ВЫБОР МЕТОДА ЗАЩИТЫ ПРОГРАММНОГО КОДА

Бузукин А.В., Губенко Н.Е.
Донецкий национальный технический университет, Украина
E-mail: buzukin.anton@gmail.com

Аннотация

Бузукин А.В., Губенко Н.Е. Выбор метода защиты программного кода. *В работе рассматриваются способы защиты авторского права, наиболее распространённые методы борьбы с компьютерным пиратством, и защиты на уровне программ методами обфускации. Анализируются основные механизмы и алгоритмы обфускации.*

1 Актуальность работы

Защита кода программы от декомпиляции всегда была, есть и будет актуальной. В первую очередь это относится к коммерческим продуктам, которые, как правило, представляют фирменную тайну для конкурентов. Абсолютной защиты от декомпиляции не существует, но на сегодняшний день у разработчиков программного обеспечения имеется множество программных продуктов, называемых обфускаторами (от английского "obfuscate - ставить в тупик, запутывать"), которые способны усложнить понимание декомпилированного кода.

2 Способы защиты интеллектуальной собственности

Сегодня существуют два основных способа защиты интеллектуальной собственности, и, следовательно, самих программных продуктов:

1) Юридический. Данный способ защиты заключается в создании определенных актов, в соответствии с законом, которые будут охранять интеллектуальную собственность (в нашем случае программные продукты) от нелегального использования. Данный способ включает в себя такие методы как патентование, оформление авторских прав на интеллектуальную собственность и т.д. Также он предусматривает возможность лицензирования программных продуктов (ПП), так, например большинство ПП поставляются вместе с лицензией, которая подтверждает право пользователя использовать этот ПП, то есть, покупая лицензионную копию программы, пользователь в некой мере производит покупку лицензии на право работы с ее копией. Можно выделить два основных вида лицензий на программные продукты:

Временная. Позволяет использовать ПП неограниченному числу пользователей в течение ограниченного периода времени.

Оптимальная. Позволяет использовать ПП ограниченному числу пользователей в течение неограниченного периода времени.

2) Технический. Реализуется путем включения в ПП, какого либо из существующих методов защиты, который будет запрещать его нелегальное использование. По сравнению с юридическим способом защиты ПП, он является наиболее распространенным, так как он практичен, и сравнительно не дорогой в реализации (в дальнейшем, будет приводиться именно его описание) [1].

3 Методы защиты программных продуктов

В настоящее время существует ряд технических способов защиты программных продуктов. Наиболее распространёнными являются такие методы, как:

выполнение на стороне сервера;
водяной знак (software watermark) и отпечаток пальца;
установка подлинности кода (tamper-proofing);

шифрование программного кода.

Выполнение на стороне сервера

Метод выполнения на стороне сервера основан на технологии клиент-сервер, он позволяет предотвратить отсылку кода программы пользователям, которые будут с ней работать, так как сама программа храниться, и выполняется на сервере, а пользователи, используя клиентскую часть этой программы, получают результаты ее выполнения.

Использование цифровых водяных знаков (ЦВЗ)

Использование водяных знаков, основывается на записи в код программы скрытой информации (кому принадлежат авторские права и т.д.), которая позволяет истинному автору программы доказать то, что она является именно его интеллектуальной собственностью (но обычно использование водяных знаков не ограничивается только этим).

Как следует из [3] такая скрытая информация (W), называемая цифровым водяным знаком, должна быть записана в программу P, таким образом, чтобы:

W было надежно расположено (запрятано) в P, и могло быть впоследствии извлечено, без каких либо изменений (повреждений).

W не влияло на работу P.

W несло какую-то определенную информацию, которая позволит доказать, то, что ее присутствие в P неслучайное, то есть является результатом преднамеренных действий [3].

Также для увеличения эффективности водяного знака, его можно записать более чем один раз, в разных местах программы.

Метод установки подлинности кода

При использовании метода установки подлинности кода, в программу помещается процедура проверки целостности самой программы, что позволяет определить, была ли программа изменена (были ли внесены какие-то либо изменения в ее код). Если эта процедура обнаруживает, что в программу внесены изменения, она делает программу не функциональной.

