессов. Для его обнаружения можно воспользоваться средствами ОС, а именно программой "Диспетчер задач Windows" (рис. 1).

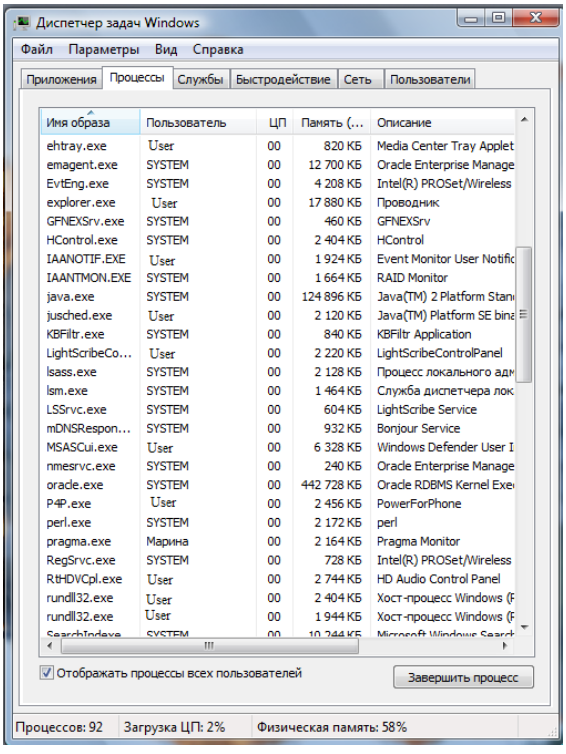

Рис. 1. – Пример окна «Диспетчера задач Windows»

В открытом окне «Диспетчера задач Windows» необходимо обратить внимание на процессы, запущенные от имени пользователя (User). Именно они могут в первую очередь являться подозреваемыми.

Задание 1. *Для выявления* подозрительных действий вредоносных программ необходимо выполнить и проанализировать следующую последовательность команд:

а) отключить все программы, которые стоят в "Автозагрузке" (*Программы-> Автозагрузка*)

б) перезапустить ПК;

в) с помощью «Диспетчера задач Windows» завершить «системные» процессы (запущенные от имени пользователя): например, такие как taskmgr.exe, ctfmon.exe, explorer.exe, wcmdmgr.exe;

г) из числа процессов «Диспетчера задач Windows», что остались, завершить работу подозреваемых процессов, запущенных от имени пользователя;

д) если эти действия не позволили выявить вредоносную программу, приступить к последовательному запуску и завершению работы всех приложений, имеющимся на компьютере. Появившийся и оставшийся процесс вероятно всего является вредоносной программой.

Задание 2. *Удаление вирусов.*

Если удалось определить имя процесса в памяти, принадлежащего вирусу, можно «исцелять» («лечить») компьютер.

Последовательность выполняемых действии:

- производится перезагрузка компьютера для освобождения оперативной памяти (ОП) от вируса;

- проверка наличия запуска подозрительного процесса в "Автозагрузке" всех пользователей. Также необходимо посмотреть наличие запуска в программе "Настройка системы" (исполнить команду msconfig – внешний вид появляющегося окна рис. 2).

