УДК 004:056

СИСТЕМА ЗАЩИТЫ АВТОРСКИХ ПРОГРАММНЫХ ПРОДУКТОВ

ГАЙВОРОНСКАЯ Г.С., СОМСИКОВ Д.А.

Одесская государственная академия холода
Одесский национальный университет им. И. И. Мечникова

PROPRIETARY SOFTWARE PROTECTION SYSTEM

GAYVORONSKA G.S., SOMSIKOV D.A.

Odessa State Academy of Refrigeration
Odessa National I.I. Mechnikov University

***Аннотация**. Представлена система, позволяющая обеспечить защиту от несанкционированного доступа и копирования авторских программных разработок, а также гибкую систему регистрации и возможность работы в пробном режиме.*

***Abstract.** The system that allows to protect proprietary software against unauthorized access and copying, and has a flexible system of registration and the opportunity to work in trial mode is presented.*

ВЕДЕНИЕ

В настоящее время остро встали вопросы защиты интеллектуальной собственности, и особенно авторского программного обеспечения (ПО), создаваемого индивидуально для решения какой-то конкретной задачи. Создание таких эксклюзивных программ обычно требует оригинальных подходов и использования нетривиальных решений и, соответственно больших интеллектуальных и трудовых затрат. В связи с этим затраты времени и сил программистов на написание ПО постоянно увеличиваются, а значит растет и его стоимость. В тоже время постоянно совершенствуются методы «взлома» программ. Поэтому каждая относительно сложная программа требует надежной системы защиты, что приводит к еще большим затратам времени и сил со стороны разработчиков, одновременно усложняя и удорожая программный продукт.

Одновременно с этим задачи защиты программного обеспечения являются, в какой то мере однотипными и могут быть реализованы в виде отдельных унифицированных модулей.

ПОСТАНОВКА ЗАДАЧИ

В данной работе представлена система, обеспечивающая гибкую и многофункциональную защиту программного обеспечения от несанкционированного доступа. С помощью этой многофункциональной системы защиты ПО можно снабдить любую программу такими качествами, как: защита от несанкционированного доступа и копирования, гибкая система регистрации и возможность работы в пробном режиме. Система защиты, названная «Сейф», использует собственные технологии защиты, такие как MDCP™ (защита от ручного изменения даты), RIP™ (защита от повторных инсталляций) и др., обеспечивающие устойчивость системы от большинства методов взлома программ. Система «Сейф» может использоваться любыми программами, написанными на языках поддерживающих технологию *ActiveX* (например, *Visual C++, Visual Basic, Visual J++, Borland Builder*).

Система «Сейф» не зря имеет такое название: принцип ее работы сходен с принципом работы сейфа, рис 1. Программист помещает программу в некий условный сейф. Когда пользователь пытается получить доступ к программе, он вначале взаимодействует

с сейфом, пытаясь его открыть. Сейф обладает определенной особенностью – его может приоткрыть (работать в пробном режим) любой, но только в течение ограниченного периода времени, и ограниченное количество раз. Однако полностью открыть сейф (получить доступ к полноценной программе) можно только, если ввести код (зарегистрироваться). Лишь после того, как пользователь откроет или приоткроет сейф, он будет взаимодействовать с тем, что в нем лежит (программой), до этого он взаимодействует с сейфом. Любые попытки получить доступ к программе, минуя сейф, немедленно пресекаются.

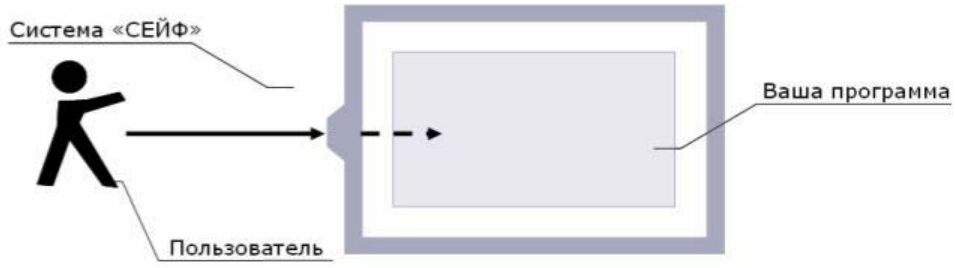

Рисунок 1 – Общий принцип работы системы «Сейф»

Особенностью реализации подобных систем, является необходимость учета трех взаимодействующих сторон. Каждая из сторон охарактеризована следующим образом:

1. Разработчик системы – т. е автор данной работы, предоставляющий возможность программистам, используя предложенную систему защищать свои программные разработки от взлома со стороны взаимодействующих с ним пользователей.

