УДК 003.26

Стеганография в файлах формата Portable Executable

И. В. Нечта

В статье рассматриваются методы сокрытия информации в файлах формата Portable Executable.

Ключевые слова: стеганография в исполняемых файлах, формат PE, отпечатки пальцев

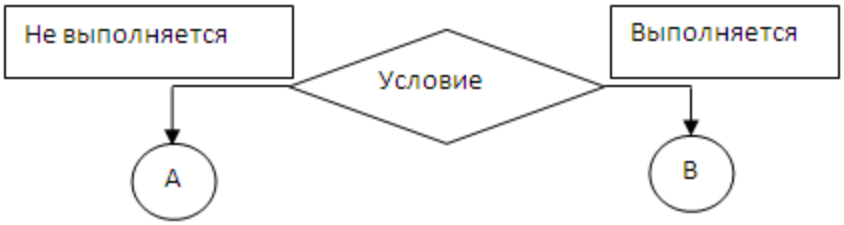
1. Введение

На сегодняшний день использование нелицензионного программного обеспечения наносит значительный экономический ущерб компаниям, его производящим. Для предотвращения такого использования существуют различные программные и аппаратные средства и методы защиты. Среди них известен метод цифровых отпечатков пальцев, суть которого состоит в том, что в защищаемый файл вносятся такие изменения, которые могли бы однозначно идентифицировать каждую копию. Если некто решит незаконно скопировать и распространять программу, то можно будет по отпечатку пальца определить, какая копия незаконно распространяется. Естественно, необходимо позаботиться о том, чтобы знаки были такими, чтобы их не мог обнаружить владелец копии. Для этого обычно прибегают к методам стеганографии.

В данной работе рассматриваются методы сокрытия информации в файлах формата Portable Executable (PE, «PE» — это основной формат исполняемых файлов приложений в 32- и 64-разрядных системах Microsoft Windows).

2. Анализ существующих методов

Известен метод, предложенный в работе [1]. Суть этого метода заключается в том, что ветвления кода программы можно записывать в различном порядке. Приведём пример части алгоритма программы, представленного в виде блок-схемы:



Здесь представлена ситуация ветвления кода. Если условие выполнится, то произойдёт переход выполнения программы на участок кода «B», а если не выполнится, то на «A». Порядок расположения участков кода «A» и «B» в памяти компьютера может быть различным. Следовательно, программа может быть представлена на псевдокоде так:

Способ 1	Способ 2

если условие то goto A
метка B:
  код “B”
  goto exit
метка A:
  код “A”
метка exit:
	если NOT условие то goto B
	метка A:
	  код “A”
	  goto exit
	метка B:
	  код “B”
	метка exit:

Расстановка участков кода происходит на стадии преобразования исходного текста программы в файл формата PE. Меняя участки кода в лексикографическом порядке, можно добиваться сокрытия информации.

Также существует и другой метод, предложенный в работе [2]. Он заключается в возможности заменить некоторые операции на тождественные. Например, $A := A + 10$ можно заменить на $A := A - (-10)$; обнуление может быть $a := 0$, или $a := a \; xor \; a$, или $a := a - a$ и т.д. Как считает автор, данный подход отнюдь не является самым эффективным: например, с помощью подсчёта вычитаний отрицательных чисел можно достаточно легко определить, содержит ли исполняемый файл скрытое сообщение или нет.

Существуют также и другие методы, предложенные, например, в работе [3].

3. Описание предлагаемых методов

В данной работе предлагается три новых метода сокрытия информации, описываемые ниже.

