Инженерный вестник Дона, №5 (2024)
ivdon.ru/ru/magazine/archive/n5y2024/9185

Алгоритм защиты программного обеспечения от незаконного копирования

А.П. Медведев

Иркутский государственный университет путей сообщения

Аннотация: С развитием цифровизации всех сфер жизни общества кратно возрастает и актуальность разработки нового программного обеспечения, а следовательно и методов его защиты от незаконного копирования и воспроизведения. Основные риски связаны как со взломом готового релиза, так и с утечкой отдельных сегментов кода еще на этапе разработки. При этом шансы утечки напрямую зависят как от количества вовлеченных на разных этапах в процесс разработки специалистов, так и количества самих этапов. Целью данной работы является разработка встраиваемого модуля, направленного на защиту программного обеспечения или отдельных его элементов от незаконного копирования и дальнейшего воспроизведения.
Ключевые слова: защита программного обеспечения, защита информации, встраиваемый модуль защиты, привязка к аппаратным средствам.

Задача защиты программного обеспечения является неотъемлемой часть процесса разработки. Вместе с тем, на текущий момент не существует единого подхода к реализации самих методов защиты программ от незаконного копирования и воспроизведения. Большая часть методов защиты, как правило, базируются на методах скрытия программного кода [1] или привязки программного обеспечения к аппаратной составляющей. Так, в работе [2] описаны два метода привязки программного обеспечения к аппаратному окружению: автоматическая защита и защита при помощи API-функций. В работе [3] рассматриваются подходы с точки зрения общего разделения методов на программные и аппаратные. В работе [4] представлен способ создания программ, скрывающий возможность отслеживая выполнения кода. В работе [5] рассматривается подход, состоящий в использовании фиктивных операций для сокрытия деталей алгоритма, выполняемого процессором, а публикация [6] призывает оценивать алгоритм защиты программного обеспечения с точки зрения разных уровней абстракций.

Нельзя не отметить также опыт защиты программного обеспечения на базе имеющихся криптографических систем защиты информации, описанных в [7] и методов защиты с использованием графических составляющих [8]. Существуют также алгоритмы, направленные на защиту потока обработки данных на основе динамической фрагментации [9]. Не перестает также быть актуальным и классический подход к защите программного обеспечения, на основе скрытия функций [10] и скрытия кода на основе алгоритмов AES [11]. В данной работе рассматривается метод, по своим характеристикам относящийся к семейству методов «Автоматической защиты», описанном в работе [1], имеющий при этом несколько особенностей, существенно повышающих его стойкость к взлому.

Условно, представленный алгоритм защиты программного обеспечения состоит из нескольких этапов:

– этап создания запроса (встраиваемый в код или в качестве отдельного приложения), на основе которого впоследствии будет сгенерирован уникальный ключ;

– этап генерации ключа;

– этап проверки ключа (встраиваемый в программное обеспечение).

В качестве примера реализации алгоритма был выбран язык программирования C#.

В качестве функции преобразования (шифрования) была выбрана функция на основе оператора XOR (исключающее ИЛИ), являющимся обратимой функцией.

Таким образом, функция шифрования будет иметь вид:

```
char SecretFunction(char character, ushort Key){
    character = (char)(character ^ Key);
    return character;}
```

Представленный алгоритм использует несколько наборов ключей – ключ пользователя и два ключа защиты. Ключ пользователя (key_user) предназначен для создания уникального идентификатора (хэша) оборудования конечного пользователя. Ключи защиты, состоящие из двух наборов ключей (secretKey1, secretKey2), необходимы для защиты самого уникального ключа и случайной вставки («балласта»).

Этап создания запроса

В общем виде процесс создания запроса представляет собой сбор идентификаторов выбранного перечня оборудования, к которому осуществляется привязка, шифрование этого набора ключом пользователя и дальнейшая запись в файл запроса.

На рис.1 изображена логическая блок-схема работы алгоритма запроса ключа.

Рис.1. Схема работы алгоритма на этапе запроса ключа

В качестве примера реализации запроса приведем фрагмент кода генерации запроса, основанном на связке серийных номеров материнской платы, жесткого диска и процессора:

```
key_user = 0x0001;                                                    (1)
serial_id=MotherBoardID+VolumeSerial["VolumeSerialNumber"].ToString().Tri
m() + ProcessorID;
foreach (var c in serial_id)
    shifr_id += SecretFunction(c, key_user);
```

Далее в работе будем рассматривать пример, когда полученная последовательность-строка записывается в файл запроса request.dat.

Этап генерации ключа

На этом этапе, в качестве примера, определим следующие переменные:

secretKey1 = 0x0008; (2)

secretKey2 = 0x0022; (3)

string name_add = "текст"; (выбирается произвольно) (4)

string name = "имя"; (задается вручную)

shifr_id = "mu!rushof@18@0DCECGDCGCGG111@1742";

Схематично этап генерации представлен на рис.2.

