УДК 004.056.52

Анализ существующих методов защиты авторского права на программное обеспечение средствами стеганографии

Н.И. Полуденный[*1], А.В. Чернышова[*2]

[*1] магистрант, Донецкий национальный технический университет,
nik.poludennyy@mail.ru

[*2] старший преподаватель, Донецкий национальный технический университет,
chernyshova.alla@rambler.ru, OrcID: 0000-0003-2546-2167, SPIN-код: 3318-2066

Полуденный Н.И., Чернышова А.В. Анализ существующих методов защиты авторского права на программное обеспечение средствами стеганографии. *Целью работы является сравнение методов сокрытия цифрового водяного знака в программном обеспечении, выделение их преимуществ и недостатков для дальнейшего поиска возможностей устранения недостатков и повышения эффективности алгоритмов защиты авторского права на программное обеспечение.*

Ключевые слова: *стеганография, защита, авторское право, программное обеспечение, цифровой водяной знак, цифровой отпечаток.*

Введение

Авторское право — это право интеллектуальной собственности. Каждый вложивший (автор) в создание чего-либо (объекта права) время, силы и другие ресурсы хочет защитить своё творение от всякого рода посягательств на свою интеллектуальную собственность со стороны третьих лиц (субъектов права). И если защита права владения физическим объектом может быть подкреплена юридически с помощью документов, то программное обеспечение – это, по сути, набор цифровой информации, а информация – объект абстрактный, и при извлечении фрагмента кода ПО и использовании его в другом программном обеспечении очень сложно проконтролировать и пресечь. Поэтому сегодня активно развиваются методы защиты авторского права на программный продукт.

Система защиты от копирования или система защиты авторских прав — комплекс средств, обеспечивающих затруднение или запрещение нелегального распространения, использования и/или изменения программных продуктов.

Под нелегальным распространением понимается продажа, обмен или бесплатное распространение программного продукта, авторские права на который принадлежат третьему лицу, без его согласия.

Нелегальное использование — использование программного продукта без согласия владельца авторских прав.

Сегодня проблема авторского права стоит очень остро, так как программные продукты, являются, по сути, информацией, а потому не могут быть защищены от кражи такими же методами как физические объекты. Стеганографические методы позволяют внедрять информацию, доказывающую авторство, но при этом не посвящать третьих лиц о существовании таковой.

Эффективной считается защита, для взлома которой требуется больше ресурсов, чем на приобретение копии продукта.

Актуальность темы

Чаще всего цифровую стеганографию связывают с файлами медиа-формата: изображениями, видеоматериалом и звуковыми файлами, реже с текстом и ещё реже с кодом программного обеспечения. Это связано со сложностью реализации алгоритма встраивания сообщения в эти контейнеры. Медиа-файлы предназначены для восприятия органами чувств человека, причём здесь работает восприятие образов, а не отдельных элементов, поэтому информацию можно легко встроить незаметно для стороннего наблюдателя, и даже компьютер не всегда сможет определить присутствие сообщения. С текстовыми файлами сложнее, так как здесь уже человек различает отдельные элементы (буквы и символы), поэтому реализовать алгоритм встраивания стегосообщения в цифровой текст сложнее. Код программного обеспечения предназначен для «понимания» его компьютером, а потому малейшие изменения в нём могут лишить компьютер возможности выполнить операции, предписанные разработчиком. По этой причине встраивание стегосообщений в программное обеспечение не нашло широкого распространения.

Защита программного обеспечения с помощью цифровых водяных знаков (ЦВЗ) имеет смысл если

продукт, по поводу которого могут возникнуть споры, выпускается небольшим тиражом (программное обеспечение, сделанное под заказ). Для программного обеспечения для широкого круга пользователей более эффективным решением будет предоставление услуг приложения через облачные сервисы, предоставляя пользователям только тонкий клиент, однако, такой способ более затратный, так как помимо затрат на разработку ПО, требуются затраты на сопровождения сервиса. Поэтому метод защиты ПО с помощью цифровых водяных знаков до сих пор является компромиссом между стоимостью и эффективностью.

