ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С. П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

ИССЛЕДОВАНИЕ РАЗРУШАЮЩИХ ПРОГРАММНЫХ ВОЗДЕЙСТВИЙ

Самара 2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

ИССЛЕДОВАНИЕ РАЗРУШАЮЩИХ ПРОГРАММНЫХ ВОЗДЕЙСТВИЙ

Составитель К.Е. Климентьев

Самара
Издательство Самарского университета
2017

УДК 003.26.09+004.056.55
ББК 32.972

Составитель ***К.Е. Климентьев***

Рецензент: к.т.н., доцент А.В.Б аландин

Исследование разрушающих программных воздействий: [Электронный ресурс]: метод. указания / сост. *К.Е. Климентьев*. - Самара: Изд-во Самарского университета, 2017. – 15 с. : ил. Электрон. текстовые и граф. дан. (Кбайт).- 1 эл. опт. диск (CD-ROM)

Предназначены для студентов, изучающих в рамках направления подготовки 09.03.01 «Информатика и вычислительная техника» курс «Защита информации» и прочие курсы аналогичной тематики.

Содержат необходимый теоретический и справочный материал для выполнения лабораторных работ. Также могут быть использованы в курсовом и дипломном проектировании.

Подготовлены на кафедре информационных систем и технологий.

УДК 003.26.09+004.056.55
ББК 32.972

Оглавление

Введение

Процесс развития информационных технологий предъявляет к его участникам требование защиты обрабатываемой информации, то есть поддержания ее качества на определенном уровне. Основные классы угроз качеству:

- нарушение конфиденциальности информации;
- искажение и уничтожение информации;
- блокирование доступа к информации.

Разрушающие программные воздействия – реализации этих угроз, выполняемые при помощи программ.

Вредоносные программы – программы, специально созданные для выполнения разрушающих воздействий. Статья 273 УК РФ предусматривает наказание за создание, использование и распространение вредоносных программ.

Так же в выполнении разрушающих программных воздействий могут участвовать и «обычные» программы – штатные утилиты операционных систем, программы-отладчики, текстовые и графические редакторы и прочее.

«*Взлом*» *программного обеспечения* – сложное разрушающее воздействие, направленное на искажение алгоритма работы программного обеспечения.

Ц_е_л_ь л_а_б_о_р_а_т_о_р_н_о_й р_а_б_о_т_ы: ознакомление с методами разрушающих программных воздействий и принципами противодействия им.

В процессе выполнения лабораторной работы необходимо:

- изучить необходимые предпосылки;
- откомпилировать «тестовую» программу;
- «взломать» эту программу предлагаемыми способами и продемонстрировать преподавателю;
- (опционально) написать «стойкий» вариант программы.

1. Технические предпосылки

1.1. Технологии создания и выполнения программ

В настоящее время используются несколько технологий создания и выполнения программ.

I. Исходный текст программы обрабатывается программой-компилятором, результатом чего является машинный код, который после запуска программы выполняется непосредственно центральным процессором компьютера (см. рис. 1).

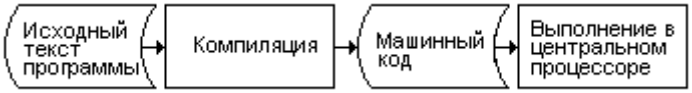

Рис. 1. Компилируемые программы

Примеры таких языков программирования: C/C++, Pascal.

II. Исходный текст программы обрабатывается первичным компилятором, результатом чего является промежуточный «байт-код» (или «p-код»). После запуска программы вторичный компилятор специальной «виртуальной машины» на лету (английский термин JIT – Just In Time) транслирует инструкции байт-кода в машинные команды и передает их для выполнения в центральный процессор компьютера (см. рис. 2).

Рис. 2. Дважды компилируемые программы

Примеры таких языков программирования: Java, C#. Примеры «виртуальных машин»: JVM (универсальная), CLR (для Windows), Mono (для Unix), Dalwik и ART (для Android).

III. Исходный текст программы подается на вход специальной программы-интерпретатора (фактически, это разновидность «виртуального процессора»), который самостоятельно распознает команды и моделирует их выполнение (см. рис. 3).

