УДК 003.26

Эффективный метод стегоанализа исполняемых файлов, базирующийся на коде Хаффмана

И.В. Нечта

В статье предлагается эффективный метод обнаружения внедрения информации в специальные байты выравнивания секций файлов формата Portable Executable. При стегоанализе используются различия статистических свойств кода и встраиваемого «скрытого» сообщения. Экспериментально получены данные о высокой вероятности обнаружения внедрённого сообщения.

Ключевые слова: Стеганография, стегоанализ, Portable Executable.

1. Введение

Цель стеганографии состоит в организации передачи секретных данных так, чтобы сам факт передачи был скрыт от стороннего наблюдателя. Обычно в «невинное» сообщение, так называемый контейнер, с помощью специального алгоритма встраивается стегосообщение. Контейнер подбирается таким образом, чтобы содержание и сам факт его передачи не вызывал никакого подозрения у третьих лиц. Широкое применение стеганография получила в сфере защиты авторских прав. В каждую продаваемую копию цифрового объекта (изображение, видео, программа) с помощью специальных алгоритмов внедряется специальное 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. Очевидно, что внедряемый водяной знак должен обладать некоторой степенью устойчивости, то есть противостоять удалению или модификации. Считается, что заранее можно предугадать, содержится ли водяной знак или нет. В настоящее время существуют методы, позволяющие защитить подобным образом от нелегального копирования различные цифровые объекты, такие как файлы мультимедиа и программ. В частности, ряд публикаций, например [1 – 11], посвящён методам внедрения скрытой информации в программы. В данной статье речь пойдёт о методе, размещающем секретное сообщение в неиспользуемых местах секции исполняемых файлов формата Portable Executable (PE) [1] . Каждая секция такого файла должна быть размером, кратным полю `FileAligment` (см. документацию Microsoft [12]). Таким образом, секция кода состоит из двоичных инструкций программы и нулевых байтов выравнивания, увеличивающих секцию до требуемых размеров. Авторы работы [11] предлагают вместо байтов выравнивания записывать секретное сообщение. По их мнению, это имеет некоторые преимущества: размер файла

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

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, как будет показано ниже, данный подход не является устойчивым из-за различных статистических свойств кода программы и передаваемого зашифрованного сообщения.

Одним из направлений стеганографии является стегоанализ, задача которого заключается в выявлении внедрения скрытого сообщения. Во многих работах, например [5], 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 (т.н. мёртвый код). В этой статье предлагается новый метод стегоанализа, базирующийся на подходе, предложенном и развитом в ряде работ Б. Я. Рябко с соавторами [13–17]. Подход предполагает использовать архиватор для выявления случайности сообщения. Известно, что зашифрованное сообщение выглядит как псевдослучайная последовательность. А это означает, что при сжатии архиватором такой последовательности её размер более уменьшаться не может. Программа, наоборот, имеет часто повторяющиеся наборы двоичных инструкций. Архиватор такие повторения обнаружит, следовательно, это даст уменьшение размера при сжатии. На этой особенности и основан предлагаемый метод стегоанализа. Стоит отметить, что для выявления случайности сообщения можно использовать не только архиваторы, но и любой другой код, применяющийся при сжатии данных. В настоящей статье предлагается использовать код Хаффмана.

Экспериментально получены данные о высокой вероятности обнаружения внедрённого сообщения.

2. Описание предлагаемого метода стегоанализа

Рассмотрим подробно этапы работы метода:

1. Извлекаем исполняемую секцию из файла программы.
2. Удаляем байты выравнивания, если таковые присутствуют. Такая ситуация может возникнуть, если лишь часть допустимого объёма внедрения была использована.
3. Применяем код Хаффмана для последних W байт секции. О выборе величины W (параметра метода) подробнее напишем ниже.
4. По длине полученного кода определяем наличие или отсутствие скрытого сообщения.

