Как ломать программы Windows (C) ED!SON [UCF], перевод Mr.Boco/TCP

СОДЕРЖАНИЕ

1. Введение в ломание Windows-программ

2. Обзор SoftICE/Win 2.oo

3. Поиск регистрационных кодов

3.1 Task Lock 3.00 - простая защита на основе серийного номера

3.2 Command Line 95 - простая регситрация "имя-код"

4. Создание генератора ключей для Command Line 95

5. Как работают инструкции PUSH и CALL когда программа вызывает функцию

6. О программах, написанных на Visual Basic

ПРИЛОЖЕНИЯ

A. Как в SoftICE загружать символьные имена (имена функций etc)

B. Синтаксис функций GetWindowText, GetDlgItemText и GetDlgItemInt

C. Где найти программы

D. Как связаться с автором

1. ВВЕДЕНИЕ В ЛОМАНИЕ WINDOWS-ПРОГРАММ

Ломать программы Windows в большинстве случаев даже проще, чем ломать программы Dos. В Windows сложно что-нибудь скрыть от того, кто ищет, особенно если программа использует стандартные функции Windows.

Первая (и часто единственная) вещь, которая Вам потребуется - это SoftICE/Win 2.oo, мощный отладчик от фирмы NuMega. Некоторым людям он кажется очень сложным в использовании, но я расскажу Вам, как с ним управляться и, я надеюсь, Вы поверите мне. =) В приложении A я привел некоторую информацию, которую Вам следует прочитать.

URL всех программ, которые Вам понадобятся, приведены в приложении C.

- ED!SON, edison@ccnux.utm.my

2. ОБЗОР SOFTICE/WIN 2.OO

|  |  |  |
| --- | --- | --- |
| Ниже | приведен очень | схематичный рисунок, демонстрирующий окно SoftICE: |
| |---  | | --------------  Регистры | -|  | 'R' - правка значения регистров |
| | | Окно данных | | 'D' - просмотр памяти, 'E' - правка памяти |
| | | Окно кода | | 'U' - просмотр кода по адресу, 'A' - вставка кода |
| | | Окно команд | | Здесь Вы набираете команды |

Другие важные клавиши (в стандартной настройке):

'H'/F1 - помощь

F5/Ctrl+D - запуск программы (или продолжение прерванной программы)

F8 - пошаговая отладка с заходом в тело функции

F10 - пошаговая отладка без захода в тело функции

F11 - выйти из функции (будет работать только до первого PUSH в функции)

3. ПОИСК РЕГИСТРАЦИОННЫХ КОДОВ

Возможно, наилучший способ попрактиковаться - это найти где-нибудь шареварную (shareware) программку и попытаться зарегистрировать ее.

3.1 Task Lock 3.00 - простая защита на основе серийного номера

Это очень простая защита: номер не зависит ни от каких факторов.

3.1.1 Медицинское обследование

Какой разрядности программа - 16 или 32 бит? Где вводится регистрационная информация? Даст ли мне справка какие-нибудь предположения о том, как устроена регистрация? Попробуйте ответить на эти вопросы перед тем, как мы продолжим.

....Сейчас Вы должны быть заняты обследованием....Вы заняты обследованием? ...Ну как, уже все?...

OK, теперь Вы знаете, что это 32-битное приложение, работающее под

Windows 95 и что регистрация заключается в заполнении регистрационного номера в диалоговом окошке, которое появляется когда Вы выбираете меню "Register|Register...". Из справки Вам также стало известно, что существует два типа регистрации: для индивидуального использования и для использования в "конторе" (в оригинале - site license). Поэтому очень вероятно, что в программе будет ДВЕ проверки регистрационных кодов.

3.1.2 Прерывание программы

Регистрационные коды чаще всего вводятся в обычных строчках ввода типа Windows Edit. Чтобы проверить код, программа должна прочитать содержимое строки ввода при помощи одной из функций:

16-бит 32-бит

------ ------

GetWindowText GetWindowTextA, GetWindowTextW

GetDlgItemText GetDlgItemTextA, GetDlgItemTextW

Последняя буква в названии 32-битных функций говорит о том, какие строки использует эта функция: однобайтовые или двухбайтовые. Двухбайтовые строки используются ОЧЕНЬ редко.