Рис. 3. Интерпретируемые программы

Примеры таких языков программирования: JavaScript, Perl, VBA (Visual Basic for Application). Программы на таких языках называются «скриптами» или «макросами».

1.2. Формат программного файла

Файлы программ-скриптов, как правило, представляют собой обычные текстовые файлы и никакой особенной внутренней структуры не имеют.

Рис. 4. Структура EXE-файла

Внутренняя структура файлов, содержащих откомпилированные программы, упрощенно может быть проиллюстрирована рис. 4. В начале файла располагается область заголовков и служебных таблиц. Остальная часть файла разбита на «секции», каждая из которых содержит ту или иную информацию. Например, внутри файла присутствует специальная секция для машинного кода, секция для инициализированных данных, секция для программных ресурсов и т.п. В случае, если программа изготовлена по технологии .NET (то есть, например, написана на C#), секция машинного кода содержит только маленькую программку для инициализации виртуальной машины CLR, а сам алгоритм программы, откомпилированный в «байт-код», располагается в отдельной секции.

В данных методических указаниях предполагается, что «тестовая» программа написана на C/C++ и компилируется сразу в машинный код (см. рис. 1).

1.3. Изготовление тестовой программы

Откомпилируйте следующую программу.

```
#include <stdio.h>
#include <string.h>
int i=0;
int flag = 0;
unsigned char etalon[]={”xyz”};
int check_pass() {
 unsigned char password[4];
 printf("Enter password > ");
 scanf("%s", password);
 while (etalon[i]!=0) if (password[i]!=etalon[i++]) flag=1;
 return flag;
}
int main() {
 _asm { cpuid };
 if (check_pass("xyz")==0)
  printf("OK");
 else
  printf("Bad password");
 return 0;
}
```

П_р_и_м_е_ч_а_н_и_е. Директива _asm { cpuid } вставляет в начало программы «редкую» машинную команду CPUID с числовым кодом $A20F_{16}$. Это нужно для того, чтобы проще найти начало программы в машинном коде.

П_р_и_м_е_ч_а_н_и_е. Важно отключить в компиляторе проверку целостности стека. Так же для 64-разрядных операционных систем целесообразно отключить запрет выполнения стека (режим NX) и случайное перемешивание системных библиотек в памяти (режим ASLR).

Например, для компиляторов Microsoft Visual C/C++ Studio – см. рис. 5 и 6 .

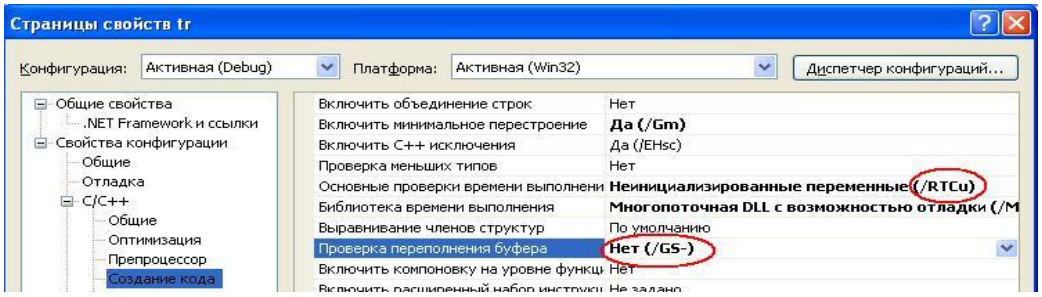

Рис 5. Отключение проверки целостности стека

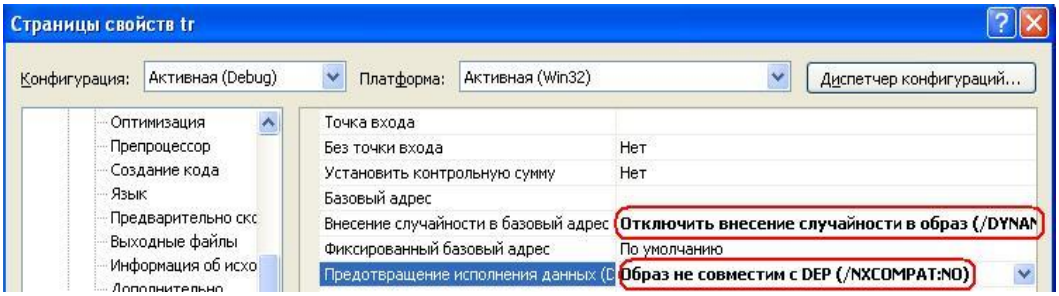

Рис. 6. Отключение режимов NX и ASLR

2. Использование утилиты HIEW32

Декомпиляция – преобразование машинного кода или байт-кода в исходный текст программы на том или ином языке программирования.

