УДК 621.397

Гайворонская Г.С., Альфалуджи С.

МНОГОФУНКЦИОНАЛЬНАЯ СИСТЕМА ЗАЩИТЫ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА

Рассмотрен один из вариантов защиты информации с использованием специально разработанной для этой цели многофункциональной системы, позволяющей обеспечить для большинства программ защиту от несанкционированного доступа и копирования; гибкую систему регистрации и возможность работы в пробном режиме.

Ключевые слова: защита, информация, система, доступ.

Постановка задачи. В процессе перехода к информационному обществу все более важной становится проблема защиты информации, составляющей коммерческую тайну. Эта проблема распадается на ряд отдельных задач. Целью этой работы является создание системы, унифицирующей задачу защиты информации. Проблема унификации является комплексной и может быть в данном случае разбита на несколько задач, главной из которых является запрет доступа к информации, минуя систему защиты. Кроме этого обеспечиваются: механизм регистрации, проверка факта регистрации, надежное хранение информации о пользователе и устойчивость к наиболее распространенным методам «взлома». Существует большое количество решений поставленных задач. Одним из возможных вариантов их комплексного решения является разработанная многофункциональная система, с помощью которой можно снабдить пользователей защитой от несанкционированного доступа и копирования, гибкой системой регистрации и возможностью работы в пробном режиме.

Реализация системы защиты. Для обеспечения запрета доступа к клиентской информации, минуя систему защиты (СЗ), предлагаемая СЗ должна являться автономной частью клиентской программы. С одной стороны, необходимо дать клиенту возможность управления и конфигурирования СЗ, с другой – клиент не должен иметь доступа к исходному коду СЗ. Поэтому, ее главный модуль (ГМ) реализован в виде компонента *ActiveX, что* позволяет достичь тесной интеграции системы защиты с защищаемой информацией при отсутствии у клиента прямого доступа к коду компонента. Для того чтобы настроить работу системы, клиент может изменить значения свойств главного модуля. Так как главный модуль 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, главный модуль системы не имеет графического представления в режиме выполнения (*Run time*), и изображается в виде небольшого квадрата в режиме разработки (*Design time*).

Существует несколько вариантов **реализации механизма регистрации**. Самым простым и наиболее распространенным является вариант, при котором каждая копия защищаемой программы обладает уникальным «серийным номером», вводимым для регистрации. 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 ситуации, когда 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 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. Недостаток – возможность использования одного «файла-ключа» с несколькими копиями программы. К тому же остается возможность прямого изменения исходного кода.

Третий вариант используют для защиты наиболее дорогих программ (например, *Discreet 3D Studio MAX, Shlumberger Dino+*). Суть его заключается в том, что на стороне пользователя программа генерирует первичный код (U-код), который пользователь сообщает программисту.

Если программист (разработчик защищаемого ПО) 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 U-коду, программа зарегистрирована.

Основное преимущество метода - код не жестко записывается в программу, а генерируется случайно на стороне пользователя, кроме того, 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 несколько измененный последний вариант. Для обеспечения полиморфизма шифрования, а также для реализации некоторых других функций, каждая программа обладает определенным основанием защиты – числом на основании, которого производятся все операции.

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, используя обратную функцию, генерирует U-код на основании 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 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.

$$A = \sum_{i=1}^{5} \gamma(U_i + 1)(i + \beta), \quad B = \sum_{i=6}^{10} \gamma(U_i + 1)(i + \beta),$$

$$C = \sum_{i=11}^{15} \gamma(U_i + 1)(i + \beta), \quad D = \sum_{i=16}^{20} \gamma(U_i + 1)(i + \beta),$$

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 устойчивости к наиболее распространенным методам «взлома» программ. Существует большое количество разнообразных методов «взлома» программ, отличающихся как по целям, так и по способу их достижения. Среди возможных целей «взлома» можно выделить 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, используемой системой защиты.

