УДК 681.513.2

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ИДЕНТИФИКАЦИИ КОМПЬЮТЕРНЫХ ПРОГРАММ

М.К. Буза, Е.Н. Ливан

Введение.

Защита программного продукта от несанкционированного воспроизведения, использования и/или модификации направлена, прежде всего, на защиту авторских прав его создателя. Авторские права разработчиков программного обеспечения охраняются специальными законами, как на национальном, так и на международном уровне.

Программы для ЭВМ Законом РФ "О правовой охране программ для электронных вычислительных машин и баз данных" относятся к объектам авторского права, причем правовая охрана программам предоставляется как произведениям литературы. Закон РФ "Об авторском праве и смежных правах" также среди объектов охраны предусматривает программы для ЭВМ, причем охрана распространяется на программы вне зависимости от языка программирования.

Законом Республики Беларусь "Об авторском праве и смежных правах" компьютерные программы относятся к объектам авторского права и охраняются как литературные произведения. Закон Украины "Об авторском праве и смежных правах" также предусматривает охрану компьютерных программ.

Однако указанные выше законы не требуют обязательного депонирования, регистрации или соблюдения иных формальностей для признания и осуществления авторского права на компьютерные программы.

Для защиты программного обеспечения (ПО), как правило, используются программные, аппаратные и программно-аппаратные средства. Традиционно они основаны либо на противодействии незаконному копированию программ, либо на противодействии попыткам запуска и/или исполнения незаконных копий продукта, включая меры противодействия попыткам нарушителя исследовать логику работы программы. Как показывает практика, методы защиты ПО, основанные на таком подходе, затрудняют его нелегальное распространение и использование, но не решают

проблему.

Альтернативой традиционному способу защиты авторских прав в области ПО является подход, основанный на сопровождении программного кода (каждой копии программы) скрытой информацией, которая может служить подтверждением авторства на программу.

Кроме того, данный подход ласт дополнительную возможность и средства специалистам в области компьютерной технической экспертизы упростить процесс производства судебных экспертиз при раскрытии и расследовании преступлений, связанных со всевозможными нарушениями авторских и смежных прав.

Предлагаемое решение.

В качестве практического средства, обеспечивающего защиту авторских прав разработчиков ПО, предлагается использовать автоматизированную систему идентификации компьютерных программ SILDE (Steganografy Identification Labels for Defencing and appraisal by Experts).

Разработанная система может быть использована также в качестве программного средства эксперта-криминалиста, специализирующегося по делам о компьютерных преступлениях и, в частности, по делам, связанным с нарушениями авторских прав создателей программного обеспечения.

Ниже рассматриваются идеи, методы, механизмы и алгоритмы, лежащие в основе функционирования данной автоматизированной системы, а также ее структура и взаимодействие подсистем.

Основные понятия и определения.

Информацию, которая сопровождает программный продукт и идентифицирует его автора, будем называть информацией об управлении правами.

Внедрение в программы и сопровождение программ скрытой информацией об управлении правами назовем идентификацией программ. Коды, в которых представлена скрытая информация об управлении правами, назовем идентификационными метками.

Аутентификация программы — установление экспертом того факта, что программа с высокой вероятностью принадлежит данному автору.

Надежность идентификационных меток, будем связывать с мощностью механизма

их зашиты от несанкционированной модификации или удаления. Под мощностью механизма зашиты понимается способность защитного механизма противостоять прямым атакам злоумышленника. Дли оценки мощности механизма зашиты, будем использовать градации мощности, определенные и существующих стандартах по оценке безопасности информационных технологий — базовую, среднюю и высокую.

Технология идентификации программ SILDE.

Автоматизированная система базируется на специально разработанной технологии идентификации программ Technology SILDE, в основе которой лежат классические и оригинальные методы криптографии и компьютерной стеганографии, а также алгоритмы теории кодирования.

Основная идея технологии — сопровождение объекта защиты скрытой информацией об управлении правами, представленной в виде специального кода — идентификатора программы.

Предлагаемая технология обеспечивает не только идентификацию программ, но и их аутентификацию.

При идентификации программ технология базируется на следующих основных положениях:

1. Объект защиты является исполняемым файлом программы.