Рис.2. Схема работы алгоритма на этапе генерации ключа

Шифруем переменную shifr_id двумя ключами и объединяем истинную часть и «балластную»:

```
foreach (var c in shifr_id)
    shifr_id1 += SecretFunction(c, secretKey1);
foreach (var c1 in shifr_id)
    shifr_id2 += SecretFunction(c1, secretKey2);
key = shifr_id1+ shifr_id2;
```

Шифруем имя пользователя (name) и добавляем случайный текст (name_add), шифрованный на другом ключе:

```
foreach (var c3 in name)
    name_tmp1 += SecretFunction(c3, secretKey1);
foreach (var c4 in name_add)
    name_tmp2 += SecretFunction(c4, secretKey2);
name = name_tmp1+name_tmp2;
```

Далее в работе будем считать, что обе переменные были записаны в файл лицензии keyfile.dat, где первая строка – переменная key, вторая – name.

Этап проверки ключа

Итак, файл ключа представляет собой текстовый документ, состоящий из двух строк, где первая строка (key) содержит в себе шифрованную информацию и «балласт», вторая строка (name) – шифрованные имя пользователя и случайный текст.

На этапе проверки ключа производится сбор серийных номеров оборудования, к которому осуществляется привязка и шифрование аналогично этапу создания запроса. Далее производится работа с полученным ключевым файлом на предмет отделения ключевой информации и имени от «балласта» и случайного текста, а затем их дешифрование на ключе защиты (secretKey1). Вычисленные и полученные из файла данные ключей сравниваются, на основании чего делается вывод об истинности соответствия ключа лицензии и аппаратных средств. На рис.3 изображена логическая блок-схема работы алгоритма этапа проверки ключа.

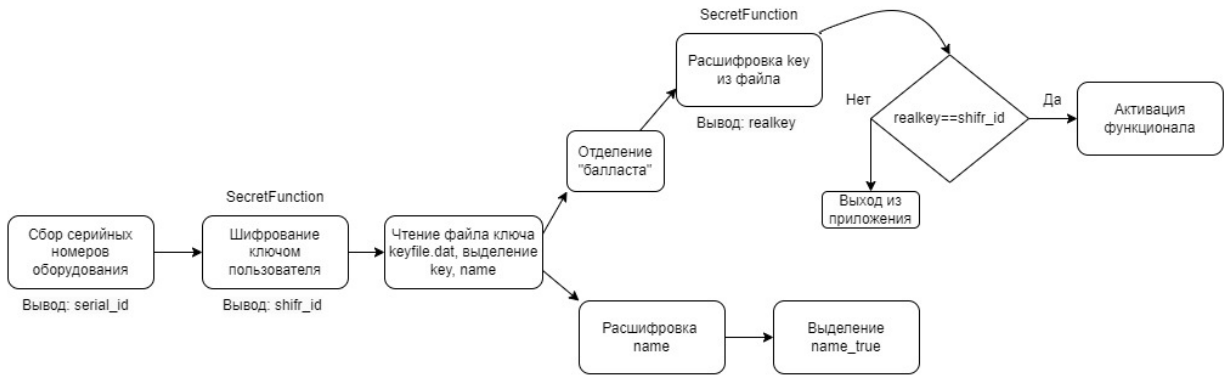

Рис.3. Схема работы алгоритма на этапе проверки ключа

Аналогично алгоритму запроса серийные номера оборудования шифруются ключом юзера (1).

```
serial_id=MotherBoardID+VolumeSerial["VolumeSerialNumber"].ToString().Trim() + ProcessorID;
foreach (var c in serial_id)
    shifr_id += SecretFunction(c, key_user);
```

Считываем из файла keyfile.dat переменную name:

```
string name = File.ReadLines(Application.StartupPath.ToString() + "\\keyfile.dat", Encoding.Default).Skip(1).First();
```

Расшифровываем name из файла:

```
foreach (var c in name)
    realname += TopSecret(c, secretKey1);
```

где realname – переменная, содержащая имя и случайный текст («балласт»).

Отделяем имя от случайного текста:

```
string name_true = realname.Substring(5);
```

где 5-длина переменной текста (4)

Работаем с ключом, считываем из файла:

```
string key = File.ReadLines(Application.StartupPath.ToString() + "\\keyfile.dat", Encoding.Default).Skip(0).First();
```

Отделяем вторую половину («балласт»):

```
string vtor_pol = key.Substring(key.Length / 2);
string per_pol = key.Remove(n, vtor_pol.Length);  //Обрезали первую половину строки (пока еще шифрованную)
```

Расшифровываем первую половину строки:

```
var key_tmp = per_pol.ToArray(); //преобразуем строку в символы
foreach (var c in key_tmp)
    realkey += SecretFunction (c, secretKey1);
```

где realkey – расшифрованный и очищенный от «балласта» ключ, который должен быть равен переменной shifr_id, то есть сборному serial_id оборудования пользователя, зашифрованном на ключе пользователя (key_user).