Известно, что обеспечить абсолютную защиту нельзя, однако можно довести качество защиты до состояния, при котором «взламывать» программу будет нерентабельно – 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 разные варианты проверки самоцелостности, в частности проверку контрольной суммы, наличия служебных битов, и т. п. Однако такие методы позволяют лишь обнаружить изменение, но не предотвратить его. В лучшем случае, при обнаружении модификации кода, программа просто не будет работоспособной. Можно также при обнаружении модификации восстанавливать файл из резервной копии, однако эта копия также может быть модифицирована, и сложность метода в таких случаях не оправдывает себя.

Следовательно, необходим метод, позволяющий сделать модификацию кода невозможной. Но любой исполняемый файл представляет собой объектный код, а любой объектный код может быть дисассемблирован и, затем, модифицирован. Выход из положения заключается в использовании архиваторов исполняемых файлов. При этом исполняемый файл хранится в заархивированном и зашифрованном виде. Во время запуска информация разархивируется в оперативную память, дешифруется, исполняется, и информация снова архивируется и шифруется. Современные архиваторы исполняемых файлов предоставляют практически полную «прозрачность» доступа, т. е пользователь не ощущает разницы в быстродействии заархивированного и обычного файла.

В предлагаемой системе применяется совокупность обоих методов: файл главного модуля системы и другие исполняемые файлы архивируются и шифруются, а также осуществляется проверка самоцелостности 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, 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 еще несколько подобных ему, применены в предлагаемой СЗ для защиты от негативных последствий изменения значений системных параметров. Для реализации задач, проанализированных выше, в предлагаемой СЗ применяется несколько 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 течении 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 защиты. Система может быть условно разделена на несколько частей, рис. 1: главный модуль; вспомогательные программы, входящие в состав системы; справочная система и система технической поддержки.

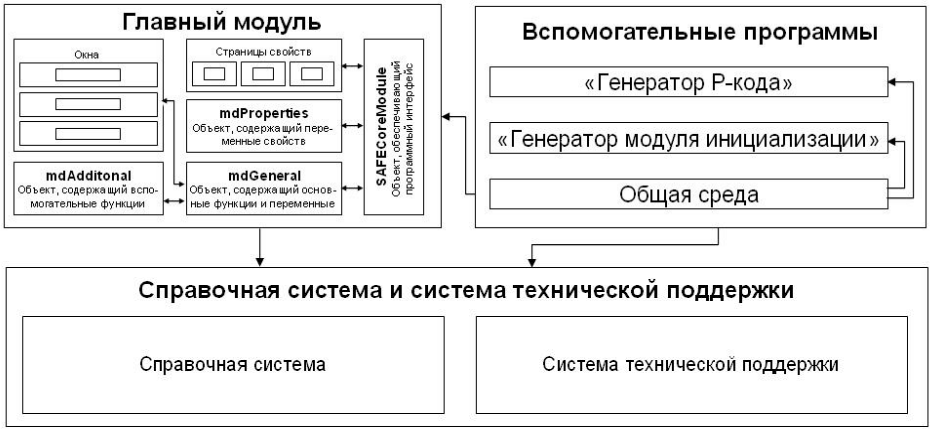

Рис. 1. Компоненты системы защиты

Главный модуль системы 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 Р-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. Так как система состоит из достаточно 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. 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 является достаточно объемным программным продуктом, она нуждается в подробном описании работы с ней и оснащена справочной системой и системой технической поддержки, выполненными в формате *HTMLHelp*.

Выводы. Разработанная система защиты 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 может использоваться для защиты информации разнообразного вида, представленной в различной форме.

ЛИТЕРАТУРА

1. Столлингс В. Основы защиты сетей. Приложения и стандарты.: Пер. с англ. – М.: Издательский дом «Вильямс», 2002.
2. Столлингс В. Криптография и защита сетей. принципы и практика, 2-е изд.: Издательский дом Вильямс, 2001.
3. Гудман С., Хидетниеми С. Введение в анализ и разработку алгоритмов. – М.: Мир, 1981. 368 с.
4. Эпплман Д. Переход на VB.NET: стратегии, концепции, код. – СПб.: Питер, 2002. 464 с.

Надійшла: 20.08.2012р.

Рецензент: д.т.н., проф. Дудикевич В.Д.