2. Идентификатор программы, представляющий собой уникальный цифровой код, создается на основе текстовой информации, содержащей сведения об авторе и сведения о защищаемом программном продукте, а также на основе анализа клавиатурного почерка пользователя системы.

3. Идентификатор программы записывается в зашифрованном виде на отчуждаемый ключевой носитель в качестве эталона.

4. Количество идентификационных меток, предназначенных для сопровождения объекта защиты, определяется на основе параметров защищаемого объекта, идентификатора программы и степени защиты (базовая, средняя, высокая), выбранной пользователем системы.

5. Идентификационные метки различных типов создаются на основе идентификатора программы с помощью классических и специально разработанных алгоритмов теории кодирования и методов криптографического преобразования

информации.

6. Идентификационные метки внедряются в объект защиты классическими и оригинальными стегано графическими методами. Причем решение об используемых стегано графических методах принимается на основе параметров защищаемого объекта.

7. В объект защиты внедряется дополнительный модуль, предназначенный для восстановления модифицированных или удаленных нарушителем меток. Пользователю системы предоставляется возможность отказаться от такого вида защиты, так как внедрение данного модуля изменяет длину исполняемого файла и время выполнения защищаемой программы.

Аутентификация программ в соответствии с рассматриваемой технологией производится следующим образом:

на ключевом носителе системе предъявляется эталонный идентификатор программы;

строятся эталонные идентификационные метки;

производится анализ параметров защищаемого объекта;

вычисляются адреса идентификационных меток и их длины;

сравниваются эталонные идентификационные метки со значениями, записанными по определенным адресам;

производятся экспертные оценки модификации и удаления, меток нарушителем:

выдается заключение.

В качестве ключевого носителя в рассматриваемой версии системы используется гибкий магнитный диск. Эталонный идентификатор на диск записывается, стегано графическими методами. Заметим, что в случае дополнения системы аппаратными средствами в качестве ключевого носителя может быть использовано любое современное устройство, предназначенное для хранения персональной информации (магнитная карта, электронный ключ, жетон и т.п.).

Пользователь системы может проводить идентификацию нескольких программных продуктов с помощью одного и того же идентификатора программы. Для этого пользователю необходимо "предъявить" ключевой носитель с эталонным идентификатором и выбрать соответствующие опции работы системы.

Проблема стойкости идентификационных меток к попыткам обнаружения и удаления (модификации) нарушителем решается комплексным подходом к созданию и

внедрению меток. Во-первых, надежность идентификационных меток обеспечивается с помощью используемых методов, механизмов и алгоритмов создания и внедрения меток. Во-вторых, используется оригинальный психологический подход, базирующийся на индивидуальном ограничительном барьере злоумышленника и его квалификации.

Предполагается вызвать у нарушителя чувство психологического напряжения и сомнения в том, что в несанкционированно используемом программном коде могут сохраняться средства защиты прав автора. Такой подход обеспечивается следующими решениями:

оповещением нарушителя о том, что в программе присутствует информация об управлении правами в виде идентификационных меток;

использованием как традиционных стегано графических методов внедрения информации в исполняемые файлы, так и оригинальных стегано графических механизмов;

внедрением различного количества меток нескольких типов в защищаемые объекты.

Таким образом, технология внедрения меток преднамеренно инициирует их раскрытие. Предполагается, что, обнаружив часть меток, нарушитель будет считать, что обнаружил их все. и прекратит дальнейший поиск.

Еще одной существенной особенностью технологии является оригинальный метод кодирования идентификационных меток. Некоторые метки закодированы таким образом, что при их модификации неавторизованными лицами они позволяют обнаруживать этот факт, а также (по желанию пользователя системы) способны само восстанавливаться.

Для реализации такой возможности технология идентификации программ предусматривает возможность внедрения в защищаемый объект дополнительного модуля, предназначенного для:

обнаружения модификации или удаления меток третьего типа;

восстановления идентификационных меток при наличии хотя бы одной из них.

Модуль внедряется по технологии, аналогичной вирусным технологиям внедрения в исполняемые файлы. При запуске идентифицированной программы коды модуля выполняются до кодов основной программы, осуществляют проверку всех меток и в случае, если некоторые из них удалены, по оставшимся меткам восстанавливают удаленные. Работа модуля зашумляется. Заметим, что механизм внедрения дополнительного

модуля позволяет избежать ложных тревог при тестировании идентифицированного файла антивирусными средствами.