Дизассемблирование – декомпиляция машинного кода или байт-кода в текст на языке ассемблера.

Программа HIEW (автор Е. Сусликов) – дизассемблер программ с возможностью редактирования машинного кода. Существуют условно-бесплатные версии этой программы.

Скопируйте в один каталог с исследуемыми программами и запустите утилиту HIEW32.EXE. При помощи появившегося списка файлов загрузите файл вашей программы (например, PRIMER.EXE). Вы увидите содержимое программного файла (а именно - заголовок, поскольку он располагается в начале файла).

2.1. Режимы отображения файла программы

Возможны три различных режима отображения внутреннего содержимого файла. Вы можете переключаться из режима в режим, нажимая F4 и выбирая желаемый вариант из меню.

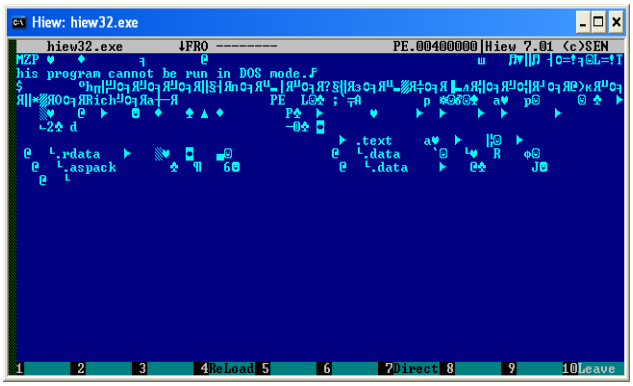

Рис.7. Текстовое представление файла

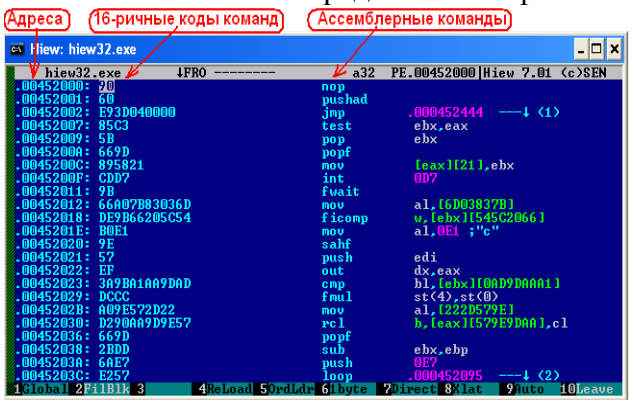

Рис. 8. 16-ричное представление (дамп) файла

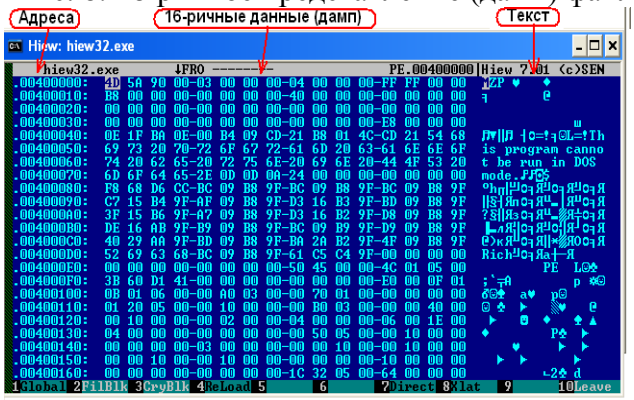

Рис. 9. Представление в виде кода

I. Текст (режим «Text» установлен по умолчанию). Значение каждого файлового байта представляется как код некоторого текстового символа, а весь файл – как некий, большей частью бессмысленный, текст (см. рис. 7).