2. Клиент (программист), использующий систему защиты для разрабатываемого им программного обеспечения

3. Пользователь, использующий клиентскую программу.

Далее в тексте разработанное программное обеспечение называется – система «Сейф», защищаемая программа названа клиентской программой.

Целью настоящей работы является создание системы, унифицирующей решение задачи защиты ПО и наделения его такими качествами, как необходимость регистрация, пробный режим и т. п. Проблема унификации такого решения является комплексной и может быть разбита на несколько задач:

- запрет доступа к программе, минуя систему защиты;
- обеспечение механизма регистрации;
- обеспечение проверки факта регистрации;
- обеспечение надежного хранения информации о пользователе;
- обеспечение устойчивости к распространенным методам «взлома» программ.

Проанализируем методы решения этих задач.

ЗАПРЕТ ДОСТУПА К КЛИЕНТСКОЙ ПРОГРАММЕ, МИНУЯ СИСТЕМУ ЗАЩИТЫ

Для того чтобы запретить доступ к клиентской программе, минуя систему защиты, система защиты должна являться автономной частью клиентской программы. С одной стороны, необходимо дать клиенту возможность управления и конфигурирования системы защиты, с другой – клиент не должен иметь доступа к исходному коду системы «Сейф». Поэтому, ее главный модуль (ГМ) реализован в виде компонента *ActiveX*. Использование технологии *ActiveX*, предполагает, что в основе программы лежит концепция *COM* (*Component Object Model* – модель составного объекта). Технология *COM* позволяет создавать приложения и компоненты, состоящие из объектов, написанных разными программистами на разных языках. Кроме того, с ее помощью объекты могут вызывать методы (команды других объектов), обращаться к свойствам других объектов и инициировать события других объектов даже в том случае, если последние находятся на другом компьютере в сети. Такой подход позволяет достичь достаточно тесной интегра-

ции системы защиты с защищаемой программой при отсутствии у клиента прямого доступа к коду компонента.

Для того чтобы настроить работу системы защиты, клиент может изменить значения свойств ГМ. Чтобы передать системе защиты команду, клиент может вызвать метод класса ГМ. Клиентское приложение информируется о происходящем путем передачи событий. Так как ГМ реализован в виде компонента *ActiveX*, с ним можно работать как с обычным элементом управления, таким как кнопка или текстовое поле. Однако, в отличие от кнопки и большинства других элементов управления, ГМ не имеет графического представления во времени выполнения (*Run time*), и изображается в виде небольшого квадрата во времени разработки (*Design time*).

МЕХАНИЗМ РЕГИСТРАЦИИ

Существует несколько вариантов реализации механизма регистрации. Самым простым и наиболее распространенным является вариант, при котором каждая копия защищаемой программы обладает уникальном «серийным номером», который нужно ввести для того, чтобы зарегистрировать программу. Основным преимуществом этого варианта является простота реализации, однако задание отдельного «серийного номера» для каждой копии программы при больших тиражах – достаточно емкая задача. Существенным недостатком этого способа является то, что, купив один дистрибутив и зарегистрировав его, пользователь может устанавливать с него неограниченное количество копий программы на неограниченное число компьютеров. Также, несмотря на достаточно большие размеры «серийного номера», его можно вычислить методом случайного подбора. К тому же достаточно часто встречаются программы, где написано нечто вроде: «Можете ввести сюда какой-нибудь серийный номер, хотя это и необязательно», то есть исходный код программы был изменен таким образом, что регистрация не требует «серийного номера» или любой «серийный номер» воспринимается как правильный.

Следующий вариант – использование «файлов-ключей». При таком варианте, программе для запуска требуется «файл-ключ» с определенной информацией. К преимуществам данного варианта относятся то, что в зависимости от «файла-ключа» программа может работать в разных режимах разное время, а также то, что «файл-ключ» практически невозможно подобрать. Недостатком является возможность использования одного «файла-ключа» с несколькими копиями программы. К тому же остается проблема возможности прямого изменения исходного кода программы.

