Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра комплексной информационной безопасности электронно-

вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ

Заведующий каф. КИБЭВС

\_\_\_\_\_\_\_\_\_ А.А. Шелупанов

\_\_\_\_.\_\_\_\_.\_\_\_\_\_\_\_\_ дата

СИСТЕМА DRM-ЗАЩИТЫ DATAWALL

Курсовая работа по дисциплине ««Программно-аппаратные средства обеспечения информационной безопасности»

Пояснительная записка

Студент гр. 727-1

\_\_\_\_\_\_\_\_\_ Усольцев Я.А.

Принял:

Доцент кафедры КИБЭВС

\_\_\_\_\_\_\_\_ И.А.Рахманенко

\_\_\_\_\_\_\_\_

Реферат

Пояснительная записка содержит листов, рисунков, таблиц, использованных источников, приложений.

DRM, ШИФРОВАНИЕ,

Полное название системы: система DRM-защиты DataWall.

Краткое наименование системы: DataWall

Цель работы – разработка системы DRM.

В результате работы было создано приложение, которое корректно выполняет свои функции.

Разработка системы DRM-защиты производилась на языках программирования SQL, С#, С, С++, в качестве СУБД использовалась СУБД MySQL, в качестве интегрированной среды разработки использовались средства и службы для разработчиков Visual Studio 2019, в качестве среды функционирования использовалась операционная система Windows 10, в качестве вспомогательных интсрументов разработки использовались набор SIMD-инструкций процессора на базе архитектуры x86-64, набор криптографических инструкций процессора на базе архитектуры x86-64, библиотека реализации протокола SSL – OpenSSL, библиотека работы с исполняемым кодом – MemoryModule.

Пояснительная записка к курсовой работе выполнена в текстовом редакторе Microsoft Word 2016.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ

Заведующий кафедрой КИБЭВС

д-р техн. наук, профессор

\_\_\_\_\_\_\_\_\_\_\_\_А.А. Шелупанов

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

ЗАДАНИЕ

На курсовую работу по дисциплине «Программно-аппаратные средства обеспечения информационной безопасности» студенту Усольцеву Якову Александровичу группы 721 факультета Безопасности

1 Тема работы: «Система DRM-защиты DataWall».

2 Срок сдачи студентом законченной работы: 8 июня 2021г.

3 Исходные данные к работе:

- операционная система Windows 10;

- набор SIMD-иструкций процессора на базе архитектуры x86-64

- набор криптографических иструкций процессора на базе архитектуры x86-64

- библиотека реализации протокола SSL – OpenSSL

- библиотека работы с исполняемым кодом – MemoryModule

- Реляционная СУБД MySQL

4. Содержание пояснительной записки:

- титульный лист;

- реферат;

- техническое задание;

- введение;

- обзор предметной области;

- проектирование архитектуры системы

- обоснование выбора программных средств;

- описание модулей прикладной программы;

- тестирование;

- заключение;

- список литературы;

- приложения.

5. Перечень подлежащих разработке вопросов:

- Разработка службы, устанавливающейся в системе клиента. Служба должна запускаться автоматически при старте клиентского ПК и выполнять следующие функции: обмен информацией с сервером; получение сведений о ПК клиента; генерация ключей шифрования; установка в систему защищаемого ПО с сопутствующим шифрованием его данных и исполняемого кода; периодический контроль целостности защищаемого ПО; обслуживание запросов защищаемого ПО.

- Разработка серверного приложения. Серверное приложение запускается на сервере провайдера услуг защиты и выполняет следующие функции: работает с базой данных, содержащей аутентификационную информацию клиентов, сведения о принадлежащих им лицензиях на защищаемого ПО и об их устройствах, а также предоставляет вышеперечисленную информацию по запросу клиентских служб.

- Разработка SDK для разработчиков защищаемого ПО и документации к нему.

6. Задание выдано:

Руководитель: к.т.н., доцент кафедры БИС

И.А. Рахманенко

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

(подпись)

7. Задание принято к исполнению:

Студент группы 727-1

Я. А. Усольцев

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

(подпись)

Содержание

[1 Введение 7](#_Toc73904137)

[2 Обзор предметной области 8](#_Toc73904138)

[3 Проектирование архитектуры системы 11](#_Toc73904139)

[4 Обоснование выбора программных средств 14](#_Toc73904140)

[5 Описание модулей прикладной программы 16](#_Toc73904141)

[6 Описание алгоритмов работы 33](#_Toc73904142)

[7 Тестирование 39](#_Toc73904143)

[Заключение 44](#_Toc73904144)

[Список литературы 45](#_Toc73904145)

[Приложение А 46](#_Toc73904146)

[Приложение Б 52](#_Toc73904147)

[Приложение В 59](#_Toc73904148)

[Приложение Г 64](#_Toc73904149)

# 1 Введение

Борьба правообладателей с нелегальным копированием ведётся давно, и не только силами закона, но и с помощью технических ухищрений, которые внедряются непосредственно в саму продукцию. Тем не менее, практически любой контент, будь то игры, софт, видео или музыкальные треки, можно найти и скачать в сети бесплатно, а потому возникает вопрос об эффективности таких технических средств защиты.

Потребность защищать свой авторский труд от массового копирования возникла с появлением самих возможностей для такого копирования, примерно в70-е гг. прошлого века, когда на рынке появились первые бытовые пишущие видеомагнитофоны от Sony, что очень сильно взволновало кинокомпанию Universal, поскольку теперь каждый владелец такого магнитофона был в состоянии записать произведенный ими фильм.

С появлением компьютерных игр все изменилось, поскольку компьютерные игры, в отличии от видеокассет, можно было перезаписывать сколько угодно раз без потери качества, что позволяло пользователям делать вполне сносные копии и распространять их. С этого момента для борьбы с пиратством начали использовать лицензионный код.

С каждым годом DRM-системы изменялись и модифицировались под потребности правообладателей. Появились программные, аппаратные и смешанные DRM. В том или ином виде DRM-системы сегодня устанавливаются во всех типах устройств: капсульные кофеварочные машинки, кошачьи лотки с картриджами, принтеры и т.д. Теперь это не сколько способ защиты авторского права, но и способ контроля использования продукта производителем.

# 2 Обзор предметной области

Наиболее известны сегодня DRM, созданные для защиты авторских прав производителей видеоигр. Первые DRM требовали ввода специального кода. Лицензионный код в том или ином виде дошел и до наших дней. Раньше его не обязательно нужно было вводить только при установке или запуске, например, в Zak McKracken and the Alien Mindbenders, выпущенном в 1988 квест-игре, пятизначный код лицензии надо было вводить во время решения очередной головоломки.

В 1980-х появилась система Система Lenslok, посредством которой защищались игры и программы для компьютеров Spectrum, Atari и пр. Система работала следующим образом: вместе с игрой поставлялась пластиковая линза, посредством которой на начальном изображении необходимо было рассмотреть код. Данная система не прижилась, поскольку многие пользователи не смогли рассмотреть код из-за большой диагонали телевизора, изображение не помещалось в линзу.

С приходом CD-накопителей и игр на данных носителях пиратство немного затормозилось, поскольку относительно недорогих пишущих CD-приводов на рынке еще не было. Они подешевели в середине 90-х, и примерно в это время крупные компании начинают разработку технических средств защиты от копирования. В это время появилось название таких систем защиты: Digital Rights Management, сокращенно DRM. Первый технический способ борьбы с массовым копированием заключался в нарушении стандарта Audio CD. Большинство проигрывателей не замечало это нарушение и воспроизводило дорожку, однако компьютерные CD-ромы их уже не считывали.

Сегодня DRM это целый набор технологии для создания ограничений при использовании оборудования или программ, защищенных авторским правом. Данные ограничения касаются не только копирования, но и перепродажи оригинального продукта и т.д.

DRM-решения можно разделить на программные и на аппаратные. К программным можно отнести лицензионные ключи, ограничение на количество установок, и активации, онлайн-аутентификация, удаление скачанной музыки после окончания подписки или использование уникального формата данных.

Аппаратные решения — это, например, шифрование Blue-Ray или DVD дисков с возможностью их расшифровки только на плеерах с установленной этой же технологией. Подобную технологию использует компания HP, которая поддерживает на своих принтерах работу только оригинальных картриджей.

Комплексные решения разрабатывали не только сами создатели софта, появились целые IT-компании, занимающиеся исключительно созданием систем защиты, например решение StarForce от компании Protection Technology.

Технология StarForce осуществляла контроль наличия диска в дисководе, проверяла серийный ключ, осуществляла онлайн-активацию, а также проверку CD-диска на оригинальность. Кроме того, данное решение устанавливало в системе свои драйвера, которые мешали работе эмуляторов, таких как Alcohol 120%. Минусами такой технологии стали:

- износ CD-диска из-за постоянных его проверок на оригинальность и наличие, поскольку это приводило к его физическому износу и система в дальнейшем переставала его распознавать;

- наличие собственной файловой системы сильно занимало ресурсы ПК, которые в то время были достаточно маломощными;

- онлайн-активация с сохранением конфигурации ПК на сервере с БД пользователей приводила к тому, что при обновлении комплектующих запустить игру становилось невозможно.