II. Шестнадцатеричные данные (режим «Hex») плюс текст. Значение каждого файлового байта представляется как 16-ричное число. Числовые значения байтов представляются в виде таблицы (по 16 байтов в строке), причем для повышения информативности в левой колонке отображается номер первого байта в строке, в следующих 16-ти колонках – числовые значения байтов, а в правой колонке - текстовое представление всех байтов строки (см. рис. 8). Например, байту с кодом 4Dh соответствует знак ‘M’, а байту с кодом 5Ah – знак ‘Z’. Попытайтесь ответить: какое числовое значение имеет символ ‘P’?

III. Машинный код (режим «Decode»). Значение каждой группы байтов интерпретируется как код некоторой машинной команды и отображается в виде соответствующей ассемблерной инструкции. Для информативности в левой колонке показываются адреса (номера байтов), с которых начинаются команды; в следующей колонке показываются 16-ричные значения байтов, образующих команду; в третьей колонке показывается ассемблерная мнемоника команды (см. рис. 9).

В любом из режимов («Текст», «Шестнадцатеричные данные» или «Машинный код») вы можете перемещаться по файлу клавишами-стрелками (построчно) или клавишами «Page Up»/«Page Down» (постранично). Сразу на начало/конец файла можно перейти при помощи «Ctrl+Home» и «Ctrl/End».

Сделайте несколько переключений из режима в режим, выполните несколько перемещений по файлу и придите к выводу: не существует универсального режима отображения, каждому региону программы (см. п. 2 и рис. 4) должен соответствовать свой режим.

2.2. Навигация по файлу программы

Утилита HIEW32 позволяет находить в файле различные структурные компоненты программы (см. рис. 4) и отображать их.

Давайте попробуем найти в файле «начало программы».

Именно для того, чтобы легко найти первые команды нашей программы, мы вставили в самое начало нашей программы редко используемую машинную инструкцию CPUID с кодом $A20F_{16}$.

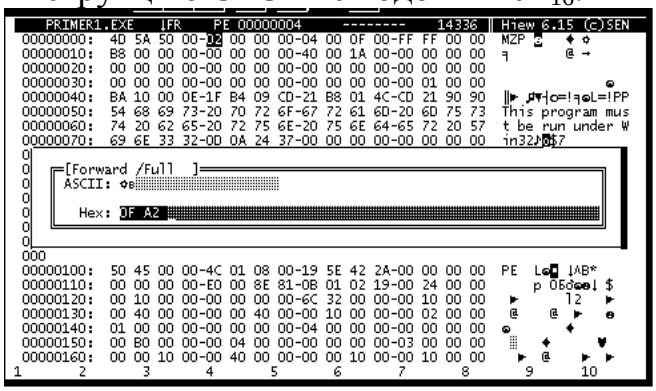

Рис. 10. Задание образца для поиска

Установите для вашей программы режим отображения «шестнадцатеричные данные» и нажмите F7 – вы попадете в режим поиска по файлу (см. рис. 10). Теперь можно задать образец для поиска либо в виде строки (поле «ASCII»), либо в виде 16-ричных чисел (поле «Hex»). Перемещаться между полями можно при помощи клавиши «Tab». Итак, задайте для поиска два 16-ричных числа 0F A2 (именно в таком порядке, поскольку микропроцессоры Intel сначала воспринимают младшие, а потом старшие байты!) и нажмите «Enter».

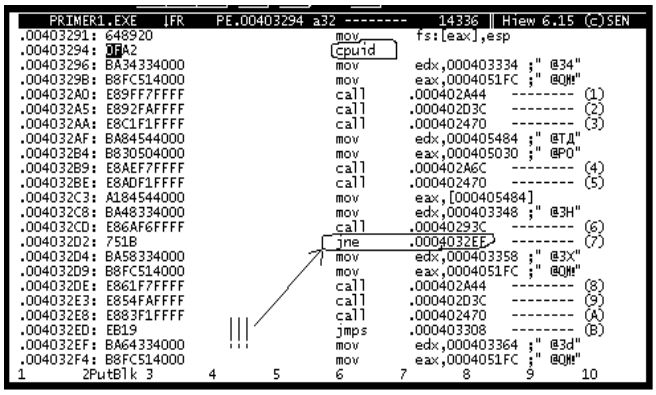

Рис. 11. Начало программы

Если вы все сделаете правильно, то попадете в «начало программы» (но не в «начало файла», т.к. там располагается заголовок, а сама программа где-то дальше!). Установите для вашей программы режим отображения «программный код». Вы должны увидеть примерно следующее (см. рис. 11).