Проверяем:

```
if (realkey == shifr_id)
    {активация функционала программы}
else
    {выход}
```

Приведенный в работе вариант алгоритма защиты программ может иметь множество модификаций по аппаратному перечню, к которому идет привязка, по длине «балластных» величин, по количеству ключей шифрования и их последовательности. Среди несомненных преимуществ можно отметить быструю скорость внедрения и возможность использовать как самостоятельно, так и в сочетании с другими механизмами защиты.

Литература

1. Behera C., Bhaskari L. Different Obfuscation Techniques for Code Protection // Procedia Computer Science, 2015, Vol. 70, pp. 757-763.

2. Барсуков О.М., Сидоренко И.А., Коренев М.О. Программный модуль защиты информации от незаконного копирования // Актуальные проблемы деятельности подразделений УИС, 2014, С. 43-45.

3. Аверин А.А., Рогожникова Е.Г. Защита программного обеспечения от незаконного копирования // Певзнеровские чтения, 2014, №1, С.5-11.

4. Goldreich O. Towards a Theory of Software Protection (Extended Abstract) // Advances in Cryptology — CRYPTO’ 86. Lecture Notes in Computer Science, 1987, Vol. 263, pp. 426–439.

5. Hollmann H.D.L.,Linnartz J.P.M.G., J.H.van Lint, Baggen C.P.M.J., Tolhuizen L.M.G. Protection of software algorithms executed on secure modules // Future Generation Computer Systems, 1997, Vol. 13, Issue 1, pp. 55-63.

6. Swinson J. Copyright or Patent or Both: An Algorithmic Approach to Computer Software Protection // Harv. JL & Tech, 1991, Vol.5, pp.145-214.

7. Мapo.E.A. Алгебраический анализ стойкости криптографических систем защиты информации // Инженерный вестник Дона, 2013, №4. URL: ivdon.ru/ru/magazine/archive/n4y2013/1996.

8. Панкратов С.А. Использование графической информации для защиты программного и информационного обеспечения // Инженерный вестник Дона, 2012, №2. URL: ivdon.ru/ru/magazine/archive/n2y2012/792.

9. Huang J., Kong X. Research on Data Privacy Security Comprehensive Protection Program Based on Computer Data Protection Algorithm // 2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA), 2023, pp. 325-329.

10. Sander T., Tschudin C.F. On Software Protection via Function Hiding // Information Hiding. IH 1998. Lecture Notes in Computer Science, 1998, Vol. 1525, pp. 111-123.

11. Ismanto R., Salman M. Improving Security Level through Obfuscation Technique for Source Code Protection using AES Algorithm // Proceedings of the 2017 7th International Conference on Communication and Network Security, 2017, pp. 18-22.

References

1. Behera C., Bhaskari L. Procedia Computer Science, 2015, Vol. 70, pp. 757-763.

2. Barsukov O.M.,Sidorenko I.A., Korenev M.O. Aktual'nye problemy deyatel'nosti podrazdeleniy UIS, 2014, pp. 43-45.

3. Averin A.A., Rogozhnikova E.G. Pevznerovskie chteniya, 2014, №1, pp. 5-11.

4. Goldreich O. Advances in Cryptology — CRYPTO’ 86. Lecture Notes in Computer Science, 1987, Vol. 263, pp. 426–439.

5. Hollmann H.D.L.,Linnartz J.P.M.G., J.H.van Lint, Baggen C.P.M.J., Tolhuizen L.M.G. Future Generation Computer Systems, 1997, Vol. 13, Issue 1, pp. 55-63.

6. Swinson J. Harv. JL & Tech, 1991, Vol.5, pp. 145-214.

7. Maro.E.A. Inzhenernyj vestnik Dona, 2013, №4. URL: ivdon.ru/ru/magazine/archive/n4y2013/1996.

8. Pankratov S.A. Inzhenernyj vestnik Dona, 2012, №2. URL: ivdon.ru/ru/magazine/archive/n2y2012/792.

9. Huang J., Kong X. 2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA), 2023, pp. 325-329.

10. Sander T., Tschudin C.F. Information Hiding. IH 1998. Lecture Notes in Computer Science, 1998, Vol. 1525, pp. 111-123.

11. Ismanto R., Salman M. Proceedings of the 2017 7th International Conference on Communication and Network Security, 2017, pp. 18-22.

Дата поступления: 11.03.2024

Дата публикации: 22.04.2024