3.1. Первый метод

Метод базируется на том, что можно переставлять местами две рядом стоящие операции присваивания, результат работы которых не зависит от последовательности выполнения. Обозначим присваивание как `:=`. Например:

Способ 1	Способ 2
a:=d	b:=c
b:=c	a:=d

Перестановка изменяет ход (порядок) вычислений, но не искажает результат. Если операции идут в лексикографическом порядке, то спрятанный бит равен 1, иначе 0. Главной задачей будет являться поиск таких операций, которые можно поменять местами. Особенность метода заключается в том, что некоторые файлы могут иметь переходы типа `jmp[регистр]`, что делает невозможным перестановку пар присваиваний. Например:

```
метка A:
  a:=1;
метка B:
  b:=b+1;
тело цикла
jmp [регистр]
```

Если регистр содержит адрес метки B, то при перестановке в теле цикла окажется другая операция, что приведёт к нарушению алгоритма работы программы. Следовательно, существуют файлы, к которым эти методы неприменимы.

Сформулируем требования к парам операций, которые можно использовать для передачи “скрытой” информации. Эти операции должны удовлетворять следующим условиям:

- являться операциями присваивания;
- располагаться рядом друг с другом;
- быть независимыми друг от друга, т.е. конечный результат их работы не должен зависеть от последовательности выполнения;
- операции не должны быть одинаковыми.

Кроме того, в программе не должно быть переходов на вторую операцию минуя первую, в том числе и косвенных (относительных) переходов, адреса которых нельзя определить или исправить.

Приведём пример подобных операций во фрагменте программы на языке ассемблера (интересующие нас операции выделены жирным шрифтом).

```
xor   ax, ax
mov   dx,ax
mov   word ptr ds:[offset var1], ax
 …
```

Теперь рассмотрим процесс сокрытия и получения сообщения. Условно назовем именем «Алиса» того, кто скрывает сообщение в файле, а именем «Боб» того, кто это сообщение получает. Если передаваемый бит скрытого сообщения равен 1, то операции расставляются в лексикографическом порядке (обозначим такую пару через <*a, b*>). Если передаваемый бит равен 0, то расставим операции в обратном порядке <*b, a*>.

Алиса выбирает файл формата PE и осуществляет в нём поиск пар операций присваивания. Далее Алиса изменяет последовательность операций в найденных парах в соответствии с передаваемым сообщением и отсылает файл Бобу. Боб также осуществляет поиск пар операций присваивания и в соответствии с лексикографическим порядком операций в парах получает переданное скрытое сообщение.

3.2. Второй метод

Метод базируется на возможности перестановки процедур внутри программы. Если в программе переставить процедуру с одного места на другое и исправить все внутренние и внешние ссылки, которые должны поменяться вследствие перемещения, то работоспособность программы не будет нарушена. Приведём пример такой перестановки:

Вариант 1	**Вариант 2**
Proc1	Proc2
Proc2	Proc1

Здесь Proc1, Proc2 – процедуры. Отличие в том, что поменялось место расположения процедур относительно друг друга.

Сформулируем требования к программам, в которых можно применять описываемый метод. Эти программы должны удовлетворять следующему условию: отсутствие косвенных переходов из одной процедуры на другие, таких, что адреса переходов нельзя определить или исправить.

Число возможных перестановок таких процедур равно $N!$, где N – число процедур. Соответственно, число бит, передаваемых таким образом, равно $\log_2(N!) \approx N \log_2 N$.

3.3. Третий метод

Метод базируется на возможности перестановки адресов в таблицах импорта, вызывающих функции Windows API. Адреса импортируемых функций API операционной системы лежат внутри массива. Заполнение этого массива производится операционной системой на стадии запуска программы. Соответствие номера элемента массива и имени функции задаётся в таблице импорта файла, которую можно модифицировать. Следует отметить, что существует следующая зависимость расположения элементов массива. Программа загружает несколько библиотек, в каждой библиотеке находится множество импортируемых функций. Зависимость заключается в том, что перечисления адресов функций сгруппированы по библиотекам (сначала перечислены функции одной библиотеки, потом другой, и т.д.). Соответственно, перестановки возможны только внутри этих групп. Число возможных перестановок равно $\sum_{i=1}^{n} M_i!$, где n – число импортируемых библиотек, M_i – число импортируемых функций для i-той библиотеки. Следовательно, число бит, передаваемых таким методом, равно $\log_2(\sum_{i=1}^{n} M_i!$.

