10. Слепухин А. В., Семенова И. Н. Наполнение содержательно-деятельностной компоненты методики подготовки студентов педагогических специальностей к формированию у обучающихся компетенций цифровой экономики // Педагогическое образование в России. 2020. № 1. С. 87–93.

11. Слепухин А. В., Семенова И. Н., Щербина И. А. Особенности организации самостоятельной работы студентов с использованием облачных технологий в контексте компетентностного подхода // Вестник Томского государственного педагогического университета. 2019. № 3 (200). С. 86–95.

12. Стариченко Б. Е. Цифровизация образования: иллюзии и ожидания // Педагогическое образование в России. 2020. № 3. С. 49–58.

13. Утвержденные ФГОС ВО с учетом профессиональных стандартов 3++ по направлениям бакалавриата, магистратуры, специалитета [Электронный ресурс]. Режим доступа: http://fgosvo.ru/fgosvo/151/150/24 (дата обращения: 15.09.2020).

Ставских Артемий Денисович
ФГБОУ ВО «Курганский государственный университет»,
студент кафедры «Безопасность информационных и автоматизированных систем»,
artemiy.st@mail.ru, Курган, Россия
Ревняков Евгений Николаевич
ФГБОУ ВО «Курганский государственный университет»,
канд. техн. наук, доцент кафедры «Безопасность информационных и автоматизированных систем»,
aphaline@mail.ru, Курган, Россия
Человечкова Анна Владимировна
ФГБОУ ВО «Курганский государственный университет»,
старший преподаватель кафедры «Безопасность информационных и автоматизированных систем»,
chelovechkova_2011@mail.ru, Курган, Россия

ВИДЫ ВЗЛОМА И СПОСОБЫ ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

УДК 004.056.53

Аннотация. Данная статья содержит информацию по основным методам взлома программного обеспечения. К каждому представленному методу предлагаются меры противодействия.
Ключевые слова: взлом, шифрование, электронная подпись, обфускация.

Abstract. This article contains information on the basic methods of hacking software. For each presented method countermeasures are proposed.
Keywords: hacking, encryption, electronic signature, obfuscation.

Прибыль, получаемая с продажи лицензий на программное обеспечение, является основной частью заработка разработчиков. Пользователи, в свою очередь, не всегда готовы платить, и появляются люди, взламывающие приложения. Из этого следует важность защиты программы от взлома. Однако стоимость систем безопасности не всегда является доступной для разрабатывающей компании. Но и оставлять программы без защиты не следует. В данной статье приведены минимальные меры по защите программ от взлома, которые можно применить, когда нет средств на платные системы защиты.

Прежде всего, рассмотрим основные виды взлома программного обеспечения [2; 3].

1. Подбор серийного номера – перебор нелегально полученного списка возможных ключей активации, пока не будет найден действующий ключ. Если такого списка нет, и процесс подбора ключа активации автоматизированный (брутфорс), то используют специальные программы, перебирающие все возможные серийные номера.

2. Генерация серийного номера – генерация ключа активации с помощью специальной программы (*keygen*). Для создания такой программы необходимо провести обратную разработку, взламываемой программного обеспечения, и выяснить алгоритм генерации серийного номера.

3. Взломанные файлы программы – замена оригинальных файлов программы на взломанные файлы. Для автоматизации создаются специальные программы (бинарные патчи). Патчи находят и заменяют в файлах

программного обеспечения байты кода, отвечающие за лицензию, тем самым снимая защиту.

4. Эмуляция ключа – использование нелегально полученного виртуального токена, прошитого лицензией на взламываемое программное обеспечение. Взлом осуществляется путем снятия дампа с лицензионного токена и эмуляции его подключения, с помощью специальных программ.

5. Ложный сервер проверки лицензии программы – подмена сервера, проверяющего лицензию программного обеспечения. Данный вид взлома требует изучения логики программы (обратная разработка) и механизма проверки лицензии программы сервером. Саму подмену сервера осуществляют с помощью использования файла «hosts» и развертывания локального сервера (или использования существующего IP-адреса).

Проанализировав основные виды взлома, были сформированы меры и рекомендации по уменьшению вероятности их возникновения.