П_р_и_м_е_ч_а_н_и_е. В тестовой программе машинный код и адреса другие!!!

Другие способы перемещения по файлу программы.

В режимах «16-ричные данные» или «машинный код» клавиша «F8» отображает служебный заголовок программы. Теперь клавиша «F5» переносит на первую исполняемую команду программы (где обычно располагается характерный для компилятора стартовый код). Кроме того, клавиша «F6» переносит к таблице импорта программы, а «F9» - к таблице экспорта.

В режимах «16-ричные данные» или «машинный код» клавиша «F5» открывает в верхнем левом углу экрана поле ввода адреса, на который требуется произвести переход. По умолчанию предполагается, что программа будет размещена в памяти по адресу «400000_{16}». Если ввести в этом поле «123», то переход будет выполнен

относительно начала в памяти, то есть на адрес «400123». Для указания абсолютного адреса перехода необходимо ввести число с точкой: «.400123».

3. Метод взлома 1: изменение строки пароля

Откройте при помощи HIEW32 тестовую программу. Перейдите на начало файла. Включите 16-ричный режим отображения. Найдите строку «xyz». Перейдите в режим редактирования, нажав «F3».

Исправьте пароль на любой другой, состоящий из 3 букв. Если курсор в левой части окна (где 16-ричные коды), то вводить надо 16-ричные ASCII-коды букв и цифр. Если он в правой части окна (где текст), то вводить можно сразу буквы и цифры. Переключиться справа налево и наоборот можно при помощи TAB. Отменить исправления можно при помощи ESC. Сохранить изменения можно при помощи F9.

Запустите программу и проверьте, что она работает с новым паролем.

4. Метод взлома 2: изменение машинного кода

Откройте при помощи HIEW32 тестовую программу. Перейдите на начало файла. Включите 16-ричный режим отображения. Найдите команду CPUID (см. пункт 2.2).

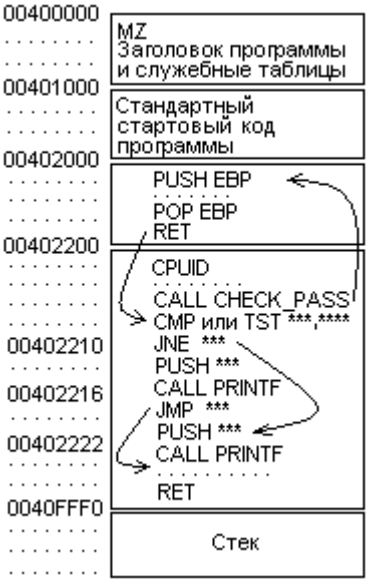

Рис. 12. Примерная структура программы

Переключитесь в режим машинных команд. Перед вами ассемблерный код программы, имеющий примерно такой вид, как на рис. 12. (В реальности адреса и команды другие!)

Командой CALL в функции main() происходит вызов функции check_pass(). После этого командой TEST анализируется результат и командами условного перехода (например, JE, JNE и т.п.) выполняется разделение программы на две ветви – ту, где выводится «OK» и ту, где «Bad password».

Определите и запишите начальные адреса обеих ветвей.

Сделайте так, чтобы переход при любом пароле происходил на ветвь «OK». Например, для этого может потребоваться удалить какую-нибудь команду условного перехода или изменить ее на противоположную. Для этого:

- перейдите в режим редактирования файла (клавиша F3);
- стрелками подведите курсор к коду команды условного перехода; запишите на это место коды **двух** команд-пустышек («NOP»), которые равны 90_{16};
- сохраните изменения в файле (клавиша F9).

П_р_и_м_е_ч_а_н_и_е. Команда условного перехода состоит из 2-х байтов. Если вы измените только первый байт команды, то второй будет рассматриваться процессором как начало какой-то другой, неправильной команды.

Выйдите из HIEW32 и проверьте, что программа выводит сообщение «OK» с любым паролем.

5. Метод взлома 3: использование уязвимостей

Откройте при помощи HIEW32 тестовую программу.