Третий вариант обычно используют для защиты наиболее дорогих программ (например, *Discreet 3D Studio MAX, Shlumberger Dino+*). При его использовании на стороне пользователя генерируется первичный код (называемый *U*-кодом). Этот код пользователь должен сообщить программисту. Если программист (клиент разработчика системы) желает зарегистрировать пользователя, он на основании *U*-кода, используя сложный и неочевидный алгоритм, генерирует вторичный код (*P*-код) и отправляет его пользователю. Пользователь вводит *P*-код в программу и, если он соответствует созданному *U*-коду, программа считается зарегистрированной. Основное преимущество данного метода – то, что код не жестко записывается в программу на стороне программиста, а генерируется случайно на стороне пользователя, а также то, что каждый раз регистрируя программу, пользователь должен обратиться к программисту и обосновать необходимость регистрации. Благодаря этому программист может контролировать число установок программы конкретным пользователем. Главным недостатком этого алгоритма является его сложность. К тому же все равно существует угроза прямого редактирования исходного кода.

В данной работе используется модифицированный третий вариант. Для обеспечения полиморфизма шифрования, а также для некоторых других функций программы,

каждая программа обладает определенным основанием защиты – числом на основании, которого производятся все операции в программе.

Прежде всего, на стороне пользователя создается *U*-код, как функция от случайного числа:

$$U = G(r),$$

где U – создаваемый *U*-код,
G – функция генерации нового *U*-кода,
r – случайное число.

Этот код пользователь передает программисту, который на основании *U*-кода создает *P*-код:

$$P = R(U,\beta),$$

где P – *P*-код, соответствующий данному U-коду,
R – функция генерации *P*-кода на основании *U*-кода,
U – *U*-код, переданный пользователем,
β – основание защиты.

Программист передает пользователю *P*-код, и программа на стороне пользователя, используя обратную функцию, генерирует *U*-код на основании полученного P-кода:

$$U' = R^{-1}(P,\beta),$$

где U' – *U*-код, соответствующий полученному *P*-коду,
R^{-1} – функция, обратная функции R, предназначенная для генерации *U*-кода на основании *P*-кода,
P – *P*-код, полученный от пользователя,
β – основание защиты.

Если оказывается, что $U' = U$, то программа считается зарегистрированной. В противном случае ($U' \neq U$), программа не регистрируется, и пользователю дается еще некоторое ограниченное число попыток ввести верный *P*-код.

Функции R и R^{-1} сводятся к следующему: U-код представляется в виде множества чисел U, где U_i – i-ый символ *U*-кода. Далее определяется четыре параметра *A, B, C* и *D.*

$$A = \sum_{i=1}^{5} \gamma(U_i + 1)(i + \beta),$$

$$B = \sum_{i=6}^{10} \gamma(U_i + 1)(i + \beta),$$

$$C = \sum_{i=11}^{15} \gamma(U_i + 1)(i + \beta),$$

$$D = \sum_{i=16}^{20} \gamma(U_i + 1)(i + \beta),$$

где γ – функция шифрования,
β – основание защиты.

Затем, при помощи функции объединения φ получаем P-код:

$$P = \varphi(A, B, C, D),$$

где P – *P*-код, соответствующий данному *U*-коду,

φ – функция объединения,
U – *U*-код, переданный пользователем.

ОБЕСПЕЧЕНИЕ НАДЕЖНОГО ХРАНЕНИЯ ИНФОРМАЦИИ О ПОЛЬЗОВАТЕЛЕ

Информация о пользователе может храниться на съемных носителях, жестком диске или удаленном компьютере. Первый вариант широко применялся в прошлом десятилетии, однако большинство программистов отказались от него из-за значительных неудобств, приносимых пользователю. В действительности, многим пользователям неудобно при каждом запуске программы вставлять дискету или диск. К тому же использование носителей небольшого объема упрощает работу взломщика, так как при малых объемах носителя нужную информацию легче найти.

Второй вариант менее надежен, так как информация о пользователе хранится на компьютере пользователя, и, следовательно, ее теоретически можно найти и модифицировать. Однако он наиболее удобен для пользователя, так как для запуска программы ему не требуется делать ни каких дополнительных действий.

Третий вариант наиболее надежен, потому что информация о пользователе хранится на компьютере разработчика или его доверенного лица, и защиту информации в таком случае осуществить достаточно легко. Однако не у каждого пользователя есть возможность подключиться к компьютеру разработчика и считать нужную информацию. Кроме того, подключаться к сети при каждом запуске программы может быть неудобно для пользователя.