1. От автоматизированного подбора серийного номера (брутфорса) можно защититься временной задержкой ввода после определенного количества некорректных попыток. Также можно ограничить количество попыток ввода пароля. Сам серийный номер не следует хранить в незашифрованном виде. Пример шифрования и дешифрования на C++:

```
BYTE inoutBuffer [64] = "const serial number 2309-3789-6591-8459";
DWORD bufferLength=64;
HCRYPTPROV cryptHandler;
CryptAcquireContextW (&cryptHandler, NULL, NULL, PROV_RSA_FULL, 0);
HCRYPTKEY keyHandler;
CryptGetUserKey (cryptHandler, AT_SIGNATURE, &keyHandler);
CryptGenKey (cryptHandler, CALG_RC4, CRYPT_EXPORTABLE, &keyHandler);
//шифруем inoutBuffer:
CryptEncrypt (keyHandler, NULL, TRUE, CRYPT_OAEP, inoutBuffer, &bufferLength,
bufferLength);
//расшифровываем inoutBuffer:
```

```
CryptDecrypt (keyHandler, NULL, TRUE, CRYPT_OAEP, inoutBuffer, &bufferLength);
```

Используя приведенный выше пример, можно защитить лицензионные данные программного обеспечения.

2. Генерацию серийного номера сторонней программой можно предотвратить, усложнив процесс формирования серийного номера. Например, использовать для формирования быстро изменяющиеся значения (такие как время) и/или адреса важных объектов программы в оперативной памяти (даже если злоумышленник получит исходный код генерации серийного номера, он никак не сможет узнать адреса объектов).

3. Чтобы избежать использования взломанных файлов, нужно встроить в программу модуль, периодически проверяющий состояние файлов программы (контрольную сумму). Чем сложнее устроена проверка, тем сложнее будет нейтрализовать этот модуль. Пример проверки файла на C++:

```
HCRYPTPROV cryptHandler;
CryptAcquireContextW (&cryptHandler, NULL, NULL, PROV_RSA_FULL, 0);
//формирование хэша (fileBuffer – файл, загруженный в оперативную память):
HCRYPTHASH hashHandler;
CryptCreateHash (cryptHandler, CALG_MD5, 0, 0, &hashHandler);
CryptHashData (hashHandler, fileBuffer, fileBufferLength, 0);
//формирование подписи хэша:
DWORD signatureLength = 0;
CryptSignHashA (hashHandler, AT_SIGNATURE, NULL, 0, NULL, &signatureLength);
BYTE *signature = new BYTE[signatureLength];
CryptSignHashA (hashHandler, AT_SIGNATURE, NULL, 0, signature, &signatureLength);
HCRYPTKEY keyHandler;
CryptGetUserKey (cryptHandler, AT_SIGNATURE, &keyHandler);
/*повторно сформировать хэш файла*/
//проверка подписи:
if (CryptVerifySignatureA (hashHandler, signature, 128, keyHandler, NULL, 0)) std: cout <<
"good signature";
```

```
else std: cout << "bad signature";
```

В программе должна храниться лишь подпись от хэша всех необходимых файлов. Периодически нужно хэшировать эти файлы и проверять сигнатуру. Хранить подпись в открытом виде не следует. Злоумышленнику не составит труда заменить ее на сигнатуру хэша от взломанных файлов.

Если для проверки подписи разработана булева функция, то такую защиту будет легко сломать – подменить исходную функцию на функцию, всегда возвращающую истинное значение. Поэтому рекомендуется со случайной вероятностью производить проверку ложного хэша, либо отказаться от булевой функции.

4. Для программ, использующих электронные ключи (токены), возможен взлом с помощью эмуляции лицензионного ключа. Чтобы защититься от этого, необходимо грамотно настроить конфигурацию ключей, а именно – запретить чтение ячеек ключа, содержащих лицензию, а также чтение алгоритмов, обрабатывающих эти ячейки.

5. Защитить программу от активации ложным сервером можно, усложнив взаимодействие сервера с приложением и проверку лицензирования. Кроме того, если IP-адрес сервера не планируется изменять, можно использовать его в программе вместо полного доменного имени. Таким образом, нельзя будет средствами операционной системы переопределить IP-адрес сервера, к которому обращается программа.

Чтобы сократить шансы злоумышленника на взлом программного обеспечения часто используют обфускацию. Хоть это и не дает гарантированную защиту от взлома, но может значительно затруднить обратную разработку. Существует множество платных и бесплатных обфускаторов. Их все можно разбить на три группы: работающие с исходным текстом; работающие с исполняемыми файлами; работающие, как с исходным кодом, так и с машинным. В качестве примера можно привести бесплатный

обфускатор, работающий с исходным кодом (со строками) – Coder's Text Obfuscator.

Исследуем программу, исходный код которой был приведен в примере шифрования/дешифрования на C++, с помощью hex-редактора. Как видим, строка с серийным номером никак не скрыта (рис. 1).