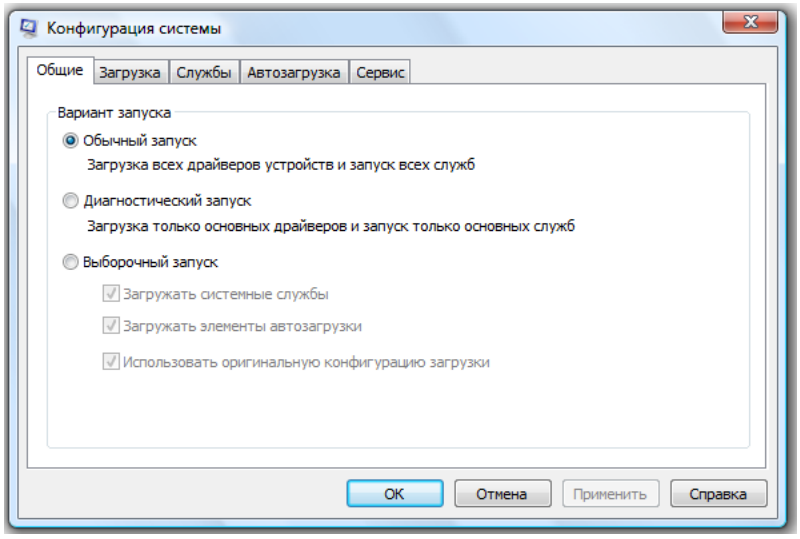

Рис. 2. – Окно программы настройки системы

Программа настройки системы является дополнительным средством для выявления проблем, которые могут помешать запуску Windows в обычном режиме. При запуске Windows можно отключить обычные службы и автоматически загружаемые программы, а затем включать их

по одной. Если проблема не возникает, когда служба отключена, но появляется после ее включения, значить эта служба может быть источником проблемы.
Программа настройки системы предназначена для поиска и изолирования неполадок, и не предусмотрена для управления загрузкой.

- производится поиск всех "измененных" EXE-файлы за последний месяц или лучше за два месяца с помощью программы поиска. Все найденное можно скопировать в один каталог. Дальше проверить программы на наличие в них вируса поможет файловый менеджер (FAR, Total Commander и т.д.). В коде программы ведется поиск строки

This program cannot be run in DOS mode или
This program must be run under Win32.

Необходимо отметить, что одна из этих строк обязательно присутствует в начале кода программы. Задача заключается в том, чтобы найти еще одну такую строку. Если строка найдена – значит, в этот файл внедрился вирус. Дальше нужно проверить все найденные файлы подобным способом. После того как вы узнали, какой из файлов заражен, они заменяются незараженными – выполняется процедура простого копирования файлов, если таковые имеются. Для этого, возможно, вам понадобится другой компьютер без вируса. Если такого компьютера нет, вам придется удалить зараженные файлы и переустановить те приложения (по необходимости и ОС), из которых вы их удалили. *Не делайте деинсталляцию программ, вы можете запустить вирус*!

Выявление действия вируса в ОС Windows с помощью анализа кода подозрительных программ

В настоящее время существуют типичные способы инфицирования исполняемых файлов и методы их выявления.

Анализ вирусного кода требует обширных знаний из различных областей программирования, а также специализированного инструментария.

Места наиболее вероятного внедрения вирусов

Объектом вирусного поражения могут выступать как исполняемые файлы (динамические библиотеки, компоненты ActiveX, плагины), так и драйвера, командные файлы операционной системы (bat, cmd), загрузочные сектора (MBR и BOOT), оперативная память, файлы сценариев (Visual Basic Script, Java Script), файлы документов (Microsoft Word, Microsoft Excel) и т.д.

Рассмотрим некоторые из вирусоносителей, а именно, исполняемые файлы.

Единственным гарантированным способом выяснения: является ли данный файл зараженным файлом или нет – является его полное дизассемблирование. Однако, дизассемблирование – крайне кропотливая работа и на глубокую реконструкцию программы размером в пять-десять мегабайт могут уйти годы**,** если не десятки человеко-лет! На первый взгляд, громаднейшие трудозатраты делают такой способ анализа чрезвычайно непривлекательным и бесперспективным. Однако, если отталкиваться от того, что подавляющее большинство вирусов и троянских программ имеют ряд характерных черт, отличающих их от всякой "нормальной" программы, то не все так плохо.

Количество всевозможных *признаков*, прямо или косвенно указывающих на зараженность файла, весьма велико, поэтому ниже перечислены лишь наиболее

характерные из них. Но даже они позволяют обнаружить до 4/5 всех существующих вирусов, а по некоторым оценкам и более того.

Текстовые строки

Прежде чем приступать к тотальному дизассемблированию исследуемого файла, нелишне просмотреть его дамп на предмет выявления потенциально небезопасных текстовых строк, к которым, в частности, относятся *команды SMTP-сервера и командного интерпретатора операционной системы* (*HELO/MAIL FROM/MAIL TO/RCPT TO, DEL/COPY/RD/RMDIR* соответственно), *ветви автозапуска реестра* (*RunServices, Run, RunOnce*), *агрессивные лозунги и высказывания ("дурак", "сам дурак"*) и т. д.