Поэтому, в данной работе используется второй вариант. При хранении информации на жестком диске также существуют различные варианты, информация может храниться, например, в системном реестре или в особых файлах.

В предлагаемой работе применяется смесь нескольких подходов. Информация хранится в двух независимых хранилищах (системном реестре и файлах динамически-подключаемых библиотек). Каждый раз при считывании информации производится проверка соответствия информации из одного хранилища информации из другого. Для исключения возможности трассировки записи (отслеживания места записи) среди записываемой информации только около 4% является полезной, остальная информация используется для сокрытия полезной информации. Вся информация шифруется с применением полиморфического шифрования (т. е. одна и та же информация в разные моменты времени шифруется по-разному).

ОБЕСПЕЧЕНИЕ УСТОЙЧИВОСТИ К РАСПРОСТРАНЕННЫМ МЕТОДАМ «ВЗЛОМА» ПРОГРАММ

В настоящее время существует достаточно большое количество разнообразных методов «взлома» программ. Эти методы различны как по целям, так и по способу достижения этих целей. Среди возможных целей «взлома» выделим получение полнофункциональной (якобы зарегистрированной) программы из пробной или незарегистрированной версии и продление срока действия пробного режима.

Возможны следующие пути достижения таких целей:

- модификация кода программы или системы защиты, путем декомпиляции или дисассемблирования;
- изменение значений системных параметров (например, даты и времени) на которых основана работа системы защиты;
- модификация или удаление информации, которую использует система защиты.

Известно, что для каждого действия существует противодействие и обеспечить абсолютную защиту нельзя, однако можно довести качество защиты до такого состояния, что «взламывать» программу будет не рентабельно – стоимость «взлома» будет больше стоимости самой программы.

Для предотвращения действий, направленных на «взлом» программы могут применяться разные приемы. Например, для защиты от модификации, можно применять различные варианты проверки самоцелостности, в частности проверку контрольной суммы, проверку наличия служебных битов, и т. п. Однако такие методы позволяют лишь обнаружить изменение, но не дают возможности предотвратить его. В лучшем случае, при обнаружении модификации кода, программа просто не будет работоспособной. Можно также при обнаружении модификации восстанавливать файл из какой-либо резервной копии, однако эта копия также может быть модифицирована, и сложность метода в таких случаях не оправдывает себя.

Следовательно, необходим метод, который бы позволял сделать модификацию кода невозможной. Но, к сожалению, любой исполняемый файл представляет объектный код, а любой объектный код может быть дисассемблирован и, затем, модифицирован. Выход из положения заключается в использовании архиваторов исполняемых файлов. Принцип работы таких архиваторов заключается в том, что исполняемый файл хранится в заархивированном и зашифрованном виде. Во время запуска информация разархивируется в оперативную память, программа исполняется, и информация снова архивируется. Современные архиваторы исполняемых файлов предоставляют практически полную «прозрачность» доступа, т. е пользователь не ощущает разницы в быстродействии заархивированного и обычного файла.

В системе «Сейф» применяются совокупность обоих методов: во-первых, файл ГМ и другие исполняемые файлы архивируются и шифруются, во-вторых, осуществляется проверка само целостности этих файлов.

Для защиты от «взлома», путем изменения значений системных параметров, для каждого параметра существуют свои методы. Например, взломщик может по истечению срока пробного режима, перевести системную дату назад, таким образом, заставив систему защиты считать, что срок пробного режима еще не истек.

В таком случае может применяться следующий алгоритм: во время каждого запуска программы, записывается дата этого запуска и во время следующего запуска срок пробного режима уменьшается на разность между текущей датой и датой прошлого запуска. Такой алгоритм и подобные ему, для защиты от негативных последствий изменения значений системных параметров применен в системе «СЕЙФ».

РЕАЛИЗАЦИЯ СИСТЕМЫ

Для реализации задач, проанализированных выше, в системе «Сейф» применяется большое число авторских решений, разработанных специально для этой системы, а именно:

Проверка самоцелостности *(Self-Integrity Check, SIC*) основанная на том, что системная информация хранится в двух независимых хранилищах и при обнаружении несоответствия между ними или повреждения информации программа клиента будет немедленно уведомлена. Уведомление предусмотрено и при обнаружении ручного изменения или повреждения ГМ системы.