Идея взлома. Команда CALL, которая вызывает функцию check_pass(), сохраняет адрес возврата в стеке. Так же, в стеке сохраняются адреса всех передаваемых в функцию check_pass() параметров. Так же в стеке сохраняются все локальные переменные и массивы функции check_pass(), в том числе и массив password длиной 4 байта. Если в ответ на запрос пароля ввести длинную строку, то она «затрет» весь стек, включая адрес возврата. Суть в том, чтобы во вводимом пароле на «нужном» месте оказался «нужный» адрес возврата.

Определите то место в строке пароля, которым затирается адрес возврата. Для этого запустите программу и введите длинный пароль, например «0123456789abcd». Программа должна аварийно завершиться с сообщением, например: «попытка перехода на адрес 37363534». Но 37 – это ASCII-код цифры «7», 36 – код цифры «6» и т.д. ASCII-коды некоторых знаков см. на рис. 13.

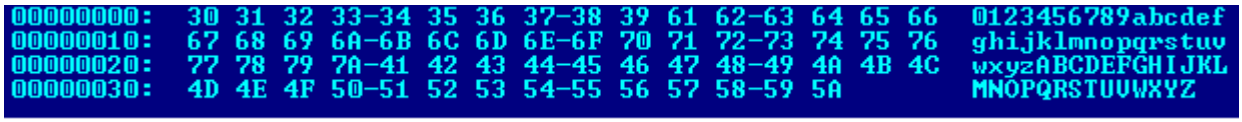

Рис 13. ASCII-коды некоторых знаков

Значит, первые 4 знака могут быть любыми, а с 5-го по 8-ой должны содержать «нужный» адрес. (П_р_и_м_е_ч_а_н_и_е. В тестовой программе эта позиция другая!).

В «Методе 2» вы записали адрес ветви «OK», например, он 00401234. (П_р_и_м_е_ч_а_н_и_е. В реальной тестовой программе этот адрес другой!). Нужно передать программе пароль, в котором на 5-8 местах стояли бы символы с кодами 34, 12, 40 и 00 (это 16-ричные коды!). Для этого переведите 16-ричные коды в 10-чные, это будут 52, 18 и 64. Запустите программу и вводите пароль: первые 4 знака любые, потом зажимаете Alt, набираете на дополнительной клавиатуре «52», отпускаете Alt - вводится символ с кодом 52. Точно так же вводите остальные два символа, а код 0 получается автоматически – как конец строки.

Нажимаете Enter. Если все сделано правильно, адрес возврата в стеке должен замениться вашими кодами, и после выхода из check_pass() переход должен произойти на ветку «OK».

6. Метод взлома 4: трейнер

Трейнер – программа, которая «на лету» изменяет код другой, уже запущенной программы. Такое свойство полезно, например, для отладчика. Принципы работы трейнера.

1. Трейнер должен запустить отлаживаемую программу.
2. Трейнер должен открыть отлаживаемую программу (примерно как файл). При этом трейнеру будет доступно все адресное пространство отлаживаемой программы от 00000000 до $FFFFFFFF_{16}$, включая области, где расположена сама программа, сервисные библиотеки и т.п.
3. Трейнер должен установить для региона адресного пространства, с которым сбирается работать, необходимые права доступа.
4. Трейнер может работать с адресным пространством примерно как с файлом – считывая оттуда данные при помощи ReadProcessMemory() и записывая при помощи WriteProcessMemory() .
5. Трейнер должен закрыть отлаживаемый процесс при помощи CloseHandle().

Откомпилируйте программу-трейнер.

```
#include "stdafx.h"
#include <stdio.h>
#include <windows.h>
#define PROGNAME L".\\crackme.exe"
STARTUPINFO si;
PROCESS_INFORMATION pi;
WORD  nopnop = 0x9090;  /* Коды двух команд NOP */
DWORD old_flags;        /* Старые флаги доступа */
DWORD p = 0x12345678;   /* Адрес, куда вписывать байты */
int main() {
  CreateProcess(PROGNAME, L"" , NULL, NULL, TRUE, 0 , NULL, NULL,
               (STARTUPINFO *) &si, (PROCESS_INFORMATION *) &pi);
  HANDLE h=OpenProcess(PROCESS_VM_OPERATION|PROCESS_VM_READ|PROCESS_VM_WRITE,
                TRUE, pi.dwProcessId);
  VirtualProtectEx(h, (void *) p, 2, PAGE_EXECUTE_READWRITE, &old_flags);
  WriteProcessMemory(h, (void *) p, (void *)&nopnop, 2, NULL);
  CloseHandle(h);
  system("pause >nul");
}
```