Тем не менее, данная система устраивала издателей, поскольку игры, вышедшие с такой защитой, в среднем взламывались не менее чем через год после выхода, что позволяло им окупиться, например, Splinter Cell: ChaosTheory от Ubisoft продержалась 422 дня.

Со временем DRM- системы модифицировались и становились более эффективными. Компания Sony создала для своих дисков защиту с помощью битых секторов. При перезаписи с оригинальной болванки эти биты сектора переноситься не будут, однако в последствии такая копия не будет воспроизводится на консоли, поскольку она будет слишком идеальна без этих самых битых секторов.

Огромный вклад в развитие DRM-систем внесли такие игровые сервисы, как Steam и battle.net. их сервис позволяет правообладателям получить контроль над санкционированностью доступа к разработанному ими ПО путем ведения онлайн-аутентификации и ограничению доступа без запуска сервиса. Таким образом, все ПО, имеющее DRM-защиту от Steam, не будет работать без запуска Steam-клиента. Такого рода DRM-защита, в отличие от StarForce, оказалась намного эргономичнее для конечных пользователей и по сей день.

Используемая наиболее известными сервисами потокового вещания система DRM –защиты – White Wine. Основная ее функция – налаживание шифрованного канала между сервером вещания и конечным приложением для воспроизведения на ПК пользователя со встроенным модулем White Wine.

В области, отличной от IT, также применяются системы DRM –защиты. Производитель сельскохозяйственной техники John Deere использует в своих тракторах электронные устройства, ремонт которых возможен только в авторизированных сервисных центрах или у производителя.

DRM-защита от пользователя также будет у производителя автомобилей BMW, поскольку данная компания анонсировала выпуск автомобилей с подпиской на дополнительные опции. Соответственно, чтобы опции автоматически становились доступны пользователю, необходимо их заранее установить в автомобиле, а после защитить от взлома.

# 3 Проектирование архитектуры системы

Для обеспечения работы клиентской части приложения реализованы следующие компоненты системы:

1 Приложение-служба Windows – DataWallService. Данное приложение запускается автоматически при старте клиентского ПК и создает именованный канал, к которому подключается оконное приложение для комфортного взаимодействия с клиентом. Для выполнения задач, связанных с установкой, запуском и защитой ПО используются соответствующие библиотеки, которые будут описаны далее;

2 Оконное приложение – DataWallClient. Приложение запускается клиентом и подключается посредством именованного канала к службе. Данное приложение получает команды клиента, передает их службе для выполнения, а также получает от службы результат их выполнения и отображает в понятном для пользователя виде.

3 Динамическая библиотека загрузки – DataWallLoader. Данная библиотека используется для загрузки защищенных компонентов ПО, предварительно расшифровывая их сгенерированным ключом. Данная библиотека вызывается непосредственно защищаемым ПО, а ключ из службы передается по специальному защищенному именованному каналу, уникальному для каждого из защищаемых средств.

4 Динамическая библиотека движок – DataWallEngine. Данная библиотека содержит в себе все основные функции системы, такие как получение информации о текущей конфигурации системы, аутентификация и отправка системной конфигурации на сервер, получение пользовательской библиотеки, генерация ключа для заданного приложения, установка приложений, рассчет и проверка хэша приложения, а также упаковка приложения в контейнер, соместимый с DataWallLoader. Библиотека вызывается по запросам системной службы и возвращает результаты ее работы.

Со стороны сервера система представлена следующими компонентами:

1 Динамическая библиотека движок – DataWallEngine.

2 База данных - содержит аутентификационную информацию клиентов, сведения о принадлежащих им лицензиях на защищаемого ПО и об их устройствах.

3 Серверное приложение – DataWallServer. Данное приложение: обслуживает запросы от клиентов, позволяет просматривать и редактировать содержимое БД, а также выводит в лог информацию о работе системы.

Таким образом, структура системы представлена на рисунке 3.1.

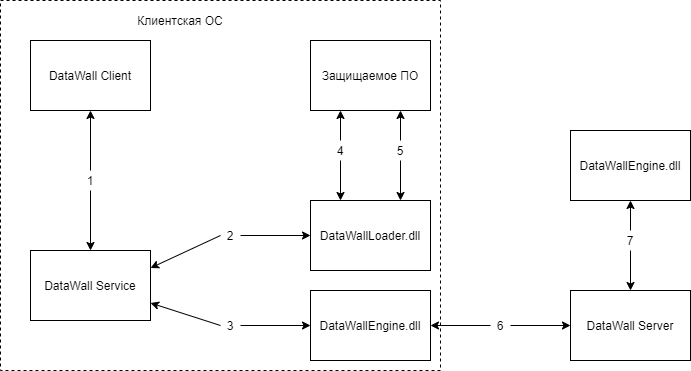


Рисунок 3.1 – Структура системы

На данной схеме связи имеют следующее значение:

1 Именованный канал между клиентским приложением и Windows-сервисом. По данному каналу от клиентского приложения отправляются коды действий, которые необходимо выполнить и, при необходимости, дополнительная информация (коды ошибок и структуры сообщений описаны в разделе 6 «Описание алгоритмов работы»), а от сервиса – коды результата выполнения команды и информация, которую необходимо отобразить.

2 Вызовы службой функций из динамической библиотеки DataWallLoader.

3 Вызовы службой функций из динамической библиотеки DataWallEngine.

4 Защищенный именованный канал между библиотекой DataWallLoader, загруженной службой и библиотекой DataWallLoader, загруженной защищаемой ПО. Данный именованный канал служит для передачи ключа шифрования, сгенерированного библиотекой DataWallEndine в службе для расшифрования защищенных компонентов ПО. Именованный канал защищен посредством 1024-битного алгоритма Диффие-Хэллмана.

5 Вызовы защищаемым ПО функций из динамической библиотеки DataWallLoader.

6 Защищенное интернет соединение между системной службой (устанавливается посредством библиотеки DataWallEndine) и сервером DataWallServer. Соединение защищено протоколом TLS версии 2.1.

# 4 Обоснование выбора программных средств

В качестве языка программирования для реализации основного функционала системы был выбран язык C++. Данный выбор обоснован следующими факторами:

1 Данный язык позволяет использовать возможности параллельных вычислений процессора (SIMD-инструкции), а также аппаратные средства шифрования (инструкции AES шифрования);

2 Данный язык позволяет реализовывать системное программное обеспечение;

3 C++ позволяет реализовывать высокопроизводительное ПО, что критично в случае разработки систем DRM-защиты, поскольку данные системы не должны негативно влиять на производительность защищаемого ПО;

4 Библиотеки, написанные на языке C++ в семантике языка C совместимы с большинством других языков и могут использоваться защищаемым ПО, написанным практически на любом компилируемом и некоторых интерпретируемых языках программирования;

5 Язык C++ позволяет выполнять низкоуровневые операции над памятью, что необходимо для реализации безопасной загрузки ПО.

В качестве языка программирования для реализации серверного ПО и клиентского приложения выбран язык C#. Данный выбор обоснован следующими факторами:

1 Данный язык позволяет быстро реализовывать оконные приложения Windows;

2 Данный язык совместим с системным ПО, написанным на языке C++;

3 Данный язык обладает встроенными средствами реализации безопасных сетевых протоколов, что значительно упрощает создание серверных приложений.

В качестве языка программирования запросов БД был выбран SQL, поскольку данный язык специально разработан для создания, модификации и управления данными в реляционной базе данных, управляемой системой управления базами данных.

Для упрощения разработки системы использовалась кроссплатформенная система автоматизации сборки программного обеспечения из исходного кода, которая позволяет отказаться от использования решений Visual Studio, основной проблемой которых являются проблемы совместимости при разворачивании на устройстве с другой версией Visual Studio или вообще без установленной VS, а также решения VS являются очень тяжеловесными.

# 5 Описание модулей прикладной программы

5.1 Библиотека-движок DataWallEngine

Данный модуль содержит в себе все основные функции системы, разворачиваемые на клиентской машине. Далее следует описание функций системы:

1 InitializeEngine – функция инициализации движка. Выполняет инициализацию системных COM-объектов, необходимых для получении конфигурации системы, инициализацию системных сокетов и безопасно подключается к серверу, также в случае необходимости инициализирует логирование. В случае сборки в конфигурации RELEASE\_ONLY в целях безопасности логирование отключено на уровне препроцессора, поэтому даже в случае активации при инициализации движка логи вестись не будут.

Данную функцию следует вызывать при старте работы приложения, использующего движок системы, поскольку без инициализации большинство функций движка не будут доступны и будут завершаться с кодом ошибки.

Входные параметры:

- const char\* logfile – имя файла ведения логов, в случае если лог вести не нужно, следует передать NULL;

- bool log\_to\_stdout – флаг, указывающий есть ли необходимость выводить сообщения в консоль приложения;

- const char\* server\_addr – адрес сервера DataWall, в случае если требуется использование без подключения к серверу, следует передать NULL, но функции требующие сетевого взаимодействия с сервером доступные будут;

- UINT16 port - порт сервера DataWall, в случае если server\_addr установвлен в NULL - значение port игнорируется;

Возвращаемый тип – WinAPI статус HRESULT [8].