Этап извлечения секции кода требует определённый уровень знания PE формата. Для начала необходимо найти смещение заголовка PE. Одно из полей заголовка — `Section headers` указывает на массив заголовков секций, каждый элемент которого описывает набор атрибутов секций файла, таких как адрес начала, размер и характеристика секции. Определять, является ли секция кодовой (т.е. искомой), необходимо по значению бит характеристики. Второй этап сводится к тривиальному удалению серии нулевых байт, завершающих секцию. Третий этап основывается на особенности исходного метода внедрения и заключается в том, что запись сообщения осуществляется в конец секции, непосредственно после кода программы. Следовательно, должен производиться анализ этих последних байтов (предположительное место размещения скрытого сообщения). Для простоты эту анализируемую часть секции мы будем называть “окном”.

Теперь необходимо определить размер окна (величина W). Здесь следует стремиться к уменьшению его размера по следующим причинам. Рассмотрим два рисунка. На рис. 3 показана ситуация выбора большого размера окна (заштрихованная область). В этом случае в анализируемую выборку попадет и скрытое сообщение, и часть кода программы. Закодированное сообщение в размерах практически не уменьшится, а код, из-за частого повторения одинаковых инструкций, сожмётся значительно лучше. Следовательно,

в результате анализа таких разнородных данных нельзя однозначно определить, какими статистическими свойствами (зашифрованного сообщения или программы) обладает содержимое анализируемого окна, а это приведёт к увеличению числа ошибок стегоанализа. Если выбрать маленький размер окна, как показано на рис. 4, то в таком случае анализируемые байты будут состоять только из секретного сообщения. Данные будут однородными, а это означает, что наш метод стегоанализа будет работать более точно.

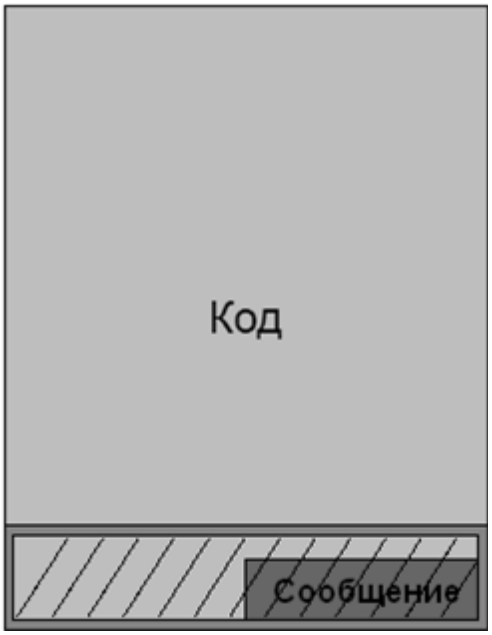

Рис. 3. Выбор большого окна

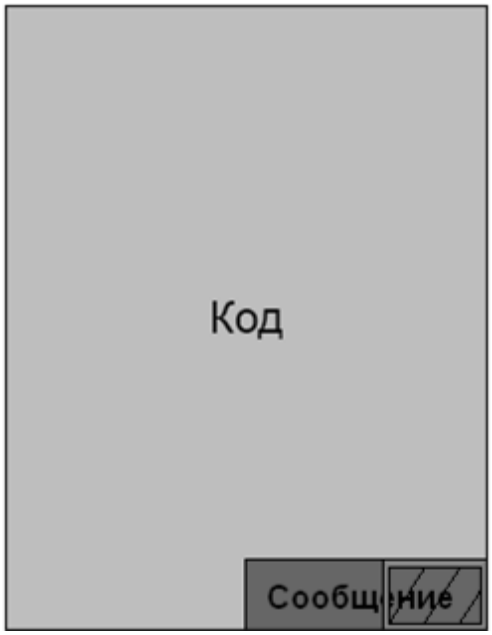

Рис. 4. Выбор маленького окна

Но в тоже время очевидно, что чем меньше размер окна, тем меньше повторяющихся инструкций может оно содержать и, как следствие, код программы будет иметь меньше статистических отличий от зашифрованного сообщения. Оптимальный размер окна будет подбираться эмпирически в разделе 4. На четвёртом этапе по полученной длине кода можно определить, принадлежит ли анализируемая последовательность к коду программы или к зашифрованному сообщению.