Метод шифрования программного кода

Метод шифрования программного кода предусматривает зашифровывание кода программы, после чего она в зашифрованном виде поставляется конечным пользователям (иногда эффективно зашифровывать только наиболее важные, критические, участки кода, а не весь код программы). Когда пользователь запускает такую программу, вначале будет запущена процедура расшифровки программы, которой потребуется секретный ключ, с помощью которого будет расшифрована запускаемая программа.

Но, как показывает опыт, методы внедрения ЦВЗ и шифрования кода являются недостаточно эффективным для защиты программного кода от исследования посторонними лицами.

4 Использование обфускации для сохранения авторства программного продукта

В большинстве случаев для обхода защиты, взломщику требуется изучить принцип работы ее кода, и то, как она взаимодействует с самой защищаемой программой, этот процесс изучения называется процессом реверсивной инженерии.

Обфускация или запутывание кода — приведение исходного текста или исполняемого кода программы к виду, сохраняющему ее функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию при декомпиляции [2].

Обфускация может применяться не только для защиты ПП, она имеет более широкое применение, например она, может быть использована создателями вирусов, для защиты их творений и т.д.

Суть процесса обфускации заключается в том, чтобы запутать программный код и устранить большинство логических связей в нем, то есть трансформировать его так, чтобы он был очень труден для изучения и модификации посторонними лицами.

Из этого следует, что обфускация одна не предназначена для обеспечения наиболее полной и эффективной защиты программных продуктов, так как она не предоставляет возможности предотвращения нелегального использования программного продукта. Поэтому обфускацию обычно используют вместе с одним из существующих методов защиты (шифрование программного кода и т.д.), это позволяет значительно повысить уровень защиты ПП в целом.

Обфускация соответствует принципу экономической целесообразности, так как ее использование не сильно, увеличивает стоимость программного продукта, и позволяет при этом снизить потери от пиратства, и уменьшить возможность плагиата в результате кражи уникального алгоритма работы защищаемого программного продукта

Существует несколько видов обфускации, подробное их описание можно найти в статье [3].

Первый вид – лексическая обфускация. Он включает в себя: удаление всех комментариев, добавление лишних операций, изменение расположения функций, замена имен идентификаторов, удаление различных пробелов, отступов.

Второй вид – обфускация данных. Её делят на три основные группы: обфускация хранения, соединения и упорядочивания.

Третий вид – обфускация графа потоков управления. Обфускация такого вида осуществляет запутывание потока управления, то есть последовательности выполнения программного кода

5 Алгоритм Колберга

Данный алгоритм оперирует следующими входными значениями:

программа "A" состоящая из исходных или объектных (двоичных) файлов {C1,C2};

стандартные библиотеки, используемые программой {L1,L2};

набор трансформирующих процессов T{T1,T2};

определенный фрагмент кода S, который извлекается из программы A, и который непосредственно будет подвержен трансформации;

набор функций E{E1,E2} которые будут определять эффективность применения определенных трансформирующих процессов {T1,T2} к фрагменту кода S;

набор функций I{I1,I2} которые будут определять важность фрагмента кода S, и в зависимости от этого будут задавать определенное значение переменной RequireObfuscation (чем S важнее тем эта переменная будет хранить большее значение);

две числовые переменные AcceptCost > 0, RequireObfuscation > 0, где первое хранит информацию о доступном максимальном увеличении системных ресурсов по требующихся программе A после того как она подвергнется обфускации, а вторая переменная будет хранить значение требуемого уровня осуществления обфускации (чем важнее фрагмент кода S, тем это значение должно быть больше).

Алгоритм Колберга имеет такую последовательность операций:

1. Загрузка элементов {C1,C2} программы A.
2. Загрузка библиотек {L1,L2}.
3. Осуществление обфускации над программой A, путем выделения фрагмента кода S и определения наиболее эффективного процесса трансформации для него.

Этот этап повторяется до тех пор, пока не будет, достигнут требуемый уровень обфускации RequireObfuscation или допустимое увеличение ресурсов AcceptCost.

4. Генерация трансформируемой программы A`.

Алгоритм Колберга считается общим алгоритмом осуществления процесса обфускации (то есть он не определяет, как именно должен осуществляться, тот или иной метод обфускации) [3].

Chenxi Wang`s алгоритм

В качестве входных данных алгоритм принимает типичную процедуру, написанную на языке высокого уровня. Процесс обфускации каждой такой процедуры состоит из трех этапов:

создание графа потока управления этой процедуры (граф задаётся множеством блоков и множеством связей соединяющих их), после чего граф разбивается, путем замены циклических конструкций в нем на конструкции типа if (условие) goto;

нумерация всех блоков в графе, и добавление в код процедуры переменной хранящей номер следующего выполняемого блока;

приведение графа к однородному (плоскому) виду.

Приведём пример работы обфускатора.

До обфускации:

```
private void CalcPayroll(SpecialList employeeGroup)
{
  while(employeeGroup.HasMore())
  {
   employee = employeeGroup.GetNext(true);
   employee.UpdateSalary(); DistributeCheck(employee);
  }
}
```

Результат после обфускации:

```
private void _1(_1 _2)
{
 while(_2._1())
 {
  _1 = _2._1(true);
  _1._1();
  _1(_1);
 }
}
```

Как видите, понять смысл программы после обфускации стало практически невозможно. В этом и состоит задача обфускации – затруднить для понимания исходный код, запутать и устранить лексические связи в коде.

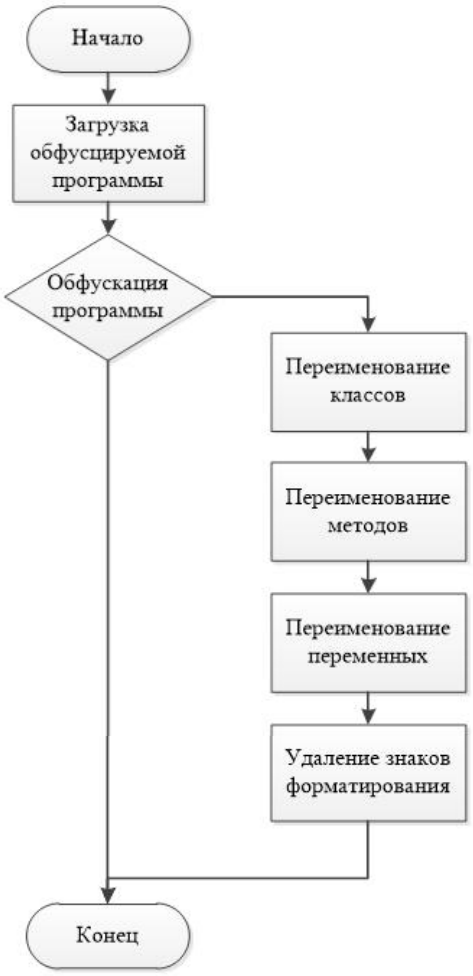

Рисунок 1 – Алгоритм обфускации программы на основе лексического метода обфускации

Выводы

В тезисах представлены основные методы защиты программных продуктов от копирования: юридический и технический. Рассмотрены наиболее распространённые методы защиты программных продуктов, и основные виды обфускации. Полученные результаты будут использованы для создания программного продукта «Обфускатор Java».

Список литературы

1. Интеллектуальная собственность [Electronic resourse] / Интернет-ресурс. - Режим доступа : www/ URL: http://ru.wikipedia.org/wiki/Интеллектуальная собственность. - Загл. с экрана.

2. Определение, основные цели обфускации [Electronic resourse] / Интернет-ресурс. - Режим доступа : www/ URL: http://ru.wikipedia.org/wiki/Обфускация. - Загл. с экрана.

3. Анализ запутывающих преобразований [Electronic resourse] / Интернет-ресурс. - Режим доступа : www/ URL: http://citforum.ru/security/articles/obfus. - Загл. с экрана.

4. Регистрация авторских прав [Electronic resourse] / Интернет-ресурс. - Режим доступа : www/ URL: http://ipstyle.ua/ru/services/copyright. - Загл. с экрана.