Большое количество вирусов обнаруживается таким элементарным способом.

Задание 3. Исследовать на наличие вируса следующий программный код. В качестве примера предлагается фрагмент известного вируса I-Worm.Kiliez.e.

```
aQuit           db 'QUIT',0Dh,0Ah,0
data:0040E050                   db '.',0Dh,0Ah,0
data:0040E058 aData             db 'DATA ',0Dh,0Ah,0
data:0040E060 aHeloS            db 'HELO %s',0Dh,0Ah,0
data:0040E06C asc_40            db '>',0Dh,0Ah,0
data:0040E070 aMailFrom         db 'MAIL FROM: <',0
data:0040E080 aRcptTo           db 'RCPT TO:<',0
data:0040F244 aSoftwareMicrosdb 'Software\Microsoft\Windows\CurrentVersion\',0
data:0040F279 aRun              db 'Run',0
data:0040F27D aRunonce          db 'RunOnce',0
data:0040F285 aSystemCurrentcdb 'System\CurrentControlSet\Services',0
data:0040F2A7 aSoftwareMicr_0db 'Software\Microsoft\WAB\WAB4\Wab File Name',0
data:0040F2D1 aRunservices      db 'RunServices',0
data:0040F2DD aInternetSettindb 'Internet Settings\Cache\Paths',0
data:0040F302 aHi               db 'Hi,',0
data:0040F306 aHello            db 'Hello,',0
data:0040F30D aRe               db 'Re:',0
data:0040F311 aFw               db 'Fw:',0
data:0040F315 aUndeliverableMdb 'Undeliverable mail--"%s"',0
data:0040F32E aReturnedMailS db 'Returned mail--"%s"',0
```

Вывод. В теле исследуемого программного кода содержатся текстовые строки, выдающие его агрессивные намерения.

Стартовый код.

В девяностых годах двадцатого века, когда вирусы создавались преимущество на ассемблере и писались преимущественно профессионалами, а коммерческие программисты в своем подавляющем большинстве полностью отказались от ассемблера и перешли на языки высокого уровня, для разработчиков антивирусов наступили «золотые» дни, ибо распознать зараженный файл зачастую удавалось с одного взгляда. Действительно, любая нормально откомпилированная программа начинается с так называемого стартового кода (Start-Up code), который легко отождествить визуально (как правило, он начинается с вызова функций GetVersion, GetModuleHandleA и т. д.), а дизассемблер IDA и вовсе идентифицирует стартовый код по обширной библиотеке сигнатур, выдавая тип и версию компилятора. Ассемблерные же программы стартового кода лишены и потому, когда ассемблерный вирус внедряется в программу, написанную на языке высокого уровня, стартовый код

отодвигается как бы "вглубь" файла, демаскируя тем самым факт своего заражения.

Сегодня, когда ассемблерные вирусы становятся редкостью, такой способ отождествления перестает работать, однако до полного списывания его со счетов еще далеко.

Строго говоря, никаких формальных признаков "нормального" start-up'-а не существует, поэтому всякий разработчик волен реализовывать его по своему. Однако редкий разработчик подключает к программе свой собственный start-up и все чаще для этих целей используется библиотечный стартовый код, поставляемый вместе с компилятором. Несмотря на то, что даже в рамках одного-единственного компилятора существует множество разновидностей стартового кода, все они легко узнаваемы и факт отсутствия стартового кода надежно обнаруживается даже самыми начинающими из исследователей.

Приблизительная структура типичного стартового кода, например, для Си++ программ такова: сначала идет пролог, затем настройка обработчика структурных исключений, обнаруживающая себя по обращению к сегментному регистру FS. Далее следует вызов функций GetVersion (GetVersionEx), GetModuleHandleA и GetStartupInfoA.