3. Выбор архиватора

В этой главе мы проанализируем работу различных архиваторов и выберем среди них наиболее подходящий для нашего метода.

Итак, произведём сравнение следующих архиваторов: `Bzip`, `Bzip2`, `Zip`, `Rar`, `Paq8`, `Nanozip`. Нам нужно выбрать такой архиватор, у которого размеры сжатых фрагментов программ и зашифрованных сообщений будут различаться. Исходя из того, что зашифрованное сообщение выглядит как равновероятная последовательность байт, мы будем имитировать его последовательностью, полученной с помощью генератора случайных чисел. Для этого возьмём по 1000 фрагментов размером 80 байт (причины выбора именно такого размера подробнее будут рассмотрены в следующем разделе) псевдослучайной последовательности и кода соответственно. Далее произведём сжатие фрагментов архиватором. Получившиеся длины сжатых фрагментов будем представлять в виде интервалов. Строго говоря, для каждого архиватора мы получим два интервала:

- интервал длин сжатых фрагментов псевдослучайной последовательности;
- интервал длин сжатых фрагментов кода.

Для нас, в лучшем случае, эти два интервала не будут пересекаться. Следовательно, при стегоанализе по размеру анализируемого фрагмента можно однозначно определить, к чему он относится: к коду или к секретному сообщению.

Таблица 1. Интервалы размеров фрагментов

Архиватор	Интервалы		Число пересекающихся элементов
	Сл. последов.	Код	
Bzip2	[141;159]	[97;142]	2
Paq8	[112;126]	[75;126]	339
Rar	[148;151]	[128;151]	823
Nanozip	[132;135]	[113;135]	864
Bzip	[109;112]	[94;112]	959
Zip	[188;194]	[180;194]	994

Под числом пересекающихся элементов понимается количество сжатых фрагментов, размеры которых попали в оба интервала (т.е. для них невозможно однозначно определить – это двоичные инструкции программы или псевдослучайная последовательность). Из таблицы видно, что наиболее подходящим архиватором для стегоанализа является `Bzip2` (использующий преобразование Барроуза – Уилера и код Хаффмана), т.к. дает минимальное количество пересечений. Как уже отмечалось ранее, вместо архиваторов можно использовать код Хаффмана. Для этого предварительно рассчитываются оценки вероятностей появления каждого символа в окне. В качестве символа мы будем брать один байт. Рассмотрим подробнее процесс получения оценок вероятностей. Пусть имеется алфавит $A=\{a_1,a_2,\ldots,a_N\}$, его мощность $|A|=N$, где $N=256$. Будем говорить, что анализируемое окно состоит из символов алфавита A. Причём символ a_i есть байт, равный $i-1$. Введём множество счётчиков $C=\{c_1,c_2,\ldots,c_N\}$. Каждый элемент этого множества содержит число повторений соответствующего символа в окне. Следовательно, искомые оценки вероятностей $P=\{p_1,p_2,\ldots,p_N\}$ появления каждого символа в окне можно вычислить по формуле:

$$p_i=\frac{c_i}{W}.$$

Далее по этим оценкам вероятностей строится дерево Хаффмана. Содержимое окна кодируется, и по длине получившегося кода определяется наличие факта внедрения. Рассмотрим интервалы для кода Хаффмана на той же выборке, что и для вышеописанных архиваторов.