2 UninitializeEngine – функиция разинициализации движка. Выполняет разинициализацию системных COM-объектов, необходимых для получении конфигурации системы, отключается от сервера и выполняет разинициализацию системных сокетов, также в случае необходимости разинициализирует логирование. Данную функцию следует вызывать при завершении работы приложения.

Функция не имеет входных параметров.

Возвращаемый тип – WinAPI статус HRESULT.

3 GetSystemConfiguration – функция получения системной конфигурации. Для успешного выполнения данной функции необходимо инициализировать движок. Данную функцию рекомендуется вызывать перед получением непосредственно информации об установленных в системе устройствах, для того чтобы выделить необходимое количество памяти для данной системной информации.

Входные параметы:

- UINT8& MBs – ссылка на переменную, в которую следует записать число установленных в системе материнских плат

- UINT8& CPUs – ссылка на переменную, в которую следует записать число установленных в системе процессоров

- UINT8& GPUs – ссылка на переменную, в которую следует записать число установленных в системе графических ускорителей.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция записывает в предоставленные переменные найденные значения конфигурации системы.

4 GetMotherboardInfo – функция получения информации об установленных в системе материнскых платах. Для успешного выполнения данной функции необходимо инициализировать движок.

Входные параметры:

- MotherboardInfo\*& info – ссылка на выделенную память массива структур, описывающих материнскую плату, в который необходимо записать полученную информацию;

- UINT8 MBs – количество элементов в массиве, под которые выделена память.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция записывает в предоставленный массив полученную информацию.

5 GetProcessorInfo - функция получения информации об установленных в системе центральных процессорах. Для успешного выполнения данной функции необходимо инициализировать движок.

Входные параметры:

- ProcessorInfo\* &info – ссылка на выделенную память массива структур, описывающих процессор, в который необходимо записать полученную информацию;

- UINT8 CPUs - количество элементов в массиве, под которые выделена память.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция записывает в предоставленный массив полученную информацию.

6 GetVideoAdapterInfo - функция получения информации об установленных в системе центральных процессорах. Для успешного выполнения данной функции необходимо инициализировать движок.

Входные параметры:

- VideoAdapterInfo\* &info – ссылка на выделенную память массива структур, описывающих графический ускоритель, в который необходимо записать полученную информацию;

- UINT8 GPUs - количество элементов в массиве, под которые выделена память.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция записывает в предоставленный массив полученную информацию.

7 NetworkAuthentication – функция сетевой аутентификации. Аутентификация производится логином и паролей. Для успешного выполнения данной функции необходимо инициализировать движок с указанием сервера. Следует иметь ввиду, что для выполнения всех взаимодействий с сервером клиент должен быть авторизован, в противном случае сервер будет отвергать клиентские запросы.

Входные параметры:

- const char\* nickname – имя пользователя;

- const char\* password – пароль.

Возвращаемый тип – WinAPI статус HRESULT.

8 SendDeviceConfiguration – функция отправки системной конфигурации на сервер. Для успешного выполнения данной функции необходимо инициализировать движок с указанием сервера и пройти аутентификацию.

Входные параметры:

- const char\* mb – сведения о материнской плате;

- const char\* cpu – сведения о процессоре;

- const char\* gpu – сведения о графическом ускорителе.

Возвращаемый тип – WinAPI статус HRESULT.

9 RequestLibrary – функция запроса библиотеки пользователя с сервера. Для успешного выполнения данной функции необходимо инициализировать движок с указанием сервера и пройти аутентификацию.

Входные параметры:

- LibraryUnit\* &library – ссылка на указатель на структуру, описывающую элемент библиотеки, куда будет записана полученная библиотека. Выделять память для данного указателя не нужно.

- int &number – ссылка на переменную, куда будет помещено количество полученных элементов.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция создает массив с полученной информацией и возврщает указатель на него.

10 GenerateKey – функция генерации ключа шифрования для заданного приложения на основе системной конфигурации и полученной от сервера информации. Для успешного выполнения данной функции необходимо инициализировать движок с указанием сервера и пройти аутентификацию.

Входные параметры:

- const char\* sys\_info – системная кинфигурация;

- UINT64 lib\_code – уникальный код шифрования приложения из библиотеки пользователя;

- BYTE\* key – указатель на массив из 16 байт, в который будет загружен сгенерированный ключ.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция записывает в предоставленный массив сгенерированную информацию.

11 InstallSoftware – функция установки приложения. Функция устанавливает на клиентском компьютере заданное приложение по заданному пути. Для успешного выполнения данной функции необходимо инициализировать движок с указанием сервера и пройти аутентификацию.

- const char\* id – идентификатор приложения;

- const char\* path – путь установки.

Возвращаемый тип – WinAPI статус HRESULT.

12 EncryptData – функция шифрования. Выполняет шифрование данных по алгоритму AES-128 с использованием аппартного ускорения.

Входные параметры:

- BYTE\* data – данные, которые необходимо зашифровать. Результат шифрования будет записан по этому же указателю;

- INT32 size – размер массива данных. Для успешного выполнения функции необходимо, чтобы размер входного массива был кратен 16 байтам;

- BYTE\* key – 16-байтовый ключ шифрования.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция записывает в предоставленный массив сгенерированную информацию.

13 DecryptData – функция расшифрования. Выполняет расшифровывание данных по алгоритму AES-128 с использованием аппартного ускорения.

Входные параметры:

- BYTE\* data – данные, которые необходимо расшифровать. Результат будет записан по этому же указателю;

- INT32 size – размер массива данных. Для успешного выполнения функции необходимо, чтобы размер входного массива был кратен 16 байтам;

- BYTE\* key – 16-байтовый ключ шифрования.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция записывает в предоставленный массив сгенерированную информацию.

14 CalculateHash – функция вычисления хэша по алгоритму SHA256.

Входные параметры:

- BYTE \* data – данные, на основании которых необходимо вычислить хэш;

- UINT64 size – размер массива данных;

- BYTE\* state – указатель на 32-байтовый массив, в который будет записан результат.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция записывает в предоставленный массив сгенерированную информацию.

15 PackInContainer – функция записи информации в безопасный контейнер.

Входные параметры:

- BYTE\* data – данные, которые необходимо упаковать в контейнер;

- INT32 size – размер упакуемых данных в байтах;

- ContentType type – тип упакуемого контента;

- BYTE\* key – ключ шифрования для упаковки в контейнер;

- BYTE\* container\_name – имя файла контейнера.

Возвращаемый тип – WinAPI статус HRESULT.

16 ReadFromContainer – функция, позволяющая считать данные из безопасного контейнера.

Входные параметры:

- const char\* container\_name – путь к файлу контейнера;

- BYTE\* key – ключ расшифровывания;

- BYTE\* &data – ссылка на указатель на массив байт. Заранее выделять память для данного массива не нужно:

- INT32 &size – ссылка на переменную, в которую будет записано количество байт в результирующем массиве;

- ContentType &type – ссылка на переменную, в которую будет записан тип контента, распакованого из контейнера.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция создает массив с полученной информацией и возвращает указатель на него.

17 CalculateSoftHASH – функция расчета значения хэша по алгоритму SHA256 для заданной папки.

Входные параметры:

- const char\* path – путь до папки, для содержимого которого необходимо рассчитать значение хэша;

BYTE \* state – указатель на 32-байтовый массив, в который необходимо записать найденный хэш.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция записывает в предоставленный массив сгенерированную информацию.

18 CheckSoftHASH – функция сетевой проверки хэша для выбранного приложения. Для успешного выполнения данной функции необходимо инициализировать движок с указанием сервера и пройти аутентификацию.

Входные параметры:

- const char\* id – идентификатор приложения;

- const char\* path – путь местоположения приложения на клиентском устройстве.

Возвращаемый тип – WinAPI статус HRESULT.

Также библиотека DataWallEngine определяет следующие структуры и типы данных:

1 ContentType – 32-битное беззнаковое целое число, определяющие тип данных, загруженных или загружаемых в защищенный контейнер. Возможные значения:

CONTENT\_DLL - 0x4C4C44;

CONTENT\_IMAGE - 0x474D49;

CONTENT\_TEXT - 0x545854;

CONTENT\_SOUND - 0x444E53;

CONTENT\_BIN - 0x4E4942.

2 MotherboardInfo – структура, описывающая материнскую плату. Элементы структуры:

BSTR SerialNumber – серийный номер материнской платы;

BSTR Manufacturer – производитель материнской платы;

BSTR Product – модель материнской платы.

3 ProcessorInfo – структура, описывающая процессор. Элементы структуры:

BSTR Manufacturer – производитель процессора;

BSTR Name – модель процессора;

BSTR ProcessorId – идентификатор процессора;

BSTR SerialNumber – серийный номер процессора;

UINT32 NumberOfCores – количество ядер процессора.

4 VideoAdapterInfo – структура, описывающая графический ускоритель. Элементы структуры:

BSTR Caption – описание графического ускорителя;

BSTR Name – модель графического ускорителя;

BSTR VideoProcessor – модель графического процессора;

BSTR DeviceID – системный идентификатор графического ускорителя.

