**Лекция 15. AntiDOg Quest.**

**На прошлой лекции Вы по диагонали или по вертикали «прочитали» о том, что такое «AntiDHog68». Был записан алгоритм функционирования антивируса, определено, что будем являться Signature, приведено описание того, что такое макрокоманда, макроопределение, макрорасширение. В процессе написания антивируса был использован аппарат ПРОЦЕДУР (Kill the Bill). Исходя из предположения, что Лекция 14 Вами была якобы «прочитана» поверхностно и Вы поступили как «секретарки», просто просмотрев текст, что был Вам предложен, не озаботившись доскональным изучением всех деталей, сегодняшняя Лекция будет построена так:**

1. **Вы ещё раз изучаете текст антивируса, но вникая в детали, так как известно, что "Дьявол скрывается в мелочах».**
2. **В процессе изучения Вы должны будете ПИСЬМЕННО НА ЛИСТОЧКАХ ОТВЕЧАТЬ НА ВОПРОСЫ ПО ТЕКСТУ ЛЕКЦИИ (именно на занятии – в течение 17-30 – 18-50). НА ЛИСТОЧКАХ!!!! А в конце занятия ЛИСТОЧКИ ФОТОГРАФИРУЮТСЯ и присылаются мне на почту!!!!**
3. **А до конца ЭТОЙ недели необходимо ВЫПОЛНИТЬ ДОМАШНЕЕ ЗАДАНИЕ - ПРОДЕМОНСТРИРОВАТЬ ПРАВИЛЬНЫЕ ОТВЕТЫ, проверив свои, набрав необходимый текст на языке ассемблер, транслировать, линковать, выполнить, продемонстрировать результат в отчёте.**

**Итак, погнали….**

**MoveFPos macro F\_Handle, FPos**

**mov ax,4200h**

**mov bx,F\_Handle**

**xor cx,cx**

**mov dx,FPos**

**int 21h**

**endm**

**PutStr macro Text**

**mov ah,09h**

**mov dx,offset Text**

**int 21h**

**endm**

***Вопрос 1:***

* 1. ***Дать описание функции 9 21h прерывания на примере, в котором область памяти Text определена следующим образом:***

***Text db ‘Mama’, 10, ‘Mila’, 13, ‘ramu’, ‘10’,13,”$”***

* 1. **И ответить на вопрос: «Что будет выведено на экран»?**

**Как видим, макроОПРЕДЕЛЕНИЯ записываются до основного текста на языке записи алгоритмов ассемблер. Переходим к описанию 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**

**Вопрос 2. Какое макрорасширение будет сформировано ПРЕПРОЦЕССОРОМ по команде PutStr.**

**Далее установим адрес области памяти DTA, в которой, как мы знаем, обычно хранится информация о найденных файлах 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)**

**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 (DS):(SI), пересылает (КОПИРУЕТ) содержимое этого байта в байт, адрес которого определяется парой 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**

**Вопрос 3: Адрес области памяти FN\_Ofs определен следующим образом:**

***FN\_Ofs equ offset DTA+1eh. Что такое адрес, что такое equ и что означает offset.***

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

***Вопрос 4: Нарисовать в шестнадцатеричном виде содержимое регистра СХ после выполнения команды MOV CX, -128***

**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**

**Вопрос 5: Дать толкование понятия Handle и записать фрагмент программы на языке ассемблер ДЛЯ ВЫВОДА НА ЭКРАН имени файла и Handle файла, имя которого находится в FName.**

**Установить начальные значения для процедур:**

**mov FHandle,ax**

**mov ah,40h**

**mov cl,44h**

**nop**

**nop**

**Вопрос 6: Зачем здесь ЗАПИСАНЫ две команды Nop. Что будет, если их убрать?**

**Вопрос 7: Далее осуществляется вызов процедуры InfoAboutFile. Что происходит при вызове процедуры, как передаётся управление процедуре и что происходит со стеком. Готовя отчёт к концу недели, используя отладчик показать, что находится в стеке до вызова процедуры и после выхода из процедуры, определить адрес ТОЧКИ ВОЗВРАТА из этой процедуры в головную программу.**

**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**

**Вопрос 8 (так как на этот вопрос из прошлой Лекции праильно ответили пару человек): В чём разница между действиями: УБИТЬExecutable и УДАЛИТЬExecutable. Ответ типа: «Это соответствующий перевод слов Kill и Remove» - НЕ ПРИНИМАЕТСЯ…**

**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**

***Вопрос 9: Построить макрорасширение, соответствующее макрокоманде***

***MoveFPos FHandle 0eh***

**mov ah,3fh**

**mov bx,FHandle**

**mov cx,14**

**mov dx,offset SignatureArray**

**int 21h**

**cld**

**push ds**

**pop es**

***Вопрос 10: Почему и зачем ЗАПИСАНЫ команды Push и Pop.***

**mov si,offset VirSignature**

**mov di,offset SignatureArray**

**mov cx,14**

**repe cmpsb**

***Вопрос 11: Дать ПОЛНОЕ (желательно нарисовать) ОПИСАНИЕ того, ЧТО ПРОИСХОДИТ в результате repe cmpsb, дать определение Префикса Repe и КОМАНДЫ Cmpsb!!!! Не помоечно!!***

**jnz Finish**

**mov SignatureFound,1**

**Finish:**

**Ret**

**Вопрос 12: Нарисовать, что происходит ПО КОМАНДЕ Ret. В отчёте до конца недели ПОКАЗАТЬ, ЧТО НАХОДИТСЯ В СТЕКЕ ДО ВЫПОЛНЕНИЯ КОМАНДЫ RET и ПОСЛЕ ВЫПОЛНЕНИЯ КОМАНДЫ 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**

**Вопрос 13: Сколько байт , каких (в 16 с.с.) ЗАПИСЫВАЕТСЯ (посредством использования какой функции и В КАКОЙ ФАЙЛ) при вызове процедуры KillExecutable.**

**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**

**Вопрос 14: Дать подробное описание команд**

**Lodsb и 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 15: Какие действия осуществляются транслятором при трансляции ДИРЕКТИВЫ End start.**