Таблица 2. Интервалы размеров фрагментов

Метод кодирования	Интервалы		Число пересекающихся элементов
	Сл. последов.	Код	
Хаффмана	[58;64]	[19;53]	0

Как видно из таблицы, в отличие от архиватора `Bzip2`, интервалы не пересекаются, следовательно, возможно однозначное определение факта внедрения скрытого сообщения.

Таким образом, эмпирически было показано, что для нашего метода стегоанализа следует использовать код Хаффмана.

4. Выбор оптимального размера окна

Для поиска минимально допустимого значения W (размера окна) была сформирована случайная выборка по 1000 фрагментов программ и псевдослучайных последовательностей соответственно. Производилось кодирование методом Хаффмана при различных значениях W (от 4 Кб до 40 байт). Ниже на диаграммах приведены размеры закодированных фрагментов при $W = 80$.

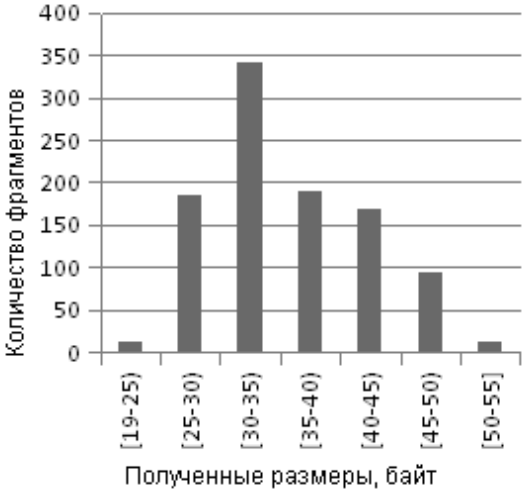

Рис. 5. Диаграмма размеров фрагментов программ $W = 80$

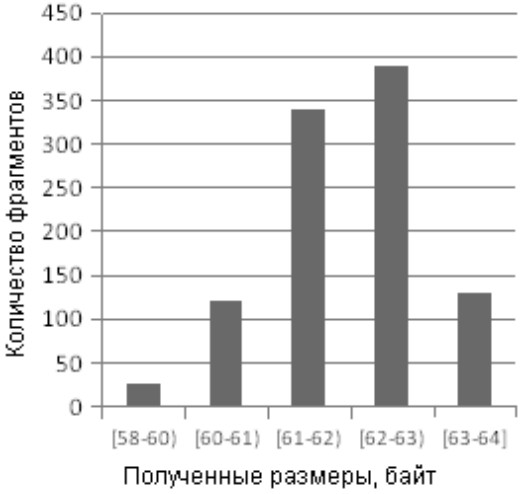

Рис. 6. Диаграмма размеров фрагментов псевдослучайной последовательности $W = 80$

Мы видим, что получены два интервала размеров: [19;55] и [58;64] для фрагментов кода и случайной последовательности соответственно. Поскольку интервалы не пересекаются, то возможно однозначное определение факта внедрения стегосообщения. Как уже упоминалось ранее, необходимо стремиться к уменьшению W. Рассмотрим диаграммы при $W = 70$ байт.