5 LibraryUnit – структура, описывающая ПО из библиотеки пользователя. Элементы структуры:

UINT64 id – идентификатор приложения;

std::string name – имя приложения;

UINT64 code – код шифрования приложения.

Также данная библиотека расширяет стандартные статусы WinAPI:

WRONG\_DATA - 0xFD000001 – Данные контейнера повреждены;

WRONG\_CONTENT - 0xFD000002 – Контейнер содержит неподдерживаемый контент.

5.2 Библиотека-загрузчик DataWallLoader

Данный модуль содержит в себе все функции системы, необходимые для загрузки защищенных приложений в память. Далее следует описание функций загрузчика:

1 EncryptData – функция шифрования. Выполняет шифрование данных по алгоритму AES-128 с использованием аппартного ускорения.

Входные параметры:

- BYTE\* data – данные, которые необходимо зашифровать. Результат шифрования будет записан по этому же указателю;

- INT32 size – размер массива данных. Для успешного выполнения функции необходимо, чтобы размер входного массива был кратен 16 байтам;

- BYTE\* key – 16-байтовый ключ шифрования.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция записывает в предоставленный массив сгенерированную информацию.

2 DecryptData – функция расшифрования. Выполняет расшифровывание данных по алгоритму AES-128 с использованием аппартного ускорения.

Входные параметры:

- BYTE\* data – данные, которые необходимо расшифровать. Результат будет записан по этому же указателю;

- INT32 size – размер массива данных. Для успешного выполнения функции необходимо, чтобы размер входного массива был кратен 16 байтам;

- BYTE\* key – 16-байтовый ключ шифрования.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция записывает в предоставленный массив сгенерированную информацию.

3 LoadEncryptedLibrary – функция загрузки в память упакованной в защищенный контейнер библиотеки динамической компановки.

Входные параметры:

- const char\* path – путь до безопасного контейнера, содержащего библиотеку;

- BYTE \* key – ключ расшифровывания для заданного контейнера.

Возвращаемый тип – дескриптор загруженной в память библиотеки HMEMORYMODULE.

4 FreeEncryptedLibrary – освобождает все ресурсы, занятые заданной библиотекой.

Входные параметры:

- HMEMORYMODULE handle - дескриптор загруженной в память библиотеки.

Функция не возвращает никаких данных.

5 InitDH – функция инициализации параметров безопасного канала, защищенного по алгоритму Диффи-Хэллмана. Данную функцию рекомендуется вызывать при старте приложения, генерирующего ключи, поскольку без инициализации защищенного канала функции передачи и загрузки ключа работать не будут. Выполнение данной функции в некоторых случаях может занимать длительное время.

Данная функция не принимает входных параметров.

Возвращаемый тип – WinAPI статус HRESULT.

6 SendKey – функция отправки ключа приложению. Для успешного выполнения данной функции необходимо инициализировать параметры безопасного канала. Данная функция создает защищенный канал и передает по нему ключ шифрования для заданного приложения.

Входные параметры:

- const char\* soft\_name – имя приложения, для которого требуется отправить ключ.

- BYTE\* key – ключ, который необходимо передать.

7 LoadKey – функция загрузки ключа приложением. Данная функция подключается с созданному с помощью функции Sendkey безопасному именованному каналу и выгружает ключ.

Входные параметры:

const char\* soft\_name – Имя приложения, для которого загружается ключ.

Выходной тип – указатель на массив байт ключа BYTE\*.

8 ReadFromContainer – функция, позволяющая считать данные из безопасного контейнера.

Входные параметры:

- const char\* container\_name – путь к файлу контейнера;

- BYTE\* key – ключ расшифровывания;

- BYTE\* &data – ссылка на указатель на массив байт. Заранее выделять память для данного массива не нужно:

- INT32 &size – ссылка на переменную, в которую будет записано количество байт в результирующем массиве;

- ContentType &type – ссылка на переменную, в которую будет записан тип контента, распакованого из контейнера.

Возвращаемый тип – WinAPI статус HRESULT.

Также функция создает массив с полученной информацией и возвращает указатель на него.

5.3 Системная служба Windows DataWallService

Системная служба запускается при старте системы и последовательно выполняет следующие действия:

1 Инициализирует именованный канал для обмена сообщениями и информацией с клиентским приложением.

2 Инициализирует параметры защищенного именованного канала для отправки ключа защищаемым приложениям.

3 Инициализирует движок.

4 Получает информацию о системной конфигурации.

5 Ожидает запросов от клиентского приложения и обрабатывает их.

Служба является неостанавливаемой, на запрос остановки служба реагировать не будет.

5.4 Клиентское приложение DataWallClient

Данное приложение является графическим интерфейсом для взаимодействия пользователя с системой. При старте приложение подключается к именованному каналу, предварительно созданному системной службой. В случае отсутствия именованного канала в системе будет выводится соответствующее сообщение (рисунок 5.1).

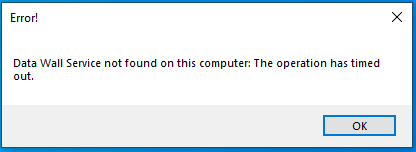


Рисунок 5.4.1 – Сообщение об ошибке запуска

При успешном старте клиентского приложения будет открыто окно с полями аутентификации (рисунок 5.2).

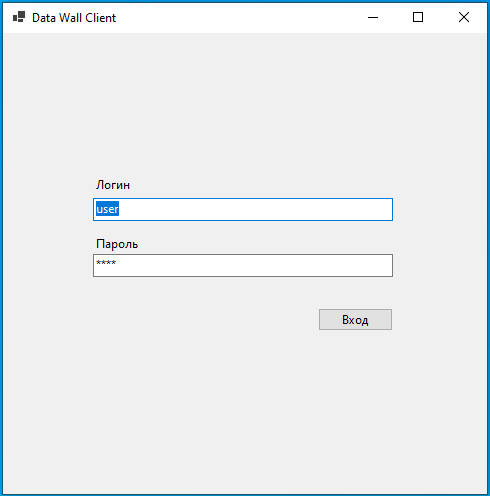


Рисунок 5.4.2 – Окно в режиме аутентификации

При неудачной попытке аутентификации пользователю будет выведено соответствующее сообщение (рисунок 5.3).

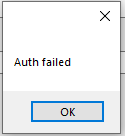


Рисунок 5.4.3 – Аутентификация провалена

При удачной аутентификации пользователь будет авторизован, будет выведено соответствующее сообщение (рисунок 5.4) и далее в окне будет выведен список доступного ПО (рисунок 5.5) с возможностью выполнения следующих действий:

1 Загрузка;

2 Удаление;

3 Переупаковка;

4 Проверка файлов;

5 Запуск.

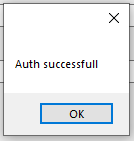


Рисунок 5.4.4 – Аутентификация завершилась успешно

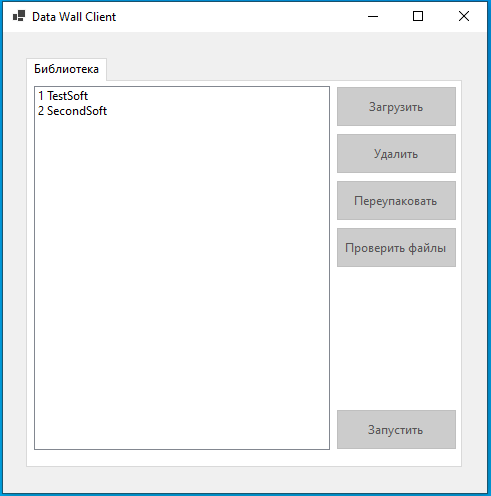


Рисунок 5.4.5 – Основное окно клиентского приложения

5.5 Серверное приложение DataWallServer

Серверное приложение запускается на серверной машине провайдера услуг. При старте данное приложение инициализирует сокет для безопасного транспортного соединения TLS 2.1 и подключается к установленной на машине БД DataWall. Серверное приложение представляет собой оконное приложение, содержащее в себе следующие инструменты:

1 Инструменты для мониторинга подключенных пользователей и вывода информации о них, а именно логина, хэша пароля, уникального кода шифрования для данного пользователя (рисунок 5.5.1).

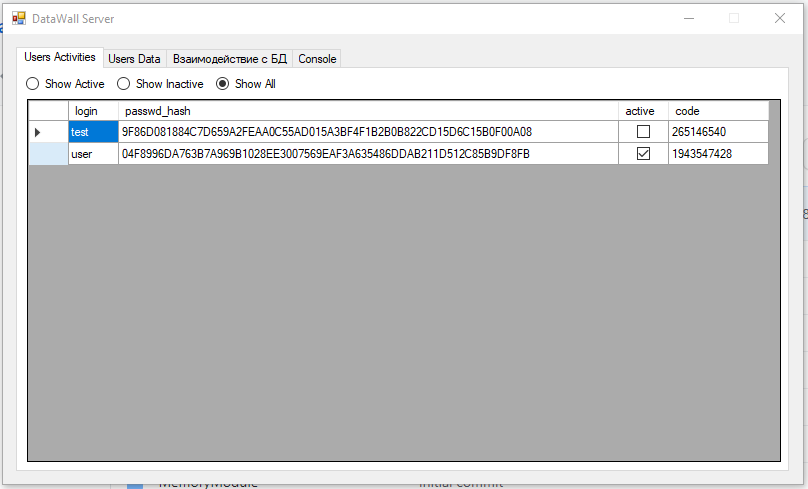


Рисунок 5.5.1 – Информация о пользователях

2 Инструменты для просмотра данных отдельных пользователей, таких как информация о пользовательской библиотеке и устройствах (рисунок 5.5.2).