Возможно, что Вы уже уловили мою мысль. "Если бы можно было прерваться по вызову GetWindowText..." - и Вы МОЖЕТЕ это сделать!!! Но сперва Вы должны убедиться, что символьные имена (имена функций) загружены SoftICE'ом. Если Вы не знаете, как это сделать - см. приложение A.

Чтобы установить "ловушку" (на самом деле это называется точкой останова или брейкпоинтом) в SoftICE, Вы должны зайти в отладчик нажатием клавиш Ctrl-D и использовать команду BPX. В качестве параметра команды можно использовать либо имя функции, либо непосредственно адрес. Так как наш "объект изучения" (Task Lock) является 32-битным приложением, мы должны поставить брейкпоинт на функцию GetWindowTextA. Если это не поможет, попробуйте поставить брейкпоинт на другие функции.

В командной строке SoftICE наберите следующее:

:bpx getwindowtexta

Если Вы получите сообщение об ошибке (например, "No LDT"), убедитесь, что в фоне у Вас не выполняются никакие другие приложения. Я заметил, что Norton Commander в фоне является причиной подобного поведения SoftICE.

Вы можете проверить наличие брейкпоинтов командой:

:bl

В результате Вы увидите что-нибудь типа:

00) BPX USER32!GetWindowTextA C=01

Чтобы выйти из отладчика, нажмите Ctrl-D (или F5) еще раз.

Продолжим... Итак, Вы установили брейкпоинт и теперь SoftICE будет "выскакивать" при каждом вызове функции GetWindowTextA. Попробуем ввести какое-нибудь значение в окне регистрации и нажмем OK. Вы нажимаете OK... ...и получаете дурацкое сообщение о том, что Ваш код был неправильным. Значит, это была не функция GetWindowTextA... Попробуем GetDlgItemTextA. Удалим старый брейкпоинт:

:bc 0

(0 - это номер брейкпоинта в списке брейкпоинтов)

И установим новый:

:bpx getdlgitemtexta

Ну что ж, попробуем еще раз...

3.1.3 В отладчике

Wow! Работает! Теперь вы в SoftICE, в самом начале функции GetDlgItemTextA. Чтобы попасть туда, откуда она была вызвана, нажмите F11. Теперь Вы внутри модуля SGLSET.EXE. Если Вы не уверены - посмотрите на строчку между окном кода и окном командной строки, она должна выглядеть так:

----------SGLSET!.text+1B13----------

Сейчас Вы уже можете запретить реакцию на вызов функции:

:bd 0

Если Вам вдруг захочется снова разрешить ее, наберите:

:be 0

Первая строка в окне кода выглядит так:

CALL [USER32!GetDlgItemTextA]

Чтобы посмотреть строчки над ней, нажимайте Ctrl+Up ("стрелка вверх") до тех пор, пока не увидите нижеприведенный кусок кода. Если Вы ничего не понимаете в Ассемблере, я добавил комментарии которые могут Вам помочь.

RET ; Конец функции

PUSH EBP ; Начало другой функции

MOV EBP, ESP ; ...

SUB ESP, 0000009C ; ...

PUSH ESI ; ...

> LEA EAX, [EBP-34] ; EAX = EBP-34

PUSH EDI ; ...

MOVE ESI, ECX ; ...

PUSH 32 ; Макс. длина строки

> PUSH EAX ; Адрес текстового буфера

PUSH 000003F4 ; Идентификатор управления

PUSH DWORD PTR [ESI+1C] ; Идентификатор окна диалога

CALL [USER32!GetDlgItemTextA] ; Получить текст

Команды PUSH означают сохранение значений для последующего использования.