Принципы создания и внедрения, идентификационных меток.

Технология идентификации программ предполагает внедрение идентификационных меток нескольких типов.

Идентификационные метки первого типа предназначены для обеспечения базовой мощности механизма защиты идентификационных меток. Они представляют собой зашифрованный с помощью одного из крипто алгоритмов идентификатор программы с последующим сжатием. В качестве ключа шифрования используется идентификатор автора.

Для внедрения идентификационных меток первого типа используются, стегано графические методы, основанные на наличии неиспользуемых (зарезервированных) полей в заголовках файлов форматов EXE (исполняемые файлы DOS) и PE (от Portable Executable, исполняемые PE-файлы Windows). Исследование форматов исполняемых файлов показало, что общий размер неиспользуемых полей является достаточным для внедрения некоторого количества идентификационных меток. Например, для формата PE такими полями могут являться поля, представленные в табл. 1.

Таблица 1

Примеры неиспользуемых полей PE-файла

Название поля (в соответствии с заголовочным файлом winnt.h)	Расположение	Смещение	Размер, байт	Назначение
C_rcs[4]	DOS-заголовок	1Ch	8	Зарезервировано
C_res2[10]	DOS-заголовок	28h	20	Зарезервировано
TimcDatcStamp	PE-заголовок	08h	4	Время создания файла
PoinlcrToSymboITablc	PE-заголовок	OCh	4	Смещение таблицы символов отладочной информации
NumbcrOfSymbol	PE-заголовок	010h	4	Количество символов в таблице символов

Magic	Optional-заголовок	018h	1	Не имеет информационной нагрузки
Reserved 1	Optional-заголовок	4Ch	4	Не используется
LoaderFlags	Optional-заголовок	70h	4	Не используется

Идентификационные метки второго типа предназначены для обеспечения средней мощности механизма защиты идентификационных меток. Рассмотрим алгоритм построения таких меток.

Представим идентификатор программы в виде последовательности десятичных цифр длины d:

$a = (a_1, a_2, ..., a_j, ..., a_d)$, где $a_j \in \{0,...,9\}$

Каждую десятичную цифру данной последовательности запишем соответствующей двоичной тетрадой. Получим последовательность длины $4d$ в двоичном алфавите:

$j = (j_1, j_2, ..., j_j, ..., j_d)$, где $j_j \in \{0,...,9\}$

Идентификационная метка второго типа с номером j строится на основе идентификатора программы j с помощью такой функции F, что

$$F_i(j) = S(P_i(V_i(j))) \oplus W_i(S(j))$$

где V_i — функция, осуществляющая выбор n символов (n кратно 3) из последовательности j по заданном правилу; P_i — операция перестановки; S – операция поблочной подстановки; W_i —операция выборки бит.

Для внедрения идентификационных меток второго типа используются, стегано графические методы, которые условно можно разделить на две группы:

1. Методы, основанные на наличии и файлах форматов EXC и PE полей, изменение значений которых не влияет на корректную работу программы. Исследование исполнимых файлов Windows показало, что такими полями могут являться, например, поля, представленные в табл. 2, а также многие другие.

2. Методы и алгоритмы, использующие принципы технологий внедрения файловых вирусов в исполняемые файлы в форматах EXE и PE.

Третий тип меток (самовосстанавливающиеся метки) предназначен для

обеспечения высокой мощности механизма защиты идентификационных меток.