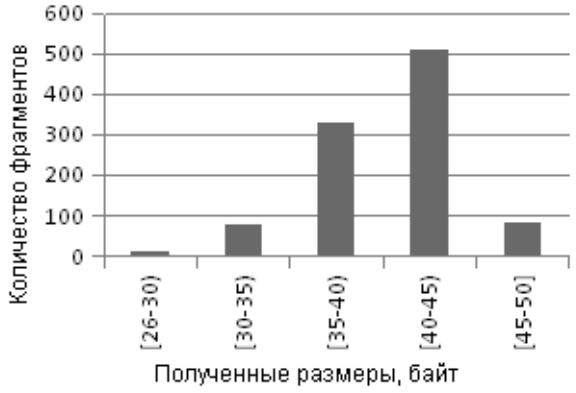

Рис. 7. Диаграмма размеров фрагментов программ $W = 70$

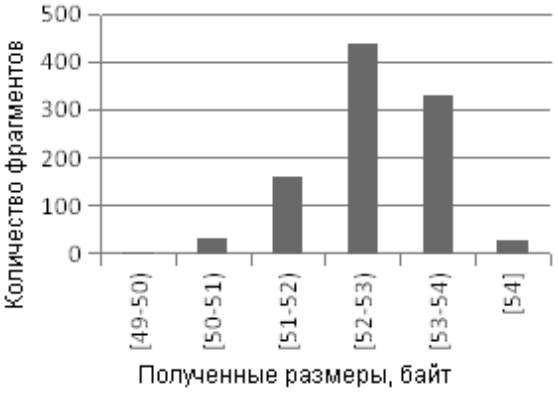

Рис. 8. Диаграмма размеров фрагментов псевдослучайной последовательности $W = 70$

Из диаграмм (рис. 7, 8) видно, что происходит пересечение интервалов длин. Следовательно, оптимальное значение W составляет 80 байт.

Итак, мы установили, что для эффективного стегоанализа необходимо анализировать последние 80 байт контейнера (не учитывая байты выравнивания секции). Если размер сжатого фрагмента окажется больше 56, то контейнер будет считаться заполненным стегосообщением. В остальных случаях — пустым.

5. Экспериментальный анализ

После того как был определён параметр метода *W*, а также обоснована целесообразность использования кода Хаффмана вместо архиваторов, определим эффективность нового метода экспериментально. В качестве критерия эффективности будем использовать отношение количества правильно определённых фактов наличия или отсутствия внедрения скрытого сообщения к общему количеству определений. Здесь необходимо различать два вида ошибок:

- ошибка первого рода — ситуация, при которой пустой контейнер признаётся заполненным;
- ошибка второго рода — ситуация, при которой заполненный контейнер признаётся пустым.

Для эксперимента была случайно произведена выборка из 1000 файлов формата PE (программ).

Рассмотрим результаты работы метода, если контейнер не заполнен.

Таблица 3. Эффективность работы метода на пустом контейнере

Внедрено, байт	Количество правильных определений
0	1000

Таким образом, ошибка первого рода на данной выборке отсутствует.

Далее будем постепенно заполнять контейнер на различное количество байт и анализировать результаты работы метода. Как уже отмечалось ранее, мы будем имитировать стегосообщение внедрением в контейнер псевдослучайной последовательности, полученной из генератора случайных чисел. На следующей диаграмме представлена зависимость количества правильных определений от размера внедрения.

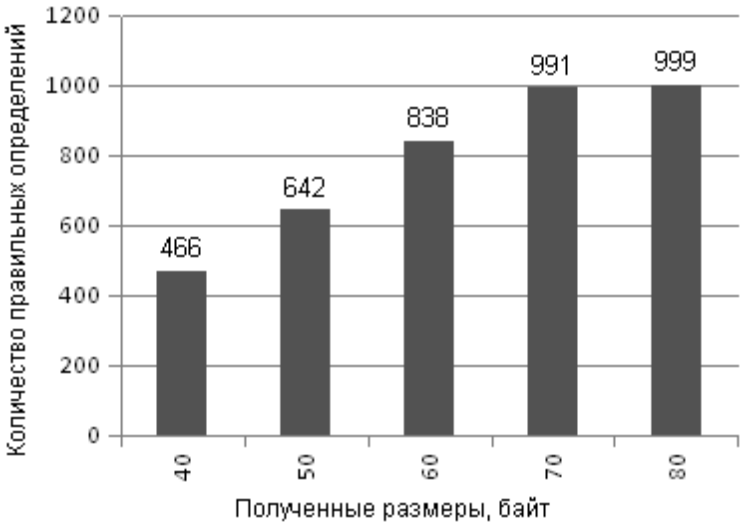

Рис. 9. Результаты работы метода на заполненном контейнере

Ранее отмечалось, что наш метод рассчитан на работу с контейнером, заполненным на 80 байт. Если контейнер заполнен менее чем на 70 байт, то ошибка второго рода достаточно велика, что делает затруднительным практическое применение метода. Причина столь высокой ошибки кроется в разнородности анализируемых данных и была описана выше. Однако уже при заполнении контейнера на 70 байт и более метод стегоанализа достаточно эффективно обнаруживает внедрение. То есть разнородность данных ощутимо не влияет.