Подробно не останавливаясь на этом вопросе, сравним start-up код нормальной программы с кодом вируса Win2K.Inta.1676.

```
.text:00401670 startproc near
.text:00401670         push    ebp
.text:00401671         mov     ebp, esp
.text:00401673         push    0FFFFFFFFh
.text:00401675         push    offset stru_420218
.text:0040167A         push    offset __except_handler3
.text:0040167F         mov     eax, large fs:0
.text:00401685         push    eax
.text:00401686         mov     large fs:0, esp
.text:00401696         call    ds:GetVersion ; Get current version number of Windows
.text:004016EC         push    0
.text:004016EE         call    _heap_init
.text:00401704         mov     [ebp+var_4], 0
.text:0040170B         call    _ioinit
.text:00401710         call    ds:GetCommandLineA
.text:00401716         mov     dword_424F44, eax
.text:0040171B         call    _crtGetEnvironmentStringsA
.text:00401720         mov     dword_4235C0, eax
.text:00401725         call    _setargv
.text:0040172A         call    _setenvp
.text:0040172F         call    _cinit
.text:00401754         call    _main
.text:00401763         call    _exit
```

Так выглядит листинг нормальный start-up от MicrosoftVisualC++ 6.0.

```
.text:00011000 start  proc  near
.text:00011000        mov     eax, [esp+arg_0]
.text:00011004        lea     edx, loc_11129
.text:0001100A        mov     [eax+34h], edx
.text:0001100D        lea     edx, dword_117A0
```

```
.text:00011013      lea     eax, aHh          ;  "HH"
.text:00011019      mov     [edx+8], eax
.text:0001101C      mov     [eax+4], aSystemrootSyst
.text:0001101C                          ;  "\\SystemRoot\\system32\\drivers\\inf.sys"
.text:00011023      push    1200h
.text:00011028      push    0
.text:0001102D      call    ExAllocatePool
.text:00011032      or      eax, eax
.text:00011034      jnz     short loc_1103E
.text:00011036      mov     eax, 0C0000001h
.text:0001103B      retn    8
```

А так выглядят листинг окрестности точки входа вируса Win2K.Inta.1676.

Таким образом, в то время как обычная программа опрашивает текущую версию операционной системы, зловредный вирус сразу же пытается установить связь с драйвером inf.sys. Правильные программы так себя не ведут, и коварность вирусных планов разоблачается с первого взгляда!

Троянские программы, в большинстве своем написанные на языках высокого уровня, имеют вполне стандартный Start-Up и потому на такую уловку не обнаруживаются. Взять например того же Kilez'a, стартовый код которого выглядит так:

```
.text:00408458  start   proc  near
.text:00408458          push  ebp       ;   sub_408458
.text:00408459          mov   ebp, esp
.text:0040845B          push  0FFFFFFFFh
.text:0040845D          push  offset stru_40D240
.text:00408462          push  offset __except_handler3
.text:00408467          mov   eax, large fs:0
.text:0040846D          push  eax
.text:0040846E          mov   large fs:0, esp
.text:0040847B          mov   [ebp+var_18], esp
.text:0040847E          call  ds:GetVersion     ;   Get current version number
                              of Windows
.text:004084AF          xor   esi, esi
.text:004084B1          push  esi
.text:004084B2          call  _heap_init
.text:004084C4          mov   [ebp+var_4], esi
.text:004084C7          call  _ioinit
.text:004084CC          call  ds:GetCommandLineA
.text:004084D2          mov   dword_494E68, eax
.text:004084D7          call  _crtGetEnvironmentStringsA
.text:004084DC          mov   dword_493920, eax
.text:004084E1          call  _setargv
.text:004084E6          call  _setenvp
.text:004084EB          call  _cinit
.text:004084F0          mov   [ebp+StartupInfo.dwFlags], esi
.text:004084F3          lea   eax, [ebp+StartupInfo]
.text:004084F6          push  eax       ;   lpStartupInfo
.text:004084F7          call  ds:GetStartupInfoA
.text:004084FD          call  _wincmdln
```

Листинг червя I-Worm.Kilez.h имеет стандартный стартовый код.

Даже "невооруженным" глазом видно, что стартовый код червя идентичен стартовому коду Microsoft Visual C++ 6.0, что и не удивительно, так именно на нем червь и написан.