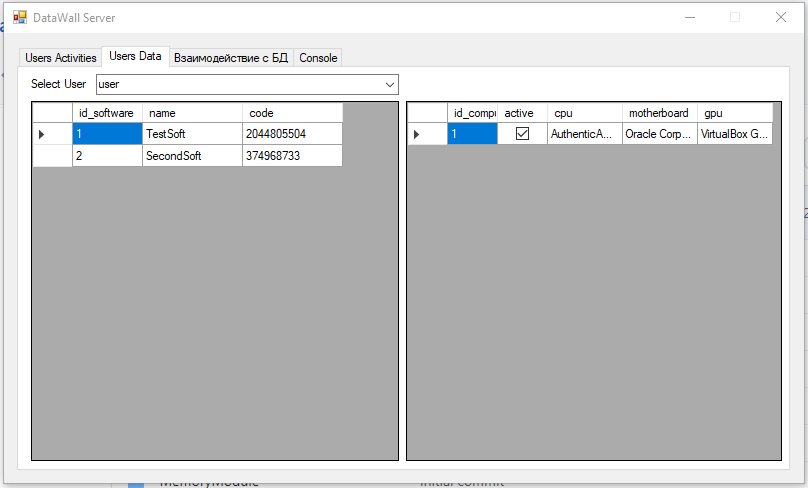


Рисунок 5.5.2 – Пользовательские данные

3 Инструменты для регистрации новых пользователей, продуктов и выдачи ПО пользователя (рисунок 5.5.3).

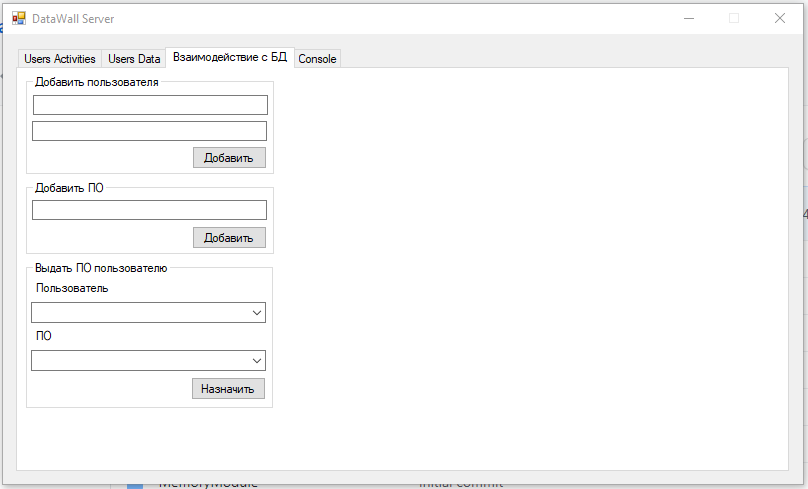


Рисунок 5.5.3 – Инструменты регистрации

4 Консоль с выводом событий, произошедших на сервере, для удобного администрирования (рисунок 5.5.4).

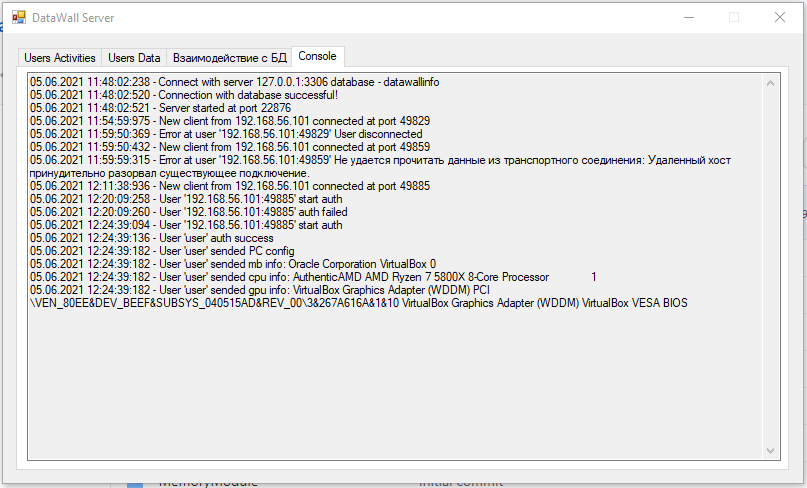


Рисунок 5.5.4 – Консоль

# 6 Описание алгоритмов работы

6.1 Описание протокола сетевого взаимодействия сервера и службы

В качестве транспортного протокола используется протокол TLS 2.1 в реализации OpenSSL.

Общая структура сообщений одинакова вне зависимости от требуемых действий и представляет собой код, описывающий требуемое действие или результат выполнения запроса – 1 байт и строку в формате UTF-8, содержащую данные, необходимые для выполнения запроса или результат его выполнения.

Коды сообщений:

200 – статус успешного завершения операции;

255 – операция завершилась с ошибкой или запрос отклонен;

230 – запрос на аутентификацию;

250 – передача системной конфигурации;

100 – запрос на получение библиотеки пользователя;

110 – запрос на загрузку ПО из библиотеки;

130 – запрос на проверку целостности ПО.

Описание клиент-серверных взаимодействий (описываются только форматы отправляемых строк):

1 Аутентификация – клиент отправляет хэш пароля и имя пользователя, сервер проводит аутентификацию и высылает результат: в случае успешной аутентификации сервер высылает код шифрования, соответствующий авторизованному пользователю, в случае отказа – сообщение, описывающее причину отказа.

2 Передача системной конфигурации – клиент отправляет разделенные нулями строки, описывающие материнскую плату, процессор и графический ускоритель. Ответ от сервера не требуется.

3 Запрос библиотеки – клиент не отправляет сообщения помимо кода запроса. Сервер обслуживает запрос и отправляет ответ с соответствующим кодом. В случае безуспешного выполнения запроса, сервер отправляет сообщение, описывающее причину провала. В случае успешного выполнения запроса сервер отправляет библиотеку в следующем формате: идентификатор приложения (8 байт), код шифрования для данного приложения (8 байт), имя приложения, нуль – для каждого приложения в библиотеке.

4 Запрос на установку приложения – клиент отправляет идентификатор приложения, которое желает получить. Сервер обслуживает запрос и отправляет ответ с соответствующим кодом. В случае безуспешного выполнения запроса, сервер отправляет сообщение, описывающее причину провала. В случае успешного выполнения запроса сервер отправляет описание файловой структуры приложения в следующем формате: флаг, указывающий, является ли файл упакованным в безопасный контейнер (1 байт), тип данных упаковынных в контейнер (4 байта), имя файла, нуль – для каждого файла в приложении. Далее сервер отправляет непосредственно данные файлов. Упакованные файлы отправляются уже в защищенном виде. Получив и установив все файлы, клиент вычисляет хэш полученного приложения и высылает его на сервер.

5 Проверка хэша приложения – клиент отправляет идентификатор приложения, для которого необходимо проверить контрольную сумму и значение вычисленного на клиенте хэша. Сервер обслуживает запрос и отправляет ответ с соответствующим кодом. В случае безуспешного выполнения запроса, сервер отправляет сообщение, описывающее причину провала. В случае успешного выполнения запроса сервер отправляет сообщение об успехе.

6.2 Описание протокола взаимодействия клиентского приложения и системной службы

В качестве транспорта используется именованный канал Windows.

Общая структура сообщений одинакова вне зависимости от требуемых действий и представляет собой код, описывающий требуемое действие или результат выполнения запроса – 1 байт и опционально строку в формате UTF-8, содержащую данные, необходимые для выполнения запроса или результат его выполнения.

Коды сообщений:

200 – статус успешного завершения операции;

255 – операция завершилась с ошибкой или запрос отклонен;

230 – запрос на аутентификацию;

100 – запрос на получение библиотеки пользователя;

110 – запрос на загрузку ПО из библиотеки;

130 – запрос на проверку целостности ПО;

150 – запустить ПО.

Описание взаимодействий (описываются только форматы отправляемых строк):

1 Аутентификация. Клиентское приложение отправляет имя пользователя и пароль, введенные пользователем. Служба обслуживает запрос и отправляет ответ с соответствующим кодом.

2 Получение библиотеки. Клиентское приложение отправляет код запроса. Служба обслуживает запрос и отправляет ответ с соответствующим кодом. В случае успешного выполнения запроса служба отправляет данные приложений в формате: идентификатор приложения и имя приложения.

3 Загрузка приложения. Клиентское приложение отправляет идентификатор приложения и путь установки. Служба обслуживает запрос и отправляет ответ с соответствующим кодом.

4 Проверка целостности ПО. Клиентское приложение отправляет идентификатор приложения и путь установки. Служба обслуживает запрос и отправляет ответ с соответствующим кодом.