6. Заключение

Итак, в данной работе была продемонстрирована нестойкость метода внедрения информации в исполняемые файлы формата Portable Executable, предложенного в докладе [11]. Также был предложен новый эффективный метод стегоанализа, определяющий внедрение скрытого сообщения размером 70 байт и более.

Список литературы

1. Hamilton J., Danicic S. An Evaluation of Static Java Bytecode Watermarking // ICCSA'10 “The World Congress on Engineering and Computer Science”. San Francisco. 2010.
2. Нечта И.В., Фионов А.Н. Цифровые водяные знаки в программах на C\C++ // Материалы XI международной научно-практической конференции “Информационная безопасность” Таганрог: Изд. ТТИ ЮФУ. 2010. Ч. 3. С. 108-113.
3. Davidson R., Myhrvold N. Method and system for generating and auditing a signature for a computer program // US Patent 5559884. 1996.
4. El-Khalil R., Keromytis A. Hydan: hiding information in program binaries // VI International Conference “Information and Communications Security”. Berlin: Springer. 2004. V. 3269. P. 187-199.
5. Blascol J., Hernandez-Castol J. Steganalysis of Hydan // IFIP Advances in Information and Communication Technology. 2009. V. 297/2009, P. 132-142.
6. Hattanda K., Ichikawa S. The Evaluation of Davidson's Digital Signature Scheme // I EICE transactions on Fundamentals of Electronics, Communications and Computer Sciences. 2004. V. E87-A. №.1. P. 224-225.
7. Naji A., Teddy S. Algorithms to Watermark Software Through Register Allocation // Lecture Notes in Computer Science, 2006. V. 3919/2006, P. 180-191.
8. Нечта И.В. Стеганография в файлах формата Portable Executable // Вестник СибГУТИ. 2009. №1. С. 85-89.
9. Naji A., Teddy S. New Approach of Hidden Data in the portable Executable File without Change the Size of Carrier File Using Statistical Technique // International Journal of Computer Science and Network Security. 2009. V. 9. № 7. P. 218-224.
10. Zaidan A., Zaidan B., Jalab H. A. A New System for Hiding Data within (Unused Area Two + Image Page) of Portable Executable File using Statistical Technique and Advance Encryption Standard // International Journal of Computer Science and Network Security. 2010. V. 2. № 2.
11. Shin D., Kim Y., Byun K., Lee S. Data Hiding in Windows Executable Files // Australian Digital Forensics Conference. 2008. P. 51.
12. 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).
13. Ryabko B., Monarev V. Using information theory approach to randomness testing // Journal of Statistical Planning and Inference. 2005. V. 133, №1. P. 95–110.
14. 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.
15. Ryabko B., Astola J. Universal codes as a basis for time series testing // Statistical Methodology. 2006. V. 3. P. 375–397.
16. 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.

17. 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 5.11.2010;
переработанный вариант —15.12.2010

Нечта Иван Васильевич
аспирант, ассистент кафедры прикладной математики и кибернетики СибГУТИ,
тел. 8-923-11-38-992, e-mail: www@inbox.ru

Effective method of steganalysis of executable files based on Huffman code

I. Nechta

Methods of data hiding in executable files (programs) are considered. Steganalysis of method which uses blank spaces in executable files is carried out. It is shown experimentally that the hidden data are detected with high confidence.

Keywords: steganography, steganalysis, Portable Executable.