Я пометил важные строчки символом '>'. Глядя на этот код, мы видим, что адрес текстового буфера хранился в регистре EAX и что EAX был EBP-34h. Поэтому нам стоит взглянуть на EBP-34h:

:d ebp-34

Вы должны были увидеть текст, который вы ввели в диалоговом окне. Теперь мы должны найти место, где Ваш номер сравнивается с реальным серийным номером. Поэтому мы пошагово трассируем программу при помощи F10 до тех пор, пока не встретим что-нибудь о EBP-34. Не пройдет и нескольких секунд, как Вы наткнетесь на следующий код:

> LEA EAX, [EBP+FFFFFF64] ; EAX = EBP-9C

LEA ECX, [EBP-34] ; ECX = EBP-34

PUSH EAX ; Сохраняет EAX

PUSH ECX ; Сохраняет ECX

> CALL 00403DD0 ; Вызывает функцию

ADD ESP, 08 ; Удаляет сохраненную информацию

TEST EAX, EAX ; Проверяет значение функции

JNZ 00402BC0 ; Прыгает, если не "ноль"

Мне кажется, что это выглядит как вызов функции сравнения двух строк.

Эта функция работает так: на входе - две строки, на выходе - 0, если они равны и любое другое значание, если не равны.

А зачем программе сравнивать какую-то строчку с той, что Вы ввели в окне диалога? Да затем, чтобы проверить правильность Вашей строки (как Вы, возможно, уже догадались)! Так-так, значит этот номер скрывался по адресу [EBP+FFFFFF64]? SoftICE не совсем корректно работает с отрицательными числами и поэтому настоящий адрес следует посчитать:

100000000 - FFFFFF64 = 9C

Вы можете сделать это вычисление прямо в SoftICE:

:? 0-FFFFFF64

Число 100000000 слишком велико для SoftICE, а вычитание из 0 дает тот же

самый результат.

Наконец пришло время взглянуть, что же скрывается по адресу EBP-9C...

:d ebp-9c

В окне данных SoftICE Вы видите длинную строчку цифр - это серийный номер!

Но Вы помните, что я говорил Вам раньше? Два типа регистрации - два разных серийных номера. Поэтому после того, как Вы записали на бумажечку первый серийный номер, продолжайте трассировать программу при помощи F10. Мы дошли до следующего куска кода:

> LEA EAX, [EBP-68] ; EAX = EBP-68

LEA ECX, [EBP-34] ; ECX = EBP-34

PUSH EAX ; Сохраняет EAX

PUSH ECX ; Сохраняет ECX

> CALL 00403DD0 ; Снова вызывает функцию

ADD ESP, 08 ; Удаляет сохраненную информацию

TEST EAX, EAX ; Проверяет значение функции

JNZ 00402BFF ; Прыгает если не "ноль"

И что Вы видите по адресу EBP-68? Второй серийный номер!

:d ebp-68

Вот и все... Я надеюсь, что у Вас все получилось как доктор прописал? =)

3.2 Command Line 95 - легкая регистрация "имя-код", создание генератора ключей

Это программа - хороший пример, с легким алгоритмом генерации кода.

3.1.1 "Обследование"

Вы осмотрели программу и увидели, что это 32-битное приложение, требующее имя и код в окне регистрации. Поехали!

3.1.2 Прерывание программы

Мы поступаем так же, как и с Task Lock'ом - ставим брейкпоинты. Можно даже поставить сразу два брейкпоинта на наиболее возможные функции: GetWindowTextA и GetDlgItemTextA. Нажмите Ctrl-D, чтобы вызвать отладчик и наберите в окне команд:

:bpx getwindowtexta

:bpx getdlgitemtexta

Теперь возвращайтесь в прерванную программу, идите в окно регистрации и вводите имя и какой-нибудь номер (обыкновенное целое число - это наиболее вероятный код). Я написал примерно следующее:

Name: ED!SON '96

Code: 12345

Программа остановилась на GetDlgItemTextA. Так же, как и в случае с Task Lock'ом, мы нажимаем F11 чтобы вернуться в вызывающюю функцию. Просматриваем окно кода при помощи Ctrl+Up. Вызов функции выглядит так:

MOV ESI, [ESP+0C]

PUSH 1E ; Максимальная длина

PUSH 0040A680 ; Адрес буфера

PUSH 000003ED ; Идентификатор управления

PUSH ESI ; Идентификатор окна диалога

CALL [User32!GetDlgItemTextA]