Эта программа запускает тестовую программу (укажите в PROGNAME путь к ней). Потом трейнер открывает вашу программу при помощи OpenProcess() и записывает в ее адресное пространство по адресу «p» содержимое переменной «nopnop». Адрес «p» – адрес того места, которое вы «исправили» в «Методе 2». Переменная «nopnop»содержит коды двух команд «NOP».

Запустите трейнер и введите любой пароль из 3-х букв. Если все сделано правильно, то программа изменится прямо в памяти, и должно появиться «OK».

7. Метод взлома 5: пересборка программ

Пересборка программ включает в себя следующие действия:

- полная декомпиляция или дизассемблирование программы;
- изменение ее исходного текста;
- повторная компиляция программы.

Наиболее просто пересборка программ выполняется для дважды компилируемых программ, написанных на языках C# или Java.

В рамках лабораторной работы пересборка тестовой программы не требуется.

Обсуждение результатов и выводы

I. Метод взлома 1 возможен благодаря тому, что эталонный пароль хранился внутри программы в открытом виде.

Способы противодействия: хранить эталон в зашифрованном виде (а перед сравнением расшифровать) или в виде контрольной суммы. Второй способ предпочтительней, так как по контрольной сумме невозможно восстановить исходный пароль.

II. Методы взлома 2 и 5 возможны благодаря тому, что алгоритм программы хранится в файле в открытом виде.

Способ противодействия: использовать специальные программы-упаковщики, например UPX. Упаковщик подвергает код программы сжатию методом LZW. При запуске загрузчик «распаковывает» защищаемую программу непосредственно в памяти.

Другой способ противодействия: *обфускация* исходного текста программы, то есть искусственное «запутывание» алгоритма, приведение его к «непонятному» виду.

III. Метод взлома 3 возможен благодаря грубым ошибкам в алгоритме и технологии программирования.

Во-первых, внутри процедуры использован статичный массив ограниченного размера. Рекомендуется использовать динамически определяемые массивы:

unsigned char *password = new unsigned char[4].

Во-вторых, во время компиляции отключены режимы проверки целостности стека. Это ускорило программу, сделало ее объем меньше, но не позволило отследить «кибератаку». Включение этих режимов рекомендуется сочетать с самостоятельной обработкой исключений при помощи директив «try» и «catch».

В-третьих, использованы стандартные, но «опасные» функции, которые не контролируют размера вводимых данных. Так, например, вместо «gets» и «scanf» рекомендуется использовать «gets_s» и «scanf_s».

Наконец, использован язык программирования C/C++, который позволяет отключить проверку целостности стека и длины вводимых данных. Существуют языки (например, Pascal), которые всегда выполняют такие проверки.

IV. Метод взлома 4 возможен благодаря тому, что использован язык программирования C/C++, предусматривающий компиляцию исходного текста в машинный код. Современные языки программирования C#, Java и прочие предусматривают компиляцию в байт-код, который не может быть подвержен воздействию со стороны программы-трейнера (см. п. 1.1).

З_а_д_а_н_и_е (опционально): напишите на языке C++ «взломостойкий» вариант тестовой программы, в котором:

- эталон пароля хранится и проверяется не в виде строки, а в виде контрольной суммы от строки;
- внутри процедуры ввода пароля использованы «безопасные» функции, а возможное переполнение стека контролируется при помощи «try» и «catch».

Упакуйте его при помощи утилиты UPX. Продемонстрируйте, что рассмотренные в методических указаниях методы «взлома» неэффективны.

Методические материалы

ИССЛЕДОВАНИЕ РАЗРУШАЮЩИХ ПРОГРАММНЫХ ВОЗДЕЙСТВИЙ

Методические указания

Составитель ***Климентьев Константин Евгеньевич***

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ имени академика С.П. Королева»
(Самарский университет)
443086, САМАРА, МОСКОВСКОЕ ШОССЕ, 34

Изд-во Самарского университета
443086, Самара, Московское шоссе, 34.