5 Запуск ПО. Клиентское приложение отправляет идентификатор приложения. Служба обслуживает запрос и отправляет ответ с соответствующим кодом.

6.3 Описание протокола взаимодействия защищаемого приложения и системной службы

В качестве транспорта используется именованный канал Windows. В качестве алгоритма шифрования используется алгоритм одноразового блокнота, поскольку объем передаваемых данных ограничен размером ключа AES-128. В качестве протокола генерации ключей с их обменом используется алгоритм Диффи-Хэллмана в реализации OpenSSL.

В качестве инициализации обмена сообщениями используется алгоритм рукопожатия:

1 Служба заранее генерирует параметры Диффи-Хэллмана;

2 При подключении защищаемого приложения, служба отправляет сгенерированные параметры и свою открытую часть ключа;

3 На основе полученных параметров Диффи-Хэллмана защищаемое приложение генерирует свои ключевые данные, отправляет службе свою открытую часть ключа и вычисляет общий секрет;

4 Получив открытую часть ключа от защищаемого приложения, служба также вычисляет общий секрет.

Передача ключа шифрования AES-128 производится следующим алгоритмом:

1 Служба шифрует ключ сгенерированным секретом и отправляет его защищаемому приложению;

2 Приложение получает ключ и расшифровывает его сгенерированным секретом;

3 Приложение отправляет код 200 для подтверждения получения;

4 Служба получает код и проверяет его.

6.4 Описание формата защищенного контейнера

Безопасный контейнер является полностью зашифрованным, поэтому его размер всегда кратен размеру блока шифрования AES-128. Первый 128-битный блок является заголовочным и содержит в себе зашифрованную информацию о контейнере:

1 Код аутентичности контейнера – 4 байта, всегда равен 0x4C575444 (DTWL в кодировке UTF-8);

2 Тип данных, упакованных в контейнер – 4 байта, возможные значения:

- 0x4C4C44 (DLL в кодировке UTF-8) – библиотека динамической компоновки;

- 0x474D49 (IMG в кодировке UTF-8) - изображение;

- 0x545854 (TXT в кодировке UTF-8) - текст;

- 0x444E53 (SND в кодировке UTF-8) - звук;

- 0x4E4942 (BIN в кодировке UTF-8) – двоичные данные.

3 Размер контейнера в байтах, включая заголовочный блок – 4 байта. Размер контейнера всегда кратен 16 байтам (128-битам).

4 Размер зашифрованных данных в байтах – 4 байта. Указывает размер зашифрованных данных без учета заголовочного блока.

5 Непосредственно данные.

6.5 Способ хранения данных об установленных приложениях

Данные об установленных в системе приложениях хранятся в системном реестре в ключе HKEY/SOFTWARE/DataWall. Для каждого пользователя создается собственная подключ, в котором создается подключ для каждого из приложений, имеющихся в библиотеке пользователя. Каждый такой подключ содержит поля Install и InstallPath, содержащие сведения о том установлено ли приложение в системе и путь установки соответственно.

6.6 Алгоритм запуска приложения

Приложение запускается в несколько этапов:

1 Клиентским приложением из реестра считывается путь установки для выбранного приложения;

2 Проверяется целостность клиентского приложения (см. пункт 6.2), в случае несоответствия хэша запуск отменяется;

3 Отправляется запрос на подготовку службы к обслуживанию запускаемого приложения (см. пункт 6.2);

4 Клиентское приложение запускает защищаемое ПО.

# 7 Тестирование

Для тестирования функционала было разработано приложение, защищенное с помощью разработанной системы. Данное приложение весь функционал, а также все данные хранит в защищенных контейнерах.

Приложение работает по следующему алгоритму:

1 Подключается к сервису и загружает ключ шифрования;

2 Загружает библиотеку, содержащую в себе функционал ПО из безопасного контейнера;

3 Из загруженной библиотеки выгружаются необходимые функции;

4 Загружает защищенные изображение и текст из безопасных контейнеров;

5 Выполняет код программы;

6 Освобождает занятые ресурсы и завершается.

Была протестирована возможность загрузки защищаемого приложения (рисунок 7.1).

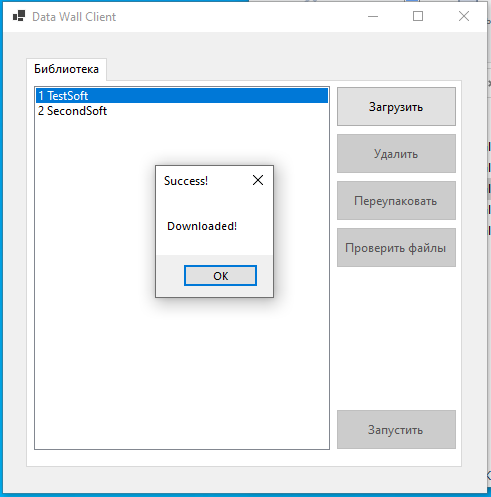


Рисунок 7.1 – Приложение загружено

Была протестирована возможность запуска защищенного приложения (рисунок 7.2).

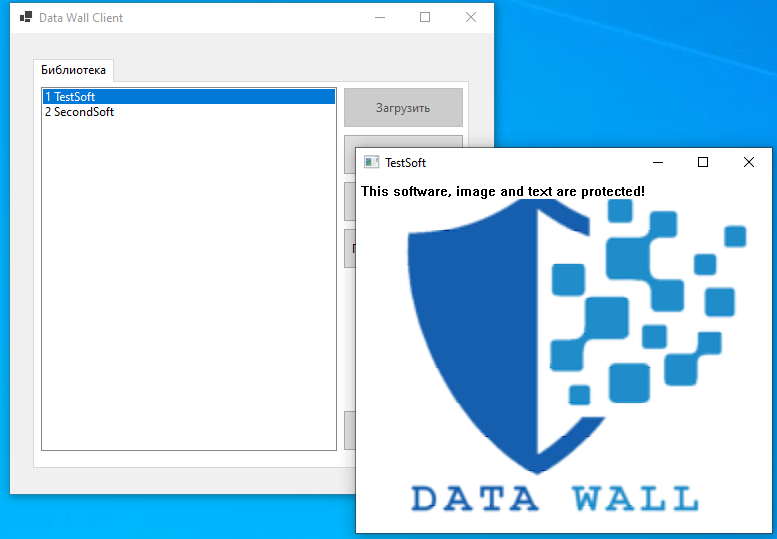


Рисунок 7.2 – Приложение успешно запущено

Была протестирована невозможность запуска приложения с нарушенной целлостностью (рисунок 7.3).

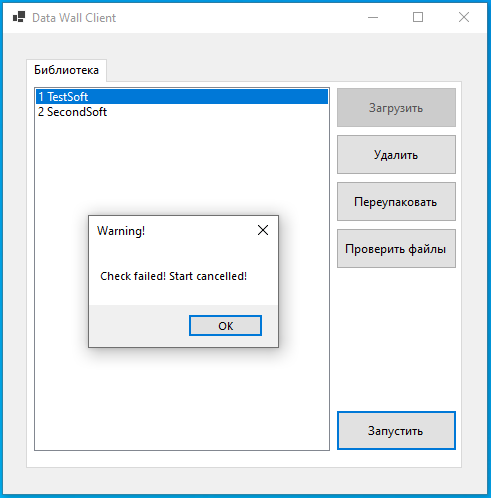


Рисунок 7.3 – Проверка целостности провалена, запуск отменен

Была протестирована невозможность запуска приложения при внезапном отключении сервера (рисунок 7.4). Поскольку сервер отключен, не удалось проверить хэш и запуск приложения был отменен.

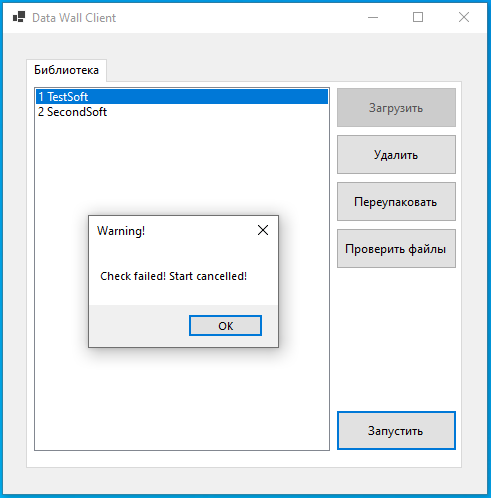


Рисунок 7.4 – Запуск не удался

Была протестирована невозможность запуска приложения непосредственно из проводника (рисунок 7.5). Поскольку не удалось подключится к именованному каналу и не был получен ключ шифрования, не удалось загрузить исполняемый код и программа завершилась с ошибкой.

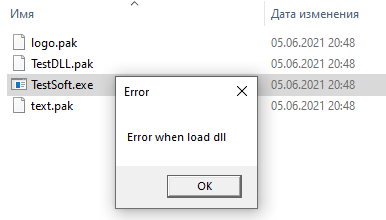


Рисунок 7.5 – Не удалось загрузить исполняемый код

Была проверена возможно проверки целостности файлов (рисунок 7.6).