Число 40A680 кажется нам интересным, поэтому мы проверяем этот адрес:

:d 40a680

Что же видно в окне данных, как не имя, которое мы ввели? =) А теперь взглянем на кусок кода под вышеприведенным:

PUSH 00 ; (не интересно)

PUSH 00 ; (не интересно)

PUSH 000003F6 ; Идентификатор управления

MOV EDI, 0040A680 ; Адрес буфера

PUSH ESI ; Идентификатор окна диалога

CALL [User32!GetDlgItemInt]

Функция GetDlgItemInt похожа на GetDlgItemTextA, но возвращает не строку, а целое число. Она возвращает его в регистре EAX, поэтому мы трассируем этот код (F10) и смотрим, что же у нас появилось в окне регистров после вызова функции... В моем случае оно выглядит так:

EAX=00003039

А что такое шестнадцатеричное 3039? Наберем:

:? 3039

И получим следующее:

00003039 0000012345 "09"

^ hex ^ dec ^ ascii

Как Вы видите (и, возможно, уже догадались) это код, который Вы ввели в диалоговом окне. Ok, что теперь? Посмотрим дальше:

MOV [0040A548], EAX ; Сохраняет рег. код

MOV EDX, EAX ; А также помещает его в EDX

3.1.3 Подсчитывание регистрационного кода

Мы достигли места, где подсчитывается реальный регистрационный код!

MOV ECX, FFFFFFFF ; Эти строчки подсчитывают

SUB EAX, EAX ; длину строки

REPNZ SCASB ; .

NOT ECX ; .

DEC ECX ; ECX теперь содержит длину

MOVSX EAX, BYTE PTR [0040A680] ; Получает байт по адр. 40A680h

IMUL ECX, EAX ; ECX = ECX \* EAX

SHL ECX, 0A ; Сдвиг влево на 0Ah бит

ADD ECX, 0002F8CC ; Добавляет 2F8CC к результату

MOV [0040A664], ECX

...И где он проверяется

CMP ECX, EDX ; Сравнивает числа

JZ 00402DA6 ; Прыгает, если равны

Когда Вы дотрассировали до сравнения чисел, Вы можете посмотреть, каким должен был быть Ваш РЕАЛЬНЫЙ регистрационный код:

:? ecx

В моем случае это дало:

000DC0CC 0000901324

То есть, правильный код для меня: 901324.

Нажмем F5 или Ctrl-D чтобы вернуться в программу и попробуем еще раз, но на этот раз с правильным кодом (в десятичной форме). Работает!

4. СОЗДАНИЕ ГЕНЕРАТОРА КЛЮЧЕЙ ДЛЯ COMMAND LINE 95

Взглянем на алгоритм генерации кода и попробуем перевести его на язык Си. Вот очень простая формула, по которой подсчитывается ключ:

code = ((uppercase\_first\_char \* length\_of\_string) << 0x0A) + 0x2f8cc;

Замечание #1: Не следует забывать, что все символы в окне ввода имени были приведены к верхнему регистру, поэтому мы должны сделать то же.

Замечание #2: "<< 0x0A" означает "умножние на 2 в степени 10"

Целиком программа на Си выглядит так:

#include <string.h>

#include <stdio.h>

int main()

unsigned long code;

unsigned char buffer[0x1e];

printf("CommandLine95 Keymaker by ED!SON '96\n"); printf("Enter name: ");

gets(buffer);

strupr(buffer);

code = ( ((unsigned long)buffer[0] \*

(unsigned long)strlen(buffer))

<< 0x0A) + 0x2f8cc;

printf("Your code is: %lu", code);

return 0;

Приятных сновидений!