Виды атак на цифровые водяные знаки

Атаки на водяные знаки в программном обеспечении разделяют на четыре основные категории:

- субтрактивные атаки (subtractive attacks);
- искажающие атаки (distortive attacks);
- аддитивные атаки (additive attacks);
- атаки по сговору (collusion attacks).

Субтрактивные атаки подразумевают удаление метки, таким образом делая невозможной процедуру её распознавания, и лишая возможности создателя доказать свою причастность к созданию спорного ПО.

Искажающие атаки, как можно понять из названия, нацелены на искажение метки, что в итоге приведёт к невозможности верной идентификации метки, и, как следствие, доказательства авторства.

Аддитивные атаки направлены на добавление дополнительной метки в спорный материал. Этот вид атаки весьма эффективен, если для создателя сложно доказать, что его метка была внедрена раньше метки злоумышленника.

Атакам по сговору подвержены продукты, которые выпускаются в количестве более одной копии, и в каждую из них встраивается индивидуальное сообщение, идентифицирующее каждую из копий. Злоумышленники имея две или более копии ПО, которое подвергается атаке, и сравнив исполнимый код нескольких копий могут обнаружить разницу, которая, с определённой долей вероятности, и есть метка, обнаружив которую, злоумышленник сможет удалить или исказить.

Обзор литературы по теме

В рассмотренной книге А. Щербакова «Защита от копирования» [1], изданной в 1992 г. рассматриваются наиболее распространённые методы защиты ПО, применявшиеся 30 лет назад. Все приведённые в книге методы завязаны на защите от копирования ПО с лицензионного носителя, дизассемблирования ПО.

На сегодняшний день защита от копирования ПО с физических магнитных и оптических носителей утратила свою актуальность, в связи с переходом большого количества ПО на цифровую дистрибуцию, а методы защиты от отладки и трассировки ПО, описанные в этой книге, актуальны для процессоров семейства x86, потому требуют модификации для семейства x64, но всё же принципы их работы остаются актуальными и теперь.

Полезной оказалась глава 6 «Выполнение защищённых программ и работа с защищёнными данными», в которой были описаны принципы построения внешнего загрузчика исполняемых модулей, которые производят некую проверку условия запуска и работы программного обеспечения. Это поможет сделать загрузчик, которых не даст работать ПО если, к примеру, метка была обнаружена и модифицирована.

В статье «Peering Inside the PE»[2] описывается структура PE-файла. Это Portable Executable файл – основной тип исполнимых файлов в операционных системах семейства Windows. Здесь поэтапно объясняется процесс запуска исполнимых файлов, назначения и структура заголовков, принцип выравнивания в памяти. Это даёт представление о возможности встраивании меток в заполненные нулевыми значениями в результате выравнивания секциями. Из структуры заголовков можно узнать правила адресации заголовков и секторов для того, чтобы избежать конфликтов при встраивании метки.

В связке с методами построения внешнего загрузчика это поможет скрыть содержимое метки (и кода программы в целом) от конечного пользователя.

В учебном пособии Монахова М.Ю. и Ташмухамедовой В.Ф. «Защита авторских прав» [3] помимо описания защиты ПО с юридической стороны, также описываются основы технических методов и средств защиты авторских прав на программное обеспечение. Как и во многих других материалах, посвящённых техническим методам защиты ПО приводятся принципы шифрования кода программы при хранении его на локальном носителе, и расшифровка кода только на момент его исполнения. Здесь описаны наиболее распространённые методы вскрытия защиты, также методы противодействия им, что весьма полезно для защиты внедрённой информационной метки. Как и в книге А. Щербакова «Защита от копирования», здесь говорится, что отладчики используют стандартные прерывания, поэтому, имея идентификаторы этих прерываний, можно изменить их векторы, и тем самым усложнить процесс отладки злоумышленниками. Здесь описаны методы внедрения метки, базирующиеся на особенностях структуры PE-файлов, описанной в статье «Peering Inside the PE».

Середа С.А. в своей работе на тему «Оценка эффективности защиты программного обеспечения»[4]

выделяет два типа средств защиты ПО: упаковщики/шифраторы; СЗ от несанкционированного копирования и несанкционированного доступа.

Первые изначально использовались для компрессии данных и уменьшения объёма исполняемого модуля, но, как говорит сам автор, позднее на первый план вышла цель защиты ПО от анализа его алгоритмов и несанкционированной модификации. Такой приём может также быть использован для сохранения целостности метки в исполнимом модуле.

Однако, автор указывает и на недостатки такого метода, самый значимый из которых - это то, что упаковка и шифрование исполняемого кода вступает в конфликт с запретом на самомодификацию кода в современных ОС.

Второй тип средств подразумевает «привязку» ПО к дистрибутивному носителю для усложнения процесса копирования ПО и потенциальной невозможности его запуска без наличия носителя. Как говорилось выше, такой метод неактуален ввиду распространения цифровой дистрибуции ПО.

В целях усложнения копирования и запуска ПО на нескольких машинах многими авторами, как возможная мера для аутентификации ПК, предлагается использовать некие неизменяемые или редко изменяемые параметры. К примеру, это может быть идентификатор жесткого диска или центрального процессора. Но проблема в том, что открытая архитектура ПК фактически «обезличивает» каждую IBM-PC-совместимую машину и, во-первых, такая защита может сработать неверно и не дать доступ к работе пользователю, который легально получил данную копию ПО, но изменил значимые для аутентификации аппаратные узлы, а во-вторых, может, наоборот, быть «обманута», если запрос на данные об аппаратной конфигурации будут перехвачены, извлечены и вместо реальных данных в ответ на запрос будут отправлены данные, удовлетворяющие проверку разрешения доступа.

Из графика, изображённом на рисунке 1 видно, что в иностранной литературе термин «стеганография» встречается чаще, особенно в литературе, изданной в США, на английском языке. В начале 2000-х годов первенство держала немецкая литература.

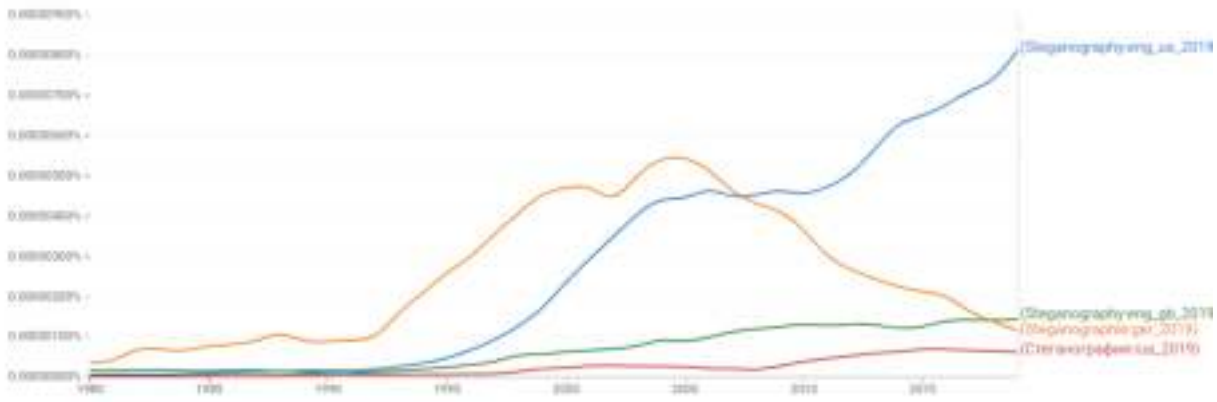

Рисунок 1 – Динамика употребления термина «стеганография» в литературе

Если же сравнивать частоту вхождения терминов «цифровая» и «стеганография», то количество американской литературы в начале 2000-х и на протяжении 10-ти годов XXI века вырывается вперёд и продолжает удерживать лидирующие позиции (см. рис. 2).

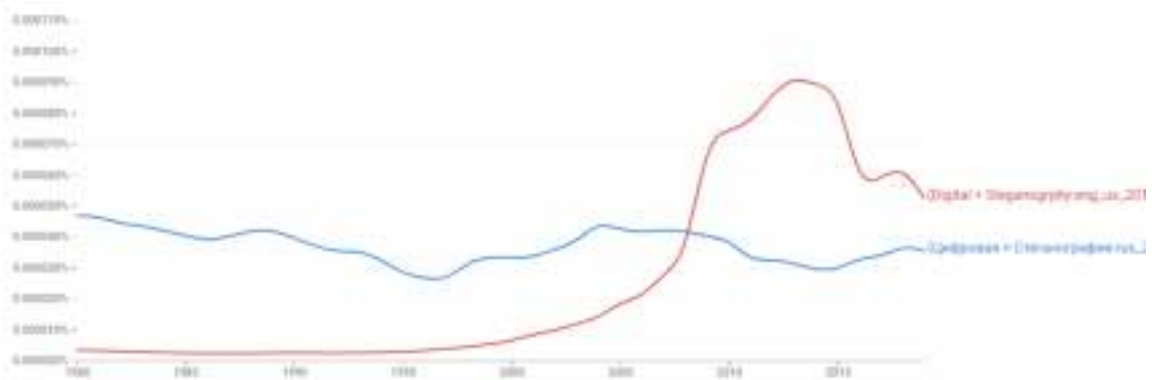

Рисунок 2 – Динамика употребления терминов «цифровая» и «стеганография» в литературе

Сравнение стеганографических методов внедрения метки в программный продукт

Способ внедрения метки зависит, во-первых, от типа и формата продукта, во-вторых, от возможности скрыть информацию в продукте определённого формата.

Одним из способов внедрения метки может быть внедрение некой информации на этапе реализации программного кода продукта, например, прописать некий не документированный функционал к программному продукту. Допустим, автор приложения сделал так, чтобы выводилась метка при вводе команды, которая не была описана в документации приложения.

Преимущества:

лёгкая реализация такого метода внедрения цифровых водяных знаков и цифровых отпечатков (ЦО).

Недостатки:

не универсально, так как для различных программных продуктов может быть различный интерфейс;

без защиты от отладки и дисассемблирования метка может быть обнаружена и удалена/модифицирована.

На сегодняшний день одним из самых распространённых расширений исполнимых файлов является EXE. Их структура не привязана к языку программирования, на котором был написан программный продукт. Поэтому внедрение меток в объектные коды имеет большую универсальность и независимость от программного интерфейса.

Один из таких способов основан на наличии свободных участков в объектных кодах программ, хранящихся в исполнимых файлах, там могут содержаться полностью или частично свободные секторы файла. Данный метод опирается на то, что секции в Portable Executable(PE) файле после процедуры выравнивания заполняют недостающие байты нулевым значением, чтобы длина секции была кратна определённому значению (см. рис. 3).

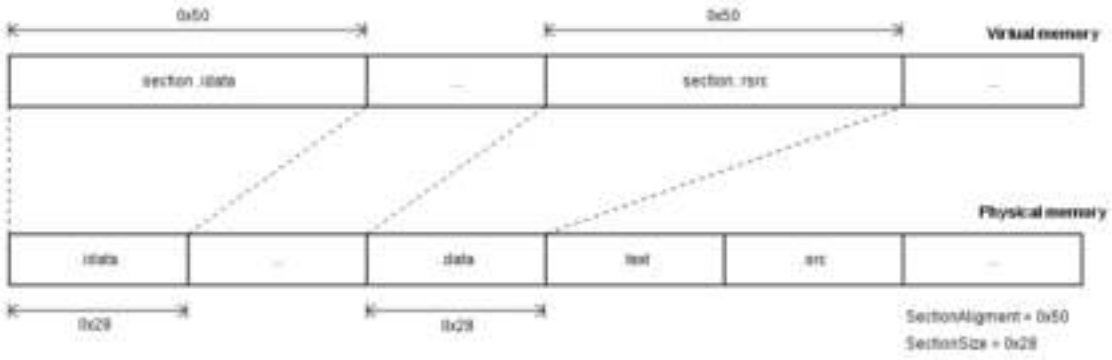

Рисунок 3 – Принцип выравнивания

На рисунке 3 видно, что размер секций в физической памяти = 0x28 байт, тогда как в виртуальной памяти, размер должен быть кратен 0x50. Для этого байты с адресами 0x28-0x50 заполнятся нулевыми байтами 0x00, в таких участках и могут быть спрятаны метки[2].

Внедрение авторской информации в свободные участки, во-первых, гарантирует правильную работу объектного кода, который был изменён, потому что не внедряется в значимую часть кода, а во-вторых, не изменяет размер файла после внедрения метки.

Преимущества:

размер объектного файла не изменится после внедрения, что снижает вероятность обнаружения метки злоумышленником;

внедрение метки не влияет на работоспособность программного продукта;

метод прост в реализации.

Недостатки:

при обнаружении метки злоумышленником она может быть удалена/модифицирована без потери работоспособности программного продукта.

Вторым способом внедрения в объектный код является изменение участков объектных кодов, которые не влияют на работоспособность программного продукта. Например, структура исполнимых PE-файлов такова, что изменение значений некоторых полей не скажется на функциональности программного продукта (см рис.4 и рис. 5)[3].

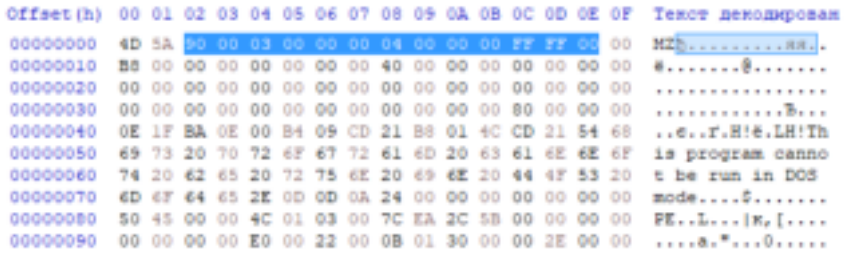

Рисунок 4 – Фрагмент объектного кода без внедрённой метки

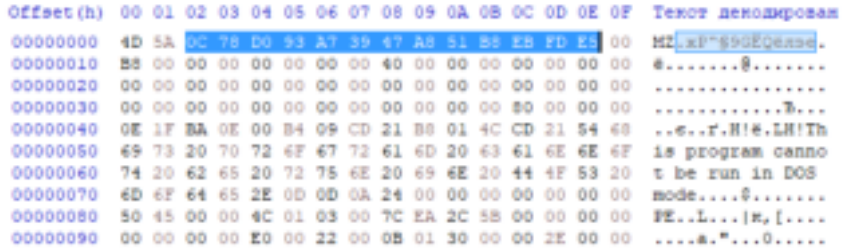

Рисунок 5 – Фрагмент объектного кода с внедрённой меткой

К примеру, DOS-заголовок имеет размер 64 байта, из которых 6 байт являются критически важными: 2 байта отведены для сигнатуры «MZ» в начале каждого PE-файла, без которого файл просто не запустится, и 4 байта для хранения смещения до PE-заголовка. Остаётся 58 байт для произвольного сообщения. [2]

Преимущества:

внедрение метки не влияет на работоспособность программного продукта;

не влияет на размер исполнимого файла;

метод прост в реализации.

Недостатки:

при обнаружении злоумышленником метка может быть удалена/модифицирована без нарушения работоспособности программного продукта.

Третий способ внедрения метки в объектный код основывается на том, чтобы в случае её изменения или удаления, нарушалась работоспособность программного продукта.

Такой метод может быть реализован на этапе кодирования продукта или реализован как отдельный модуль, который, если будет обнаружено нарушение целостности метки в случае ЦВЗ, и если обнаружится несовпадение контрольных данных в случае ЦО, предпримет соответствующие меры. [2]

Преимущества:

удаление/модификация метки напрямую в объектном коде нарушает работоспособность программного продукта;

Недостатки:

без защиты от дизассемблирования может быть модифицирована;

сложнее в реализации в сравнении с предыдущими способами, так как внедрение происходит в значимую часть кода.

Алгоритмы встраивания меток с защитой от атак

В литературе [5] предлагается метод встраивания метки, основанный на доказательстве с нулевым разглашением (англ. *Zero-knowledge proof*). Для его описания нужно ввести понятия «проверяющего» (англ. *Verifier*) и «доказывающего» (англ. *Prover*). Доказывающий – это агент или субъект, который заявляет, что знает доказательство утверждения и пытается его доказать. Проверяющий – это агент или субъект, который пытается узнать доказательство от проверяющего. В конце взаимодействия, называемого протоколом, доказывающий убеждает проверяющего в том, что он знает доказательство, при этом не передавая никаких дополнительных знаний о содержимом информации, знание которой проверяется.

Одним из наиболее известных протоколов идентификации личности с помощью доказательства с нулевым знанием является протокол, предложенный Амосом Фиатом (англ. *Amos Fiat*) и Ади Шамиром (англ. *Adi Shamir*), стойкость которого основывается на сложности извлечения квадратного корня по модулю достаточно большого составного числа n, факторизация которого неизвестна[6].В той же книге автор подтверждает информацию о том, что стеганографические методы для медиа-объектов активно развивались с 90-х годов XX в., но те же методы, применяемые в сфере программного обеспечения, стали темой для изучения относительно недавно.

Иной способ предлагается в материалах международной конференции 2015 года по архитектурной, энергетической и информационной инженерии[7]. Он называется «Усовершенствованный алгоритм динамического программного водяного знака на основе R-дерева» (англ. "*Improved dynamic software watermarking algorithm based on R-tree"*). Этот алгоритм разбит на 4 этапа:

- распределение водяного знака на основе правила переноса переменных m-n (Watermarking sharing based on m-n variable carrying rule);
- этап «нулевого кодирования» (Zero coding);
- преобразование с помощью факториальной системы счисления с переменным основанием (Variable-Base Factorial Number System);
- выполнение функций, связанных с R-деревьями (R-tree related features).

Этот метод, как видно из его названия, является усовершенствованным алгоритмом оригинального алгоритма динамического программирования ЦВЗ на основе R-дерева[8], и в сравнении с ним имеет большую устойчивость к аддитивным атакам, и атакам, связанными с подменой ЦВЗ.

Выводы

В рамках статьи выполнен анализ существующих методов защиты авторского права на программное обеспечение средствами стеганографии. Приведены достоинства и недостатки существующих методов. В дальнейшем планируется разработка авторского алгоритма защиты программного обеспечения средствами стеганографии путем повышения эффективности существующих алгоритмов для защиты авторского права с использованием стеганографии.

Литература

1. Щербаков А. Защита от копирования. – М.: ЭДЭЛЬ, 1992. – 79 с.
2. Peering Inside the PE: A Tour of the Win32 Portable Executable File Format. – Режим доступа: https://docs.microsoft.com/ru-ru/previous-versions/ms809762(v=msdn.10)
3. Монахов М., Ташмухамедова В. Защита авторских прав на программное обеспечение. Владимирский государственный университет, 2009. – 58 с.
4. Середа С. Оценка эффективности систем защиты программного обеспечения. — Режим доступа: http://www.security.ase.md/publ/ru/pubru30.html
5. M. Barni et al. (Eds.): IWDW 2005, LNCS 3710, pp. 299–312, 2005.©Springer-Verlag Berlin Heidelberg 2005.
6. Feige U., Fiat A., Shamir A. Zero Knowledge Proofs of Identity // Journal of Cryptology. - 1988. - vol. 1, Iss. 2. - pp. 77-94.
7. L. He & J.F. Xu Improved dynamic software watermarking algorithm based on R-tree// Architectural, energy and information engineering (AEIE 2015), Xiamen, China, 2015 – С. 531-535.
8. Xu, H. & Chen, H. & Feng, D. & Li, D.. (2005). Dynamic software watermarking algorithm. 33. 172-174.

Полуденный Н.И., Чернышова А.В. Анализ существующих методов защиты авторского права на программное обеспечение средствами стеганографии. *Целью работы является сравнение методов сокрытия цифрового водяного знака (ЦВЗ) в программном обеспечении, выделение их преимуществ и недостатков для дальнейшего поиска возможностей устранения недостатков и повышения эффективности алгоритмов защиты авторского права на программное обеспечение.*

Ключевые слова: *стеганография, защита, авторское право, программное обеспечение, цифровой водяной знак, цифровой отпечаток.*

Poludennyy Nikita, Chernyshova Alla. Improving the effectiveness of software copyright protection algorithms by means of steganography. *Analysis of existing methods of software copyright protection by means of steganography. The aim of the work is to compare methods of hiding a digital watermark in software, highlighting their advantages and disadvantages to further search for opportunities to eliminate shortcomings and improve the efficiency of algorithms for protecting copyright for software.*

Key words: *steganography, protection, copyright, software, digital watermark, digital fingerprint.*