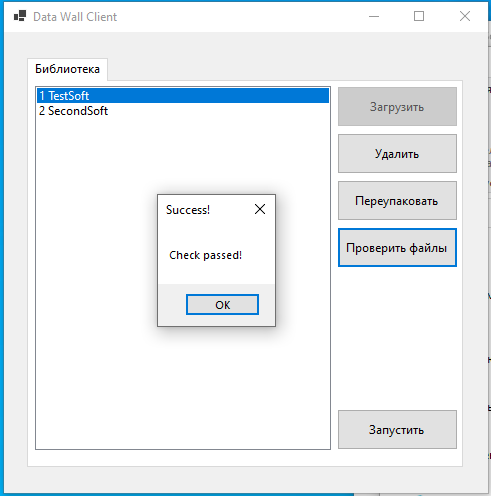


Рисунок 7.6 – Приложение неизменно (результат верен)

Была проверена возможность переупаковки приложения (рисунок 7.7).

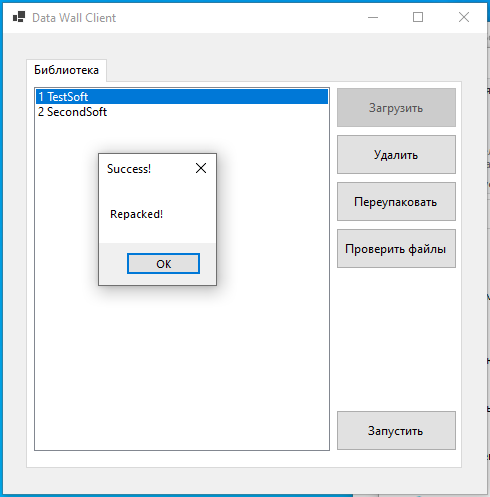


Рисунок 7.7 – Приложение переупаковано

Была проверена возможность удаления приложения (рисунок 7.8).

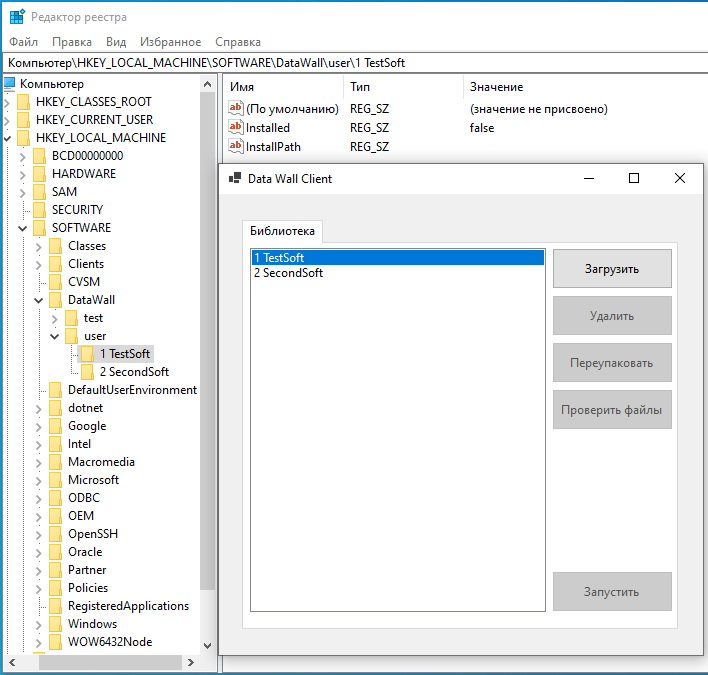


Рисунок 7.8 – Приложение удалено

# Заключение

В ходе выполнения курсовой работы была разработана система DRM-защиты согласно техническому заданию. Были освоены следующие навыки:

1 Реализация низкоуровневой загрузки исполняемого кода в память из безопасных контейнеров;

2 Использование процессорных аппаратных средств шифрования;

3 Реализация безопасного клиент-серверного приложения;

4 Использование средств автоматической сборки;

5 Разработка системных служб;

6 Управление системными службами;

7 Реализация межпроцессного взаимодействия.

# Список литературы

1 Weinberg J. Hardware-Based ID, Rights Management, and Trusted Systems // Stan. L. Rev. 1999. Vol. 52. P. 1251.

2 Alrashid T.M. et al. Safeguarding Copyrighted Contents: Digital Libraries and Intellectual Property Management. CWRU’s Rights Management System // D-Lib Magazine. 1998.

3 Iannella R. Digital Rights Management (DRM) Architectures // D-Lib Magazine. 2001. Vol. 7.

4 Subramanya S.R., Yi B.K. Digital rights management // IEEE Potentials. 2006. Vol. 25, № 2. P. 31–34.

5 Руководство по программированию на C#. [Электронный ресурс]. – Режим доступа https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/ (дата обращения 01.03.2021)

6 Руководство по MySQL. [Электронный ресурс]. – Режим доступа https://metanit.com/sql/mysql/ (дата обращения 14.04.2021)

7 ОС ТУСУР. [Электронный ресурс]. – Режим доступа https://storage.tusur.ru/files/40668/rules\_tech\_01-2013.pdf (дата обращения 26.05.2021)

8 Common HRESULT Values [Электронный ресурс]. – https://docs.microsoft.com/en-us/windows/win32/seccrypto/common-hresult-values (дата обращения 05.06.2021)

Приложение А

(обязательное)

Руководство администратора

1 Условия применения программы

1.1. Требования к техническим средствам

Требования к техническим средствам системы DRM-защиты DataWall. Для работы серверной части системы необходима следующая минимальная конфигурация:

- процессор x86-64 с поддержкой аппаратного шифрования AES-128;

- оперативная память – не менее 1 Гб;

- свободного места на накопителе не менее 100 Мб;

- наличие интернет соединения.

1.2. Требования к общему программному обеспечению (ОПО)

Требования к общему программному обеспечению (ОПО), необходимому для серверной части системы DRM-защиты DataWall:

- операционная система Windows 7 с пакетом обновления 1 или выше или Windows Server 2008 R2 с пакетом обновления 1 или выше;

- СУБД MySQL;

- MySQL Server;

- Microsoft .NET Framework 4.7.2 или выше;

- OpenSSL.

2 Характеристики программы

Система рассчитана на круглосуточный режим работы в течение 365 дней в году. Средствами контроля и мониторинга являются автоматизированные встроенные в платформу программные компоненты Серверное приложение представляет собой оконное приложение, содержащее в себе следующие инструменты:

1 Инструменты для мониторинга подключенных пользователей и вывода информации о них, а именно логина, хэша пароля, уникального кода шифрования для данного пользователя.

2 Инструменты для просмотра данных отдельных пользователей, таких как информация о пользовательской библиотеке и устройствах.

3 Инструменты для регистрации новых пользователей, продуктов и выдачи ПО пользователя.

4 Консоль с выводом событий, произошедших на сервере, для удобного администрирования.

3 Обращение к программе

Серверное приложение содержит в себе себе следующие вкладки: «Users Activities», «Users Data», «Взаимодействие с БД» и «Console».

На вкладке «Users Activities» располагается информация о пользователях системы с указанием активен ли пользователь в данный момент. С помощью флагов «Show Active». «Show Inactine», «Show All» можно фильтровать список пользователей, отображая только активных в данный момент, только отстутствующих или всех пользователей соответственно (рисунок 3.1).

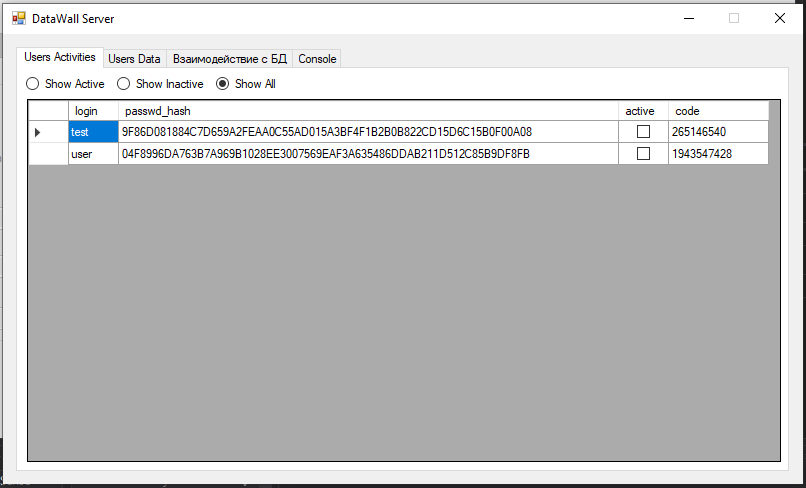


Рисунок 3.1 – Вкладка «Users Activities»

На вкладке «Users Data» располагается информация о принадлежащих пользователям программах и их устройствах. Для просмотра информации о пользователе его необходимо выбрать из выпадающего списка «Select User».

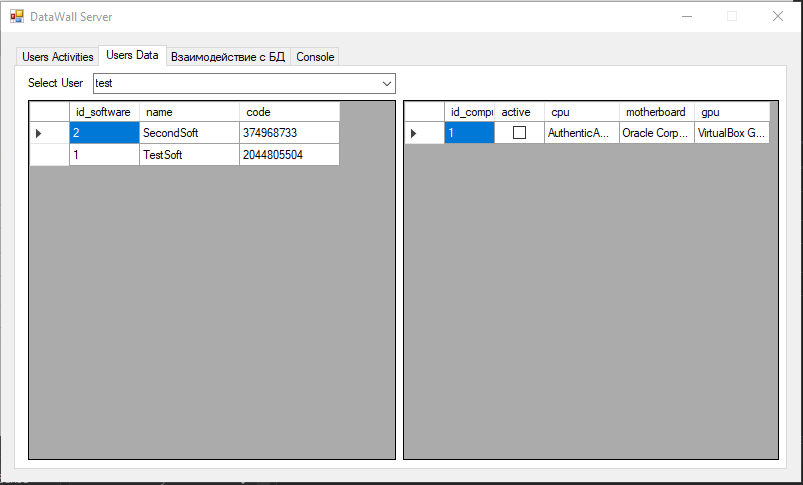


Рисунок 3.2 – Вкладка «Users Data»

Вкладка «Взаимодействие с БД предназначена» для регистрации новых пользователей, продуктов и выдачи ПО пользователя (рисунок 3.3).

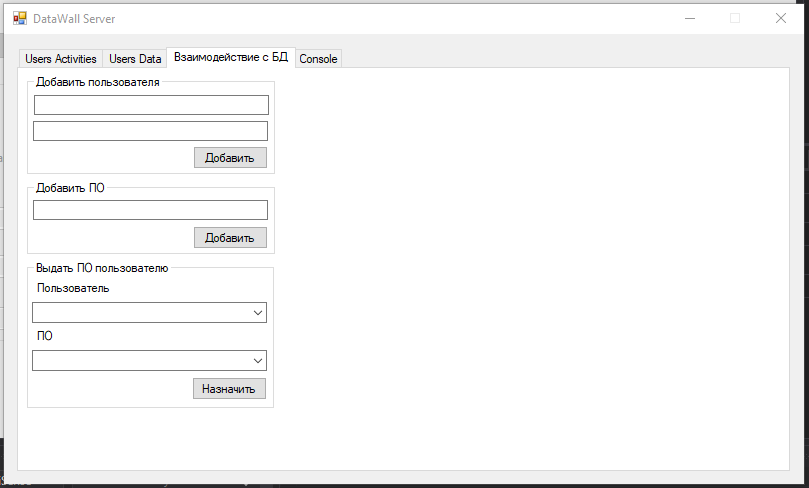


Рисунок 3.3 – Вкладка «Взаимодействие с БД»

Чтобы зарегистрировать в системе нового пользователя в группе «Добавить пользователя» в первое сверху текстовое поле введите имя нового пользователя, а во второе – пароль и нажмите кнопку «Добавить».

Чтобы добавить ПО в группе «Добавить ПО» в текстовое полее введите имя нового продукта и нажмите кнопку «Добавить».

Чтобы выдать ПО пользователю в группе «Выдать ПО пользователю» в первом сверху выпадающем списке выберите пользователя, а во втором – ПО, которое хотите ему назначить и нажмите кнопку «Назначить».

На вкладке «Console» выводятся все события, предупреждения и ошибки, происходящие в системе (рисунок 3.4).

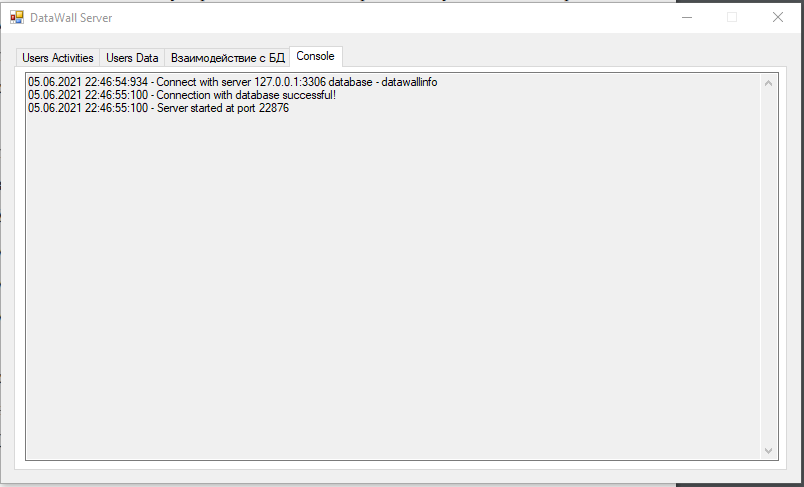


Рисунок 3.4 – Вкладка «Console»

Сообщение в консоли имеет следующую структуру: дата в формате дд.мм.гггг, время в формате чч:мм:сс:ммм, тире и сообщение.

4 Сообщения

Во время работы приложения или взаимодействия с ним администратор может получать сообщения.

При добавлении пользователя, ПО или назначении ПО пользователю возможно появление окна-сообщения «Wrong value(s)» - это сообщение означает, что введенные данные не могут быть приняты. Измените входные данные, чтобы исправить эту ошибку.

Остальные сообщения выводятся в лог на вкладке «Console».

При старте сервера в лог выводятся следующие сообщения:

«Connect with server <адрес сервера>:<порт> database - <имя базы данных>» - означает, что сервер инициировал подключение к БД;

«Connection with database successful!» - сервер успешно подключился к БД;

«Server started at port <порт>» - сервер запущен на всех доступных сетевых интерфейсах на заданном порту.

Сообщения взаимодействий с клиентом:

«Error, when accept new client» - Произошла ошибка сервера при подключении нового клиента.

«New client from <адрес клиента> connected at port <порт>» - Подключен новый клиент. До момента авторизации клиента в системе комбинация адрес пользователя и порт будет использоваться в качестве его идентификатора, а после – идентификатором пользователя будет его логин.

«Failed to add new client with error: <описание ошибки>» - Не удалось добавить нового клиента.

«User '<идентификатор>' start auth» - Пользователь начал процедуру аутентификации.

«User '<идентификатор>' auth success» - Пользователь успешно автоизован.

«User '<идентификатор>' auth failed» - Пользователь не автоизован.

«User '<идентификатор>' sended PC config» - Пользователь начал отправку конфигурации устройства.

«User '<идентификатор>' sended mb info: <конфигурация материнской платы>» - Пользователь прислал конфигурацию материнской платы.

«User '<идентификатор>' sended cpu info: <конфигурация ЦП>» - Пользователь прислал конфигурацию ЦП.

«User '<идентификатор>' sended gpu info: <конфигурация графического ускорителя>» - Пользователь прислал конфигурацию графического ускорителя".

«User '<идентификатор>' want software <имя приложения>» - Пользователь инициировал установку приложения.

«User '<идентификатор>' Soft '<имя приложения>' hash: <значение хэша>» - Пользователь прислал рассчитанный для приложения хэш.

«User '<идентификатор>' check hash passed» - Пользователь успешно прошел проверку хэша приложения.

«User '<идентификатор>' check hash failed» - Пользователь провалил проверку хэша приложения.

«Error when remove dead client: <описание ошибки>» - Не удалось удалить структуры, связанные с отключившимся пользователем.

«Error at user '<идентификатор>' <описание ошибки>» - Произошла сетевая ошибка или пользователь был отключен.

Сообщения взаимодействий с базой данных:

«Connection error - <описание ошибки>» - Не удалось подключится к БД.

«Database error - <описание ошибки>» - Не удалось выполнить действие к БД.

«Database error - <описание ошибки> for query : <запрос>» - Не удалось выполнить запрос к БД.

Приложение Б

(обязательное)

Руководство пользователя

1 Введение

1.1 Область применения

Требования настоящего документа применяются при эксплуатации.

1.2 Краткое описание возможностей

Система DRM-защиты DataWall предназначена в первую очередь для защиты программного обеспечения от неконтролируемого распространения. Для удобства использования конечным потребителем предоставляется клиентское приложение, предоставляющее возможности работы с приложениями.

1.3 Уровень подготовки пользователя

Пользователь должен иметь опыт работы с:

- операционными системами семейства Windows;

- оконными приложениями Windows.

1.4. Перечень эксплуатационной документации

Руководство пользователя.

2 Назначение и условия применения

2.1 Виды деятельности, функции

Клиентское приложение позволяет устанавливать и запускать защищенные приложения, удалять их, проверять целостность установленных файлов, а также легко и быстро переустанавливать приложения в случае повреждения установочной директории или изменения аппаратной конфигурации ПК.

Система использует возможности аппаратного шифрования процессором, что позволяет не влиять на производительность защищаемого приложения. В качестве секрета, используемого для защиты данных, используется помимо прочего конфигурация клиентского компьютера. Поэтому, при смене аппаратной конфигурации защищаемое ПО необходимо переупаковывать, для чего выделена соответствующая кнопка клиентского приложения.

2.2 Программные и аппаратные требования к системе

Со стороны пользователя требования к аппаратуре следующие:

- процессор на архитектуре x86-64;

- наличие в процессоре аппаратных блоков шифрования AES;

- наличие интернет соединения;

- требования со стороны защищаемого ПО.

Требования к общему программному обеспечению (ОПО), необходимому для работы DRM-защиты:

- операционная система Windows 7 с пакетом обновления 1 или выше;

- OpenSSL;

- Распространяