УДК 003.26

Метод внедрения скрытых сообщений в исполняемые файлы

И. В. Нечта

В статье предлагается модификация существующего метода внедрения информации в исполняемые файлы. Предыдущий метод внедрял секретное сообщение в неиспользуемые места исполняемого файла. Внедрённое сообщение имеет статистические отличие от кода программы, что делает метод внедрения уязвимым к статистическим атакам. В статье предлагается предварительно кодировать внедряемое сообщение так, чтобы распределения вероятностей байт секретного сообщения и кода программы были неотличимы. Предложенный подход повышает устойчивость метода внедрения к статистическим атакам.

Ключевые слова: стеганография, стегоанализ, Portable Executable.

1. Введение

Стеганография — это наука о скрытой передачи информации путём сохранения в тайне самого факта передачи. В настоящее время стеганография широко применяется в целях защиты авторских прав программного обеспечения. В каждую продаваемую копию программы (так называемый “контейнер”) с помощью специальных алгоритмов внедряется особое сообщение — водяной знак, по которому в случае обнаружения нелегальной (пиратской) копии с лёгкостью может быть прослежен исходный файл, с которого была снята эта копия и соответственно пользователь, нарушивший лицензионное соглашение. Считается, что внедряемый водяной знак должен обладать некоторой степенью скрытности, то есть нельзя ничего сказать ни о наличии в файле водяного знака, ни о его местоположении. В настоящее время существуют различные методы, позволяющие внедрить водяной знак в программу. В ходе этой работы рассматривался метод, опубликованный в работе [1] и основанный на внедрении секретного сообщения в неиспользуемые места секции исполняемых файлов формата Portable Executable (PE)[1] . Каждая секция такого файла должна быть размером, кратным полю FileAligment (см. документацию Microsoft [2]). Таким образом, секция кода состоит из двоичных инструкций программы и нулевых байтов выравнивания, увеличивающих секцию до требуемых размеров. Авторы работы [1] предлагают вместо нулевых байтов выравнивания записывать секретное сообщение. По их мнению, это имеет некоторые преимущества: размер файла остаётся неизменным и внедрённое сообщение никак не влияет на ход выполнения программы. Предполагается, что передаваемое сообщение будет предварительно зашифровано и для его прочтения необходимо знать секретный ключ, а также начальную позицию сообщения в файле. Однако в статье [3] было показано, что данный подход не является устойчивым из-за различия статистических свойств кода программы и передаваемого за-

1 Portable Executable — формат исполняемых файлов (программ); используется в операционных системах семейства Windows NT.

шифрованного сообщения. Существует также и обратная задача стеганографии — стегоанализ, заключающаяся в выявлении факта внедрения скрытого сообщения. В некоторых работах, например [4], используется подход, применяющий статистический анализ для выявления нетипичного для подавляющего большинства исполняемых файлов набора команд, способов выделения регистров, большой избыточности кода или неиспользуемых ветвей программы (так называемый «мёртвый код»). В работе [3] предлагался метод стегоанализа, базирующийся на подходе, предложенном и развитом в ряде работ, например, [5−9]. Подход предполагает использовать архиватор для выявления случайности сообщения. В данной статье предлагается способ, позволяющий устранить отличительную особенность передаваемого секретного сообщения (передаваемое сообщение выглядит, как случайная последовательность), что позволит сделать неэффективным применяемый метод стегоанализа.

2. Описание метода и результаты

2.1. Описание схемы передачи секретного сообщения

Исходный метод внедрения информации, предложенный в работе [1], имеет существенный недостаток — различие статистических свойств кода программы и передаваемого секретного сообщения. Это различие может быть выявлено архиватором, как например, в работе [3]. Передаваемое сообщение должно быть предварительно зашифровано. Известно, что распределение вероятностей байт зашифрованного сообщения выглядит случайным. Код программы, наоборот, содержит часто повторяющиеся наборы двоичных инструкций, т.е. неслучайное распределение байт. Часть секции кода, в которую предположительно внедрено секретное сообщение, подвергается сжатию с помощью архиватора. Если сжатая последовательность в размерах не уменьшится, то это означает, что она случайная, то есть внедрено секретное сообщение. Очевидно, что если секретное сообщение перед внедрением будет закодировано так, что распределение вероятностей будет такое же, как у кода программы, то атаки подобного рода станут неэффективными. В данной статье предлагается использовать оптимальный код Хаффмана для кодирования зашифрованного сообщения перед внедрением. Приведём пример:

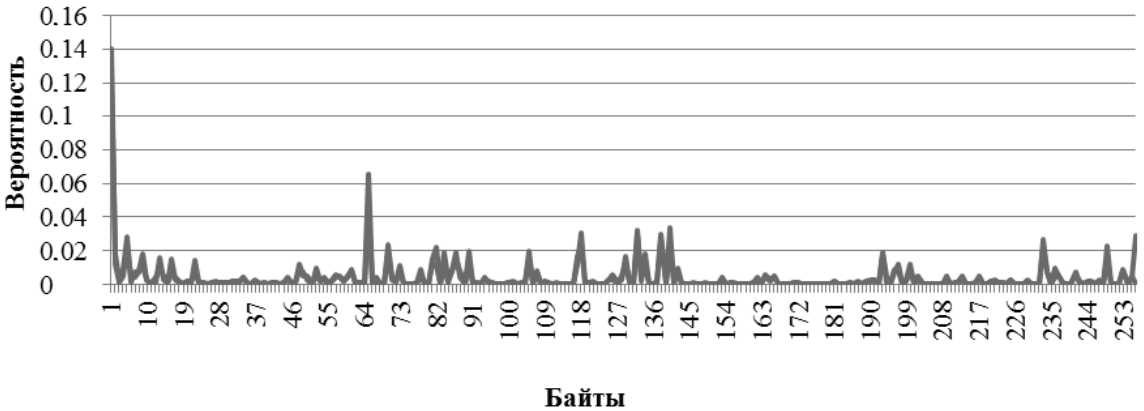

Рис. 1. Распределение вероятностей байт инструкций программы

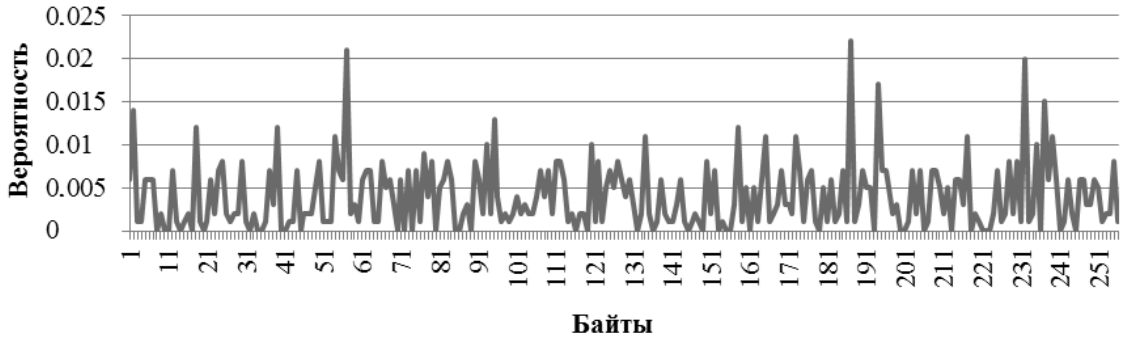

Рис. 2. Распределение вероятностей байт случайной последовательности до кодирования

Мы видим, что распределения существенно отличаются, что с лёгкостью выявляется архиватором.

Рис. 3. Распределение вероятностей байт случайной последовательности после кодирования

Однако после применения предлагаемого подхода, распределения на рис. 1 и 3 практически совпадают. Таким образом, повышается устойчивость к обнаружению метода внедрения сообщения.

Рассмотрим схему передачи секретного сообщения в общем виде, представленную на рис. 4.

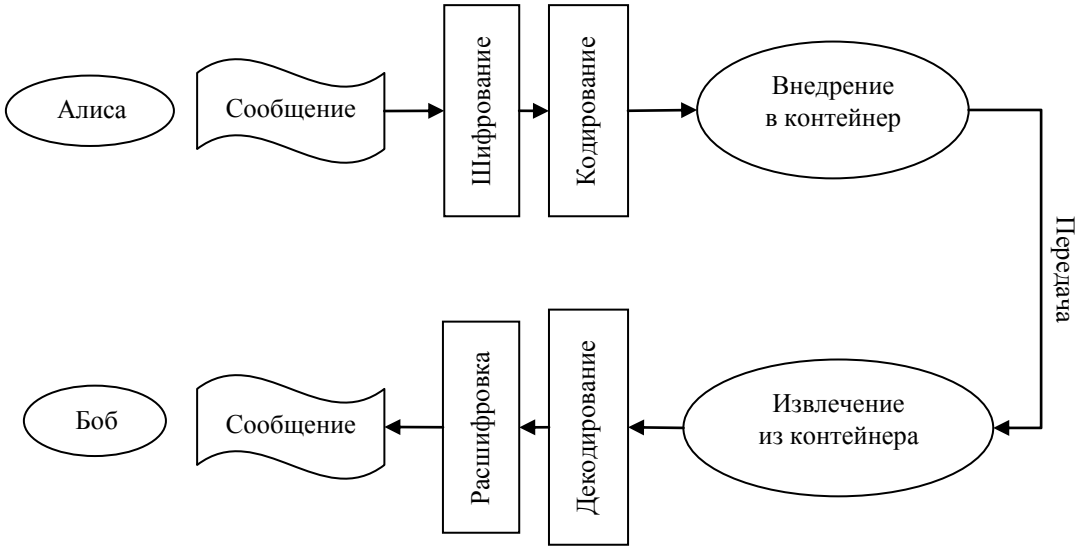

Рис. 4. Общая схема передачи секретного сообщения

Алиса передает Бобу секретное сообщение. Алиса совершает следующие шаги:
Шаг 1. Шифрует исходное сообщение, используя секретный ключ.
Шаг 2. Кодирует зашифрованное сообщение оптимальным кодом Хаффмана.
Шаг 3. Внедряет полученное сообщение в секцию кода программы.
Шаг 4. Передает файл по открытому каналу связи.
Затем Боб совершает следующие шаги:
Шаг 6. Принимает файл от Алисы.
Шаг 7. Извлекает сообщение из секции кода программы.
Шаг 8. Декодирует сообщение, используя оптимальный код Хаффмана.
Шаг 9. Расшифровывает сообщение, используя секретный ключ.

Таким образом, Боб получает от Алисы исходное сообщение. Участники обмена сообщениями заранее договариваются об используемом при шифровании секретном ключе. На шаге 7 Бобу необходимо знание начальной позиции в файле внедрённого сообщения. Об этом также следует договориться заранее. Для шагов 2 и 8 (кодирования / декодирования) используется оптимальный код Хаффмана. Процесс декодирования происходит аналогично.

2.2. Описание способа получения распределения вероятностей байт

Как уже отмечалось, для построения дерева Хаффмана требуется распределение вероятностей байт. В данной работе рассматривается два способа получения такого распределения, которые используется на этапе кодирования / декодирования передаваемого сообщения:

- распределение считается в передаваемом пустом контейнере;
- выбирается единственное распределение, которое используется во всех случаях обмена сообщениями.

Рассмотрим первый способ. На шаге 2 Алисе необходимо знать распределение вероятностей байт пустого контейнера. Так как изначально Алиса имеет пустой контейнер, то она без труда сможет его получить. На шаге 8 Бобу также нужно знать распределение вероятностей байт пустого контейнера. Боб получает уже заполненный контейнер, но так как известна начальная позиция встроенного сообщения, то он не будет учитывать байты внедрённого сообщения при получении этого распределения. Таким образом, Алиса и Боб могут получить распределение, необходимое на этапах кодирования и декодирования сообщения.

Проверим эффективность предлагаемого подхода экспериментально. Будем осуществлять внедрение закодированного секретного сообщения и применять стеготест, описанный в работе [3]. Для эксперимента была сформирована выборка случайным образом из 1000 исполняемых файлов. Далее, используя вышеописанную схему, заполним контейнеры сообщениями. Зашифрованное сообщение можно имитировать последовательностью, полученной из генератора случайных чисел. Обозначим размер секции кода файла – N. В дальнейшем анализируемую часть контейнера будем обозначать в виде: [*начало*;*конец*] (здесь указаны позиции начала и конца анализируемой части в секции).

После применения стеготеста получим результат, приведённый в табл. 1.

Таблица 1. Результаты стеготеста при использовании распределения вероятностей байт пустого контейнера

Анализируемая часть	Обнаружено, %
[0;N]	6.6

Как видно из табл. 1, в 66 контейнерах из 1000 сообщение было обнаружено несмотря на предварительное кодирование. Можно утверждать, что распределение вероятностей обнаруженных стеготестом контейнеров было близко (с точки зрения стегоанализа) к случайному. При изучении структуры исполняемых файлов было обнаружено, что секция кода исполняе-

мого файла может содержать не только набор двоичных инструкций, но и некоторый набор данных, например, таблицы адресов импорта. Обычно для таблицы адресов импорта используется дополнительная секция (например, для компиляторов фирмы *Borland* такая секция называется *.idata*). Но компиляторы сторонних производителей могут размещать её в кодовой секции и, как правило, она размещается в конце секции. Строго говоря, формат PE допускает размещение таблицы импорта в начале, середине и конце секции кода. Данные, расположенные в этой таблице, могут существенно влиять на получаемое распределение.

На рис. 5 и 6 показаны распределения средних вероятностей первой тысячи байт секции кода и последней тысячи, взятых из 1000 файлов.

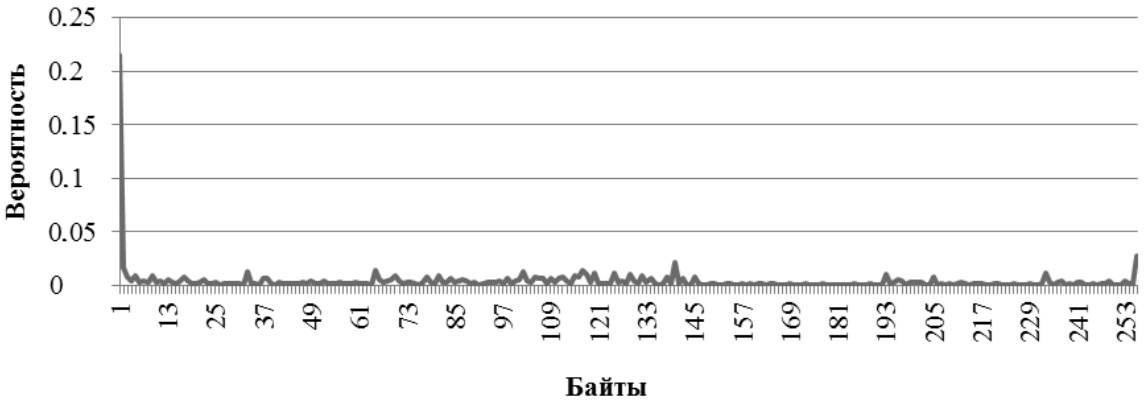

Рис. 5. Распределение средних вероятностей первой тысячи байт секции кода

Рис. 6. Распределение средних вероятностей последней тысячи байт секции кода

Из графиков видно, что распределения первых и последних байт секции кода действительно отличаются.

Таким образом, следует получать распределение вероятностей, анализируя не всю секцию, а только какую-то её часть. Далее будет рассмотрено, какая из частей секции должна анализироваться.

Рассмотрим результаты работы стеготеста [3] при использовании распределения вероятностей N последних байт секции кода, представленных в табл. 2.

Таблица 2. Результаты стеготеста при использовании распределения вероятностей байт первых N байт секции

Анализируемая часть N, байт	Обнаружено, %
80	0.0
300	0.2
500	0.1
1000	0.6
2000	1.5
3000	2.7
4000	2.0
5000	3.0

Как мы уже выяснили, распределения вероятностей первых и последних байт секции кода различаются. Теперь будем использовать для кодирования распределение вероятностей N первых байт (табл. 3).

Таблица 3. Результаты стеготеста при использовании распределения вероятностей байт последних N байт секции

Анализируемая часть N, байт	Обнаружено, %
80	0.0
300	0.1
500	0.2
1000	0.8
2000	0.8
3000	1.2
4000	1.7
5000	1.8

Итак, из табл. 2 и 3 мы видим, что для получения распределения лучше всего анализировать 80 первых или последних байт секции кода.

Рассмотрим второй способ получения распределения вероятностей. При получении распределения следует учитывать два требования:

- распределение является «типичным» для большинства файлов (очевидно, что любое нетипичное для большинства файлов распределение может вызывать подозрение);
- распределение позволяет успешно проходить стеготест.