Точка входа

При внедрении вируса в файл точка входа в него неизбежно изменяется. Лишь немногие из вирусов ухитряются заразить файл, не прикасаясь к точке останова (вирус может вписать по адресу оригинальной точки останова JUMP на свое тело, слегка подправить таблицу перемещаемых элементов или вклиниться в массив RVA-адрес таблицы импорта, внедриться в незанятые области кодовой секции файла и т. д.).

И если "нормальные" точки входа практически всегда находятся в кодовой секции исполняемого файла (".text"), точнее – в области библиотечных функций, непосредственно предшествуя секции данных, то *точка входа зараженного файла традиционно располагается между секцией инициализированных и неинициализированных данных, практически у самого конца исполняемого файла*.

Так происходит потому, что, дописывая свое тело в конец файла, "вирусная" секция оказывается самой последней секцией инициализированных ячеек памяти, за которой простирается обширный регион неинициализированных данных, без которого не обходится ни одна программа. Это вирус и демаскирует.

Нестандартные секции

При заражении исполняемого файла методом дозаписи своего тела в его конец у вируса есть две альтернативы: либо увеличить размер последней секции файла (а это, как правило, секция ***.data***), присвоив ей атрибут Executable, либо создать свою собственную секцию. Оба этих способа легко распознаются визуально.

Код, расположенный в конце секции данных, – весьма характерный признак вируса, равно как и секция ***.text***, замыкающая собой файл и после передающая управление "вперед" – на нормальную точку входа. То же самое относиться и к секциям с нестандартными именами, зачастую совпадающими с именем самого вируса или маскирующимися под секции, создаваемые упаковщиками исполняемых файлов (но сам файл при этом остается неупакован!).

Достаточно часто вирусы внедряются в создаваемую ими секцию ***.reloc***, штатно предназначенную для хранения перемещаемых элементов. Поэтому, встретив в исследуемом файле секцию .reloc, содержащую исполняемый код, можно, с вероятностью близкой к единице, сделать вывод о заражении программы вирусом.

Таблица импорта

Операционные системы семейства Windows поддерживают два основных способа компоновки: статический и динамический. При статической компоновке имена вызываемых API-функций выносятся в специальную таблицу – таблицу импорта, изучение которой дает более или менее полное представление о природе исследуемой программы и круге ее интересов. К потенциально опасным функциям в первую очередь относятся сетевые

функции, функции поиска, вызова и удаления файлов, ***TOOLHELP***-функции, используемые для просмотра списка активных процессов и внедрения в них.

Зловредной программе ничего не стоит загрузить все эти функции и самостоятельно, путем *динамической компоновки*, в простейшем случае опирающейся на вызов *LoadLibrary/GetProcAddress*, или на "ручной" поиск API-функций в памяти (адрес системного обработчика структурных исключений дает нам адрес, принадлежащий модулю KERNEL32.DLL, базовый адрес которого определяется сканированием памяти на предмет выявления сигнатур "MZ" и "PE" с последующим разбором PE-заголовка). Но в этом случае текстовые строки с именами соответствующих функций должны так или иначе присутствовать в теле программы (если только они не зашифрованы).

Однако статистика показывает, что таблица импорта троянских программ носит резко полярный характер. Либо она вообще практически пуста, что крайне нетипично для нормальных, – неупакованных, – программ, либо содержит обращения к потенциально опасным функциям в явном виде. Конечно, сам факт наличия потенциально опасных функций еще не свидетельствует о троянской природе программы, но без особой нужны ее все-таки лучше не запускать.

Обнаружение вирусов в ИС с помощью антивирусных программ

Задание 4. Проверить содержимое компьютера (смартфона) на наличие вредоносных программ с помощью нескольких различных доступных антивирусных программ и обезвредить обнаруженные вредоносные программы.

Результаты работы

1. Продемонстрировать результаты проведенной работы в соответствии с заданием на лабораторную работу преподавателю и отобразить их в отчете.

2. ***Особое внимание уделить*** результатам работы каждой антивирусной программы (с подробным указанием установленных опций проверки антивирусной программы, количества проверенных папок, файлов и их размер, времени проверки и т.д.) и сравнительного анализа выявленных злонамеренных программ различными антивирусными программами, которые представить и объяснить в отчете.