Таблица 2

Предполагаемые ноля PE-файла для внедрения меток

Название поля (в соответствии с заголовочным файлом wirmt.h)	Расположение	Смещение	Размер, байт	Назначение
C_crlc	DOS-заголовок	06 h	2	Число элементов в таблице настройки адресов
C_csum	DOS-заголовок	12h	2	Контрольная сумма
SizeOfOptionalHeader	PE-заголовок	14h	2	Размер опционального заголовка
MajorLinkerVcrsion	Optional-заголовок	1Ah	1	Старшая часть версии линковщика, создавшего файл
MinorLinkerVcrsion	Optional-заголовок	1Bh	1	Младшая часть версии линковщика, создавшего файл
MajorOperatingSystcmVersion	Optional-заголовок	40h	2	Старшая часть номера самой старой версии ОС, которая позволяет запускать данный файл
MinorOpcrationSystcmVersion	Optional-заголовок	42h	2	Младшая часть номера самой старой версии ОС, которая позволяет запускать данный файл
MajorImageVersion	Optional-заголовок	44h	2	Старшая часть номера версии исполняемого файла
MinorImageVersion	Optional-заголовок	46h	2	Младшая часть номера версии исполняемого файла
MajorSubsystcmVersion	Optional-заголовок	48h	2	Старшая часть самой старой версии Win32 — подсистемы, позволяющей использовать данный файл
MinorSubsystemVersion	Optional-заголовок	4Ah	2	Младшая часть самой старой версии Win32 — подсистемы позволяющей использовать данный файл

Метки данного типа строятся на основе последовательности символов длины n в двоичном алфавите

$$g = V_i(j) = (g_1, g_2, ..., g_j, ..., g_n)$$

где V_i — функция, осуществляющая выбор n символом (где n кратно 3) из последовательности j по заданному правилу, $g_i \in \{0,1\}$.

Последовательность g разбивается на три части. Представим ее в виде:

$$g = (x_1, x_2, ..., x_k, y_1, y_2, ..., y_k, h_1, h_2, ..., h_k),$$
$$где\_k = n/3$$

Обозначим $x = (x_1, x_2, ..., x_k), y = (y_1, y_2, ..., y_k), h = (h_1, h_2, ..., h_k)$

Каждый из наборов x, y, h в отдельности подвергается дальнейшему кодированию:

$$x \to c_x; y \to c_y; h \to c_h$$

Идентификационные метки, внедряемые в исполняемый файл программы, строятся на основе наборов x, c_y, y, c_y, h, c_h и могут быть представлены в следующем виде:

$M_1 = (x, c_y, c_h);$
$M_2 = (c_x, y, c_h);$
$M_3 = (c_x, c_y, h);$

Заметим, что длина каждой метки составляет 9 к бит.

Рассмотрим алгоритм кодирования наборов x, y, h на примере набора x.

Критическим участком исполнимого файла назовем такую последовательность байт, в которой каждый бит существенно влияет на работоспособность файла, т. е. при изменении хотя бы одного бита в данной последовательности исполнимый файл либо не работает, либо работает некорректно.

По определенному правилу из некоторого критического участка исполнимого файла выбирается последовательность к байт:

$$b_i = (b_{i1}, b_{i2}, ..., b_{ij}, ..., b_{ix}), i = [1, ..., k] \quad (1)$$

Для каждого набора b_i строится набор b_i^1 такой, что

$$b_i^1 = Y_m(b_i) = (b_{i1}, ..., b_{im-1}, .b_{im+1}, ..., b_{i8}), m = [1, ..., 8] \quad (2)$$

где Y_m — функция, удаляющая символ с заданным номером m, который вычисляется с помощью хеш-функции h_i, по правилу $m = h_j(j) \bmod 8$.

Далее к каждому набору b_i^1 применяется функция I_j, которая для $I_j = h_j(j) \bmod 7$ (h_2 —хеш-функция) преобразует b_{ij} по следующему правилу:

$$b_{ij} = 0, если b_{ij} = x_i,$$
$$b_{ij} = 1, если b_{ij} \neq x_i, где x_i \in x \quad (3)$$

Получаем $b_i^{11} = I_j(b_i^1)$, где $i = [1, ..., k], j \in [1, ..., 7]$

Заметим, что в результате применения функции I_j наборы b_i^1 и b_i^{11} могут различаться только в символе с номером j. Длина набора b_i^{11} равна 7.

Далее для каждого набора b_i^{11} строится код Хемминга: последовательность b_i^{11} кодируется набором

$$c_i = (c_{i1}, c_{i2}, b_{i1}, c_{i3}, b_{i2}, b_{i3}, b_{i4}, c_{i4}, b_{i5}, b_{i6}, b_{i7}) \quad (4)$$

где $c_{i1}, c_{i2}, c_{i3}, c_{i4}$ – контрольные биты;

$b_{i1}, b_{i2}, b_{i3}, b_{i4}, b_{i5}, b_{i6}, b_{i7}$ — биты последовательности b_i^{11}.

Для кодирования набора $x = (x_1, x_2, ..., x_k)$ используется последовательность, состоящая только из контрольных бит кола Хемминга, построенного для каждого x:

$$x \to c_x = (c_{11}, c_{12}, c_{13}, ..., c_{i1}, c_{i2}, c_{i3}, ..., c_{k1}, c_{k2}, c_{k3}) \quad (6)$$

Таким образом, каждая тетрада последовательности c_x контролирует соответствующий бит идентификационной метки. Заметим при этом, что длина кода c_x составляем 4k бит.

Для внедрения самовосстанавливающихся меток используются оригинальные стегано графические методы внедрения информации в исполнимые файлы Windows.

Структура автоматизированной системы идентификации программ.

Система идентификации программных продуктов является совокупностью обеспечивающей и функциональной подсистем. Подсистемы взаимодействуют посредством отчуждаемого ключевого носителя.

Обеспечивающая подсистема. С помощью обеспечивающей подсистемы пользователь идентифицирует объект защиты, при этом эталонный идентификатор программы записывается на отчуждаемый ключевой носитель. Функциями данной подсистемы являются:

создание идентификационных меток на основе авторской информации об управлении правами и других параметров;

сохранение эталонной информации на отчуждаемый носитель;

внедрение в защищаемый объект скрытой информации об управлении правами;

внедрение в защищаемый объект специальною модуля, обнаруживающего модификацию (удаление) меток и восстанавливающего их.

На рис. 1 изображена структура обеспечивающей подсистемы.

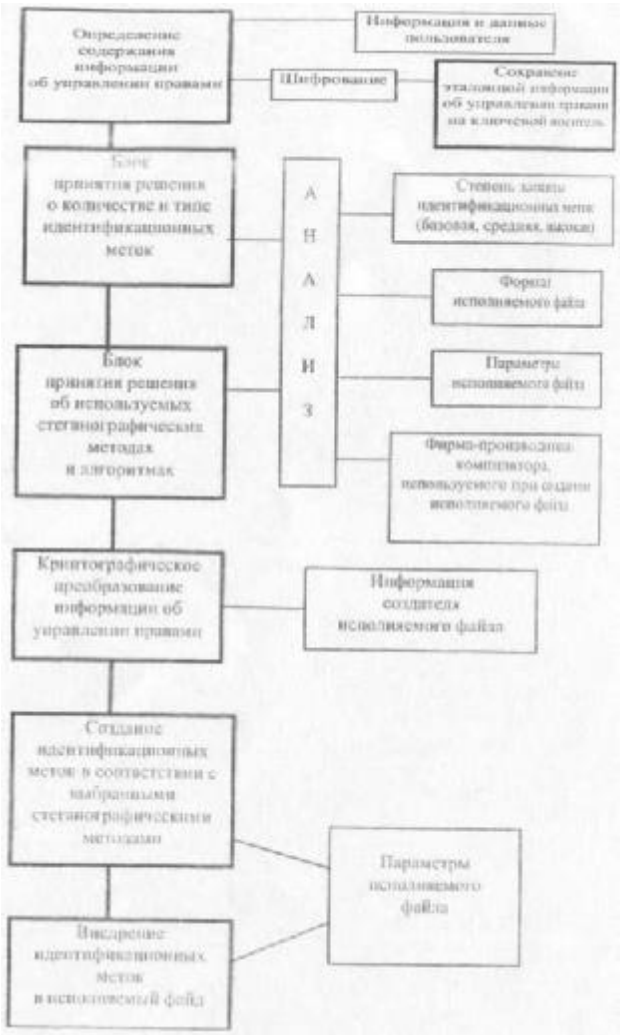

Рис. I. Структура обеспечивающей подсистемы

Первоначально на основе вводимой пользователем информации строится идентификатор программы, представляющий собой уникальный цифровой код. Построение идентификатора программы происходит в несколько этапов:

1. Текстовая информация, содержащая сведения об авторе (имя разработчика, адрес, e-mail, телефон и т.п.) и сведения о защищаемом программном продукте (название, номер версии, дата выхода версии и другие характеристики программы), колируется с помощью специальною алгоритма. Полученный в результате кол назовем идентификатором автора.

2. На основе анализа клавиатурного почерка автора вычисляется значение специальной хеш-функции.

3. В результате преобразования идентификатора автора и значения упомянутой хеш-функции получается идентификатор программы.

4. В зашифрованном виде идентификатор программы записывается на ключевой носитель.

В процессе анализа перечисленных на рис. 1 параметров принимаются решения:

Об используемых типах идентификационных меток;

о количестве идентификационных меток, внедряемых в исполняемый файл;

об используемых стегано графических методах и алгоритмах.

Блок подсистемы, производящий анализ метров исполняемого файла – один из наиболее ответственных. На рис. 2 изображены анализируемые структуры исполняемого файла в формате PE.

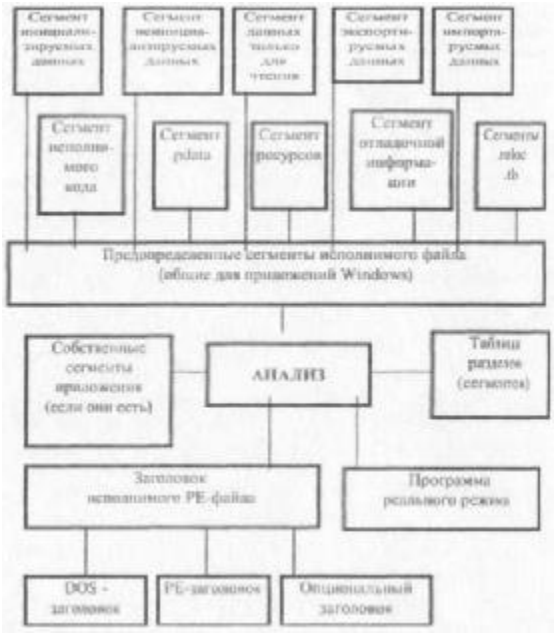

Рис. 2. Анализируемые параметры исполняемого PE-файла

В процессе анализа параметров исполнимого файла определяется:

общее количество байт, доступных для внедрении меток;

смещения и размер отдельных участков файла, доступных для внедрения меток.

Функциональная подсистема. С помощью функциональной подсистемы эксперт производит аутентификацию программ сравнением информации об управлении правами, внедренной в программу, и являемой эталонной информации на ключевом носителе. Подсистема позволяет:

считывать информацию об управлении правами, сопровождающую защищенный объект;

считывать эталонную информацию, хранящуюся на ключевом носителе;

устанавливать факт авторства сравнением с предъявляемой эталонной информацией об управлении правами.

На рис. 3 изображена структура функциональной подсистемы.

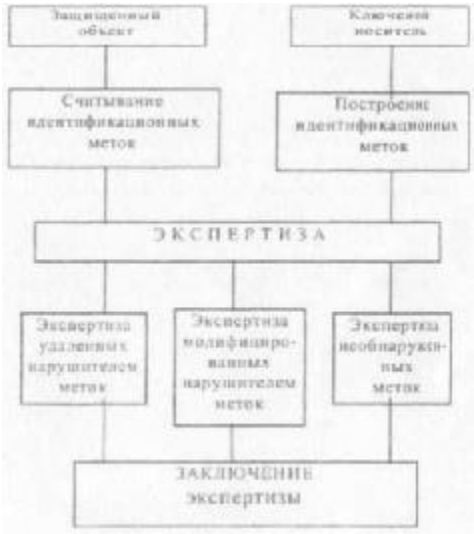

Рис. 3. Структура функциональной подсистемы

Заключение.

Автоматизированная система идентификации программ SILDE предназначена для зашиты авторских прав создателей уже готовых программ (DOS- и Windows-приложений) и функционирует в операционных системах Windows 95/98/2000, Windows NT. Система поддерживает форматы исполняемых файлов EXE, New EXE. и PE.

Для работы с системой разработан удобный пользовательский интерфейс, который реализован на языке C+ + .

Основные алгоритмы и защитные механизмы технологии идентификации программ реализованы на языке Assembler.