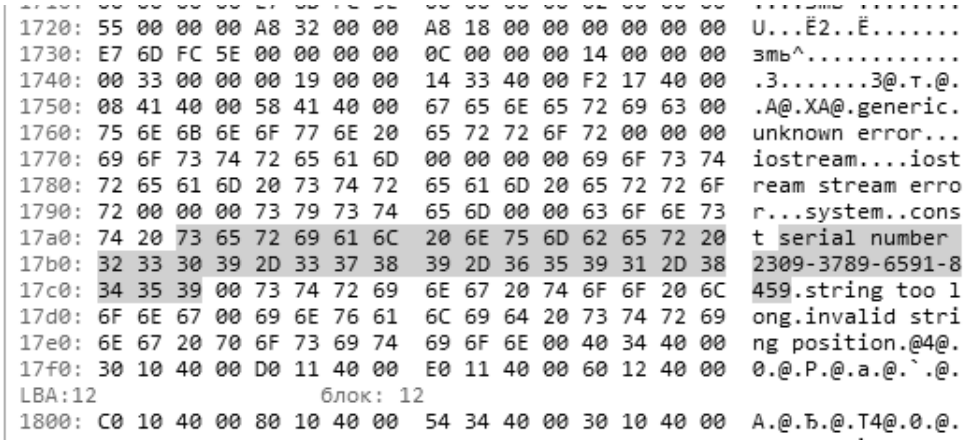

Рис. 1. Исследование программы hex-редактором

Преобразуем строку через Coder's Text Obfuscator (рис. 2). На выходе получаем исходный код, который сгенерирует строку при выполнении программы.

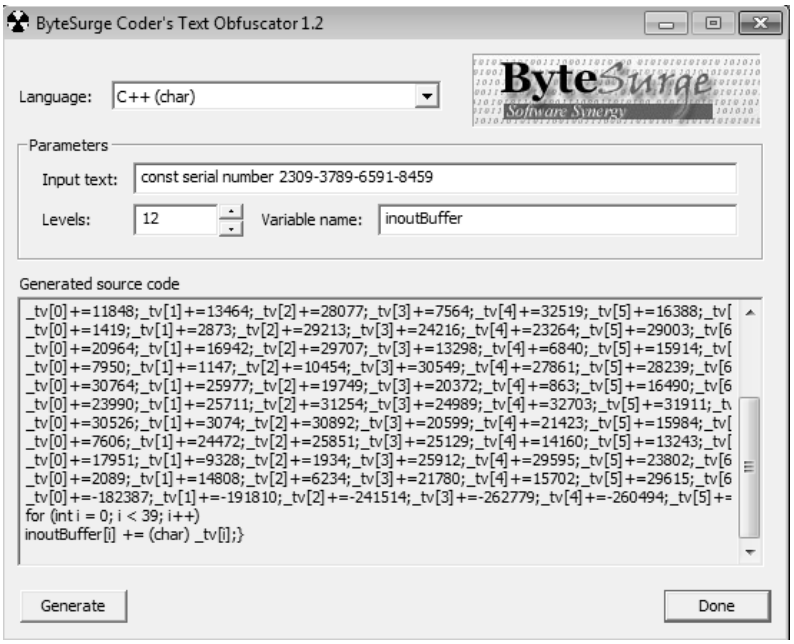

Рис. 2. Обфускация строки с помощью Coder's Text Obfuscator

Проверим hex-редактором новый исполняемый файл на наличие серийного номера. Искомой строки, как и ожидалось, не найдено.

Разумеется, в программном обеспечении, разработанным на коммерческой основе, никто не оставит серийный номер или какой-либо пароль в открытом виде. Обфускация строк имеет смысл, если рядом в коде производится действие, требующее сокрытия от злоумышленника. Например, проверка лицензии программы сопровождается сообщениями о том, что лицензия закончилась, либо функция приложения недоступна в демоверсии, и т.д. В таком случае по строке с сообщением злоумышленник может найти часть, отвечающую за лицензию, и взломать программное обеспечение.

С помощью приведенных рекомендаций можно уменьшить вероятность взлома программного обеспечения, и тем самым увеличить шансы на получение прибыли разрабатывающей компанией.

Список использованной литературы

1. Зырянов А. Н., Москвин В. В. Защищенность современных мессенджеров // Безопасность информационного пространства: статья в сборнике трудов конференции «Безопасность информационного пространства» 30 ноября – 1 декабря 2016 г. Курган: Курганский государственный университет, 2016. С. 132–134.

2. Взлом программ: сайт. URL: https://www.anti-malware.ru/threats/programs-crack (дата обращения: 23.06.2020).

3. Взлом программ (крэкинг): сайт. URL: https://it-black.ru/kreking/ (дата обращения: 23.06.2020).

4. wincrypt.h header – Win32 apps | Microsoft Docs: сайт. URL: https://docs.microsoft.com/en-us/windows/win32/api/wincrypt/#functions (дата обращения: 25.06.2020).