Для получения распределения, наиболее «типичного» для всех файлов, были проанализированы 1000 файлов. Для каждого из них были получены распределения вероятностей первых 1000 байт. Далее было выбрано одно распределение, наиболее близкое ко всем остальным. Для этого составлялась матрица, элементами которой являются расстояния между распределениями. Под расстоянием понимается расстояние Кульбака − Лейблера

$$D(p,q) = \sum_{x=0}^{255} p(x) \ln \frac{p(x)}{q(x)}. \quad (1)$$

На рис. 7 изображена матрица, содержащая расстояния между двумя распределениями. Для того чтобы выбрать искомое распределение, суммировались элементы строк матрицы (как показано на рис. 7) и выбиралось то распределение, которое давало минимальную

№	1	2	…	1000	Сумма
1	1	10	…	31	1+10+…+31
2	10	1	…	25	10+1+…+25
…	…	…	…	…	…

Рис. 7. Матрица расстояний

сумму по соответствующей строке матрицы. Таким образом было получено следующее распределение:

Рис. 8. Выбранное распределение вероятностей байт

Будем использовать при кодировании полученное распределение. Рассмотрим результаты стеготеста, представленные в табл. 4.

Таблица 4. Результаты стеготеста при использовании выбранного распределения вероятностей байт

Число заполненных контейнеров	Обнаружено, %
1000	0

Результаты стеготеста показывают, что при данном способе выбора распределения вероятностей стеготест не обнаруживает ни один контейнер.

3. Заключение

Целью данной работы была модификация существующего метода внедрения информации в исполняемый файл. Было предложено перед внедрением сообщения в контейнер предварительно его кодировать. Тогда распределение вероятностей байт внедряемого сообщения будет неотличимо от распределения кода программы, что делает стеготест [3] неэффективным. Было установлено, что на этапе кодирования сообщения для построения дерева Хаффмана следует использовать следующие распределения вероятностей:

- распределение вероятностей первых 80 байт секции кода;
- распределение вероятностей последних 80 байт секции кода;
- распределение вероятностей, показанное на рис. 8.

Так как некоторые компиляторы размещают таблицу импорта в конце секции кода, то наличие ненулевых байт после таблицы может вызывать подозрение. Поскольку задача перемещения таблицы импорта после внедрённого сообщения достаточно трудоёмкая и требующая дезассемблирования (производимого в полуавтоматическом режиме с участием человека), то рекомендуется не использовать такие контейнеры. Для передачи секретного сообщения лучше подбирать контейнеры, у которых таблица располагается в отдельной секции.

Литература

1. Thorpe D. Development System with Methodology Providing Information Hiding in Executable Program // US Patent 20060136875 (2006).
2. Pietrek M. Peering Inside the PE: A Tour of the Win32 Portable Executable File Format // URL: `http://msdn.microsoft.com/enus/magazine/ cc301805.aspx` (дата обращения: 01.11.2010).
3. Нечта И.В. Эффективный метод стегоанализа, базирующийся на коде Хаффмана // Вестник СибГУТИ 2010. № 4. С. 47-53.
4. Anckaert B., De Sutter B., Chanet D., De Bosschere K. Steganography for Executables and Code Transformation Signatures // 7th International Conference on Information Security and Cryptology. 2005. P. 425–439.
5. Ryabko B., Monarev V. Using information theory approach to randomness testing // Journal of Statistical Planning and Inference. 2005. V. 133, № 1. P. 95–110.
6. Ryabko B., Monarev V. Experimental investigation of forecasting methods based on data compression algorithms // Problems of Information Transmission. 2005. V. 41, № 1. P. 65–69.
7. Ryabko B., Astola J. Universal codes as a basis for time series testing // Statistical Methodology. 2006. V. 3. P. 375–397.
8. Ryabko B., Astola J. Universal codes as a basis for nonparametric testing of serial independence for time series // Journal of Statistical Planning and Inference. 2006. V. 136, № 12. P. 4119–4128.
9. Ryabko B. Compression-based methods for nonparametric density estimation, on-line prediction, regression and classification for time series // 2008 IEEE Information Theory Workshop. Porto, Portugal. 2008.

Статья поступила в редакцию 01.04.2011;
переработанный вариант —24.05.2011

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

Method of Secret Messages Embedding into Executable Files

I. Nechta

This paper is about modification of data hiding method in executable files. The previous embedding method hides a secret message in an unused space of the executable file. Statistical difference exists between a program code and an embedded message, which makes this embedding method vulnerable to statistical attacks. This paper offers to encode a message before embedding so that the probability distribution of secret message bytes and the program code bytes were indistinguishable. The proposed approach improves robustness of data hiding method to statistical attacks.

Keywords: steganography, steganalysis, Portable Executable.