Описанные методы подходят к файлам исполняемого формата PE. Все три метода могут быть одновременно применены к файлу для сокрытия информации.

4. Экспериментальный анализ

Теперь определим эффективность методов экспериментально. В качестве критерия эффективности возьмём среднее отношение размера файла к максимально возможному объёму скрываемого сообщения.

Для исследования эффективности вышеописанных методов было случайно отобрано и проанализировано 1000 файлов, составлена таблица значений выбранного критерия эффективности. Средний размер файла равен 51250 байт.

Таблица 1. Эффективность по каждому методу

Первый метод	Второй метод	Третий метод	Три метода вместе
0.07 %	0.248 %	0.074 %	0.4 %

Из обработанных файлов 0.8 % оказались негодными для вставки. Максимальное значение критерия эффективности составляет 1.7 % .

5. Статистический анализ

При вставке скрытого сообщения описанными методами может нарушиться присущее большинству файлов соотношение пар операций присваивания, идущих в лексикографическом и обратном к нему порядках. В результате для выявления факта передачи сообщения может использоваться статистический тест, выявляющий отклонение этого соотношения от среднестатистического. Следовательно, необходимо убедиться в том, что при вставке сообщения доля пар <*a, b*> в файле существенно не изменится, что будет означать невозможность выявления факта передачи сообщения статистическим тестом.

Перейдём к статистическому исследованию первого метода. Для этого определим долю пар, идущих в лексикографическом порядке, для файлов, не содержащих скрытую информацию, и долю пар в файлах с максимально возможным объёмом скрытой информации. Для этого случайно отберём 1000 файлов и произведём в них подсчёт доли пар $<a, b>$. Для каждого файла возьмём в качестве скрываемого сообщения сгенерированную случайную последовательность бит и встроим в него. Объём сообщения равен максимально возможному объёму скрываемой информации для данного файла, т.е. файл будет заполнен на 100 %. Далее заново произведём подсчёт доли пар $<a, b>$. Стандартные отклонения и доверительные интервалы посчитаны по стандартной методике, см., например, [4].

Таблица 2. Результаты статистического теста

Уровень заполнения	Средняя доля пар в лексикограф. порядке	Стандартное отклонение	Доверительный интервал
0 %	53 %	0.198	[51.45 %; 54.96 %]
100 %	50 %	0.902	[44.4 %; 55.59 %]

Исследование показало, что распределение пар практически не меняется. Следовательно, обнаружение скрытой информации с помощью статистических тестов невозможно.

Литература

1. Непомнящий А. Стеганография в исполняемых файлах [Электронный ресурс]. URL: http://www.rol.ru/n-ews/it/soft/03/02/27_006.htm (дата обращения: 30.12.2008).
2. Davidson R. I. Method and system for generating and auditing a signature for a computer program - United States Patent 5559884 [Электронный ресурс]. URL: http://www.free-patentsonline.com/5999972.html (дата обращения: 30.12.2008).
3. Stern J. P., Hachez G., Koeune F., Quisquater J.-J. [Электронный ресурс]. URL: http://www.dice.ucl.ac.be/ crypto/publications/code_wotermarki-ng.pdf (дата обращения: 30.12.2008).
4. Вентцель Е. С., Овчаров Л. А. Теория вероятностей и её инженерные приложения. М.: Наука, 1988. 480 с.

Статья поступила в редакцию 19.01.2008

Нечта Иван Васильевич
аспирант, ассистент кафедры прикладной математики и кибернетики СибГУТИ,
тел. (383) 2-698-272, e-mail: www@inbox.ru.

Steganography in Portable Executable format files

I. Nechta

Methods of data hiding in Portable Executable format files are considered.

Keywords: executable files steganography, PE format, fingerprints