Защита от удаления системной информации (*System Information Deleting Protection, SIDP*). Благодаря использованию модуля инициализации, удаление системной информации будет обнаружено и оповещена программа клиента.

Самовосстановление *(Self-Restore System, SRS*). Система обладает возможностью восстановления собственной информации. В случае обнаружения повреждения одного из хранилищ клиент может дать команду восстановить его по информации из другого хранилища.

Система полиморфического шифрования *(Polymorphic Encryption System, PES),* обеспечивает изменение методов шифрования в зависимости от времени и обстоятельств. Одна и та же информация в разных обстоятельствах шифруется по-разному.

Система сокрытия информации *(Information Hiding System, IHS*) благодаря которой отследить расположение особо важной информации (например, количество оставшихся дней, в течение которых программа может запускаться в пробном режиме) практически невозможно.

Защита от ручного изменения даты *(Manual Date Changing Protection, MDCP).* За счет предложенного метода ручное изменение системной даты будет сразу же определено и не приведет к увеличению количества оставшихся дней, в течение которых программа может запускаться в пробном режиме.

Защита от повторных инсталляций *(Re-Installing Protection, RIP)* – информация о состоянии программы (зарегистрирована/не зарегистрирована, текущий **U**-код и т. п.) сохраняется и после повторной инсталляции.

СТРУКТУРА СИСТЕМЫ «СЕЙФ»

Система «Сейф» может быть условно разделена на несколько частей, рис.2: главный модуль; вспомогательные программы, входящие в состав системы; справочная система и система технической поддержки**.** Главный модуль системы «Сейф» обеспечивает основные функции. Он встраивается в защищаемое приложение клиента и взаимодействует с ним. Файл главного модуля (*SAFECore.ocx*) занимает всего 282 кбайт, что, удобно при распространении клиентских приложений. Взаимодействие ГМ с клиентским приложением осуществляется через его программный интерфейс.

К вспомогательным программам относятся: генератор *P*-кода; генератор модуля инициализации и общая среда системы. Создание и внедрение модуля инициализации в систему инсталляции клиентского приложения требуется для обеспечения проверки самоцелостности, защиты от удаления системной информации и некоторых других функций.

Рисунок 2 - Компоненты системы «Сейф»

Так как система «Сейф» состоит из большого числа отдельных компонентов, для облегчения работы с ней, в ее состав входит программа «Общая среда», предназначенная для обеспечения быстрого и удобного доступа ко всем компонентам системы «Сейф». Клиенту для получения инструкций и доступа к программе, реализующей нужную функцию необходимо всего лишь выбрать интересующую его задачу из общего списка задач. Ввиду того, что система «СЕЙФ» является достаточно объемным программным продуктом, она оснащена справочной системой и системой технической поддержки, выполненными в формате *HTMLHelp*. Справочная система предназначена для того, чтобы помочь программисту приступить к работе с системой защиты программ «Сейф». В ней изложены процедуры, используемые при выполнении типовых задач. Однако не все проблемы могут быть решены с использованием справки. Иногда у клиента возникает необходимость получить конкретный ответ на возникший вопрос. В таком случае он может воспользоваться системой технической поддержки, позволяющей клиенту, описать проблемную ситуацию, возникшую у него, и, отправив запрос разработчику, получить объяснения.

ВЫВОДЫ

Разработанная система защиты программного обеспечения «Сейф» компактна, занимает сравнительно небольшой объем дискового пространства (3,92 Мбайт) и может быть использована с любыми программами, написанными на языках программирования, поддерживающих технологию ActiveX и концепцию COM (например, Visual C++, Visual Basic, Visual J++, Borland Builder). Благодаря такому подходу система «Сейф» может использоваться для защиты программ, созданных на различных языках программирования.

Система «Сейф» прошла тестирование в нескольких компаниях, занимающихся разработкой программного обеспечения, некоторые фирмы уже используют ее в своей работе, получила первое место на пятом международном форуме «Мир высоких технологий HI-TECH - 2004» и может широко использоваться программистами с целью защиты своего интеллектуального труда.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Дал У., Дейкстра Э., Хоор К. Структурное программирование. – М.: Мир, 1975.
2. Столлингс В. Основы защиты сетей. Приложения и стандарты.: Пер. с англ. – М.: Издательский дом «Вильямс», 2002.
3. Столлингс В. Криптография и защита сетей. Принципы и практика, 2-е изд.: Издательский дом Вильямс, 2001.