4. КАК РАБОТАЮТ PUSH И CALL КОГДА ПРОГРАММА ВЫЗЫВАЕТ ФУНКЦИЮ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Снова | взглянем  PUSH  PUSH  PUSH  PUSH  CALL | на кусок кода из Task Lock'а:  32  EAX  000003F4  DWORD PTR [ESI+1C] [USER32!GetDlgItemTextA] | ;  ;  ;  ;  ; | Макс. длина строки  Адрес текстового буфера  Идентификатор управления  Идентификатор окна диалога  Получает текст |

Когда Вы вызываете функцию GetDlgItemTextA из программы на C, вызов выглядит так:

GetDlgItemTextA(hwndDlg, 0x3F4, buffer, 0x32);

^ [ESI+1C] ^ EAX

PUSH сохраняет данные в области памяти, называемой стеком. В результате каждого PUSH'а новый кусок данных помещается в верхушку стека и затем вызываемая функция проверяет, что лежит в стеке и использует эти данные по своему усмотрению.

5. О ПРОГРАММАХ НА VISUAL BASIC

EXE файлы, производимые Visual Basic'ом, не являются настоящими EXE. Они просто содержат код для вызова VBRUNxxx.DLL, который затем читает данные из EXE и выполняет программу. Такое устройство псевдо-EXE файлов является также причиной того, что программы на Visual Basic'е такие медленные.

А так как EXE файлы не являются настоящими EXE файлами, Вы не можете трассировать и дизассемблировать их - Вы найдете вызов функции из DLL и кучу мусора. И когда Вы будете трассировать такую программу, Вы "заблудитесь" в DLL.

Решением этой проблемы является декомпилятор. Существует декомпилятор для программ, написанных на Visual Basic'е версий 2 и 3, созданный кем-то, называющим себя DoDi. Эта программя является шареварной и ее можно найти в InterNet'е (см. Приложение C). Для программ, написанных на Visual Basic'е версии 4 (VB для Windows 95), не существует декомпилятора, насколько мне известно, хотя я бы хотел, чтобы он существовал. =)

Примечание: Настоящие программисты на пишут на Basic'е. =)

ПРИЛОЖЕНИЯ

A. КАК В SOFTICE ЗАГРУЖАТЬ СИМВОЛЬНЫЕ ИМЕНА

Чтобы проверить, загрузил ли SoftICE символьные имена GetWindowText, Вы должны войти в отладчик нажатием на клавиши Ctrl-D и в окне команд ввести следующее:

:exp getwindowtext

Если Вы не получили списка всех функций GetWindowText, Вам нужно отредактировать файл \SIW95\WINICE.DAT, удалив символ комментария (';') перед одной из строчек 'exp=', которые следуют за текстом: "Examples of export symbols that can be included for chicago" в конце этого файла.

Вы можете удалить комментарии из всех строчек 'exp=' или сохранить немножко памяти, раскомментировав только строчки с файлами kernel32.dll, user32.dll и gdi32.dll, которые являются самыми важными. После этого Вы должны перегрузить компьютер.

B. СИНТАКСИС НЕКОТОРЫХ ФУНКЦИЙ

Вам будет легче понять, как вызываются функции, о которых мы говорили, если Вы будете знать их описания (декларации):

int GetWindowText(int windowhandle, char \*buffer, int maxlen);

int GetDlgItemText(int dialoghandle, int controlid, char \*buffer, int maxlen); int GetDlgItemInt(int dialoghandle, int controlid, int \*flag, int type);

Если Вам нужна более подробная информация, посмотрите в руководстве программиста Windows/Win32.

C. ГДЕ НАЙТИ ПРОГРАММЫ

ПРОГРАММЫ ДЛЯ ВЗЛОМА

SoftICE/Win 2.oo: http://www.geocities.com/SoHo/2680/cracking.html Декомпилятор VB: ftp://ftp.sn.no/user/balchen/vb/decompiler/

ПРОГРАММЫ, ИСПОЛЬЗОВАННЫЕ В КАЧЕСТВЕ ПРИМЕРА

TaskLock: http://users.aol.com/Sajernigan/sgllck30.zip

CommandLine 95: ftp://ftp.winsite.com/pub/pc/win95/miscutil/cline95.zip

D. КАК СВЯЗАТЬСЯ С АВТОРОМ

На IRC (EFNet): Каналы #Ucf96, #Cracking

E-mail: edison@ccnux.utm.my или an461165@anon.penet.fi

На моей WWW-страничке: http://www.geocities.com/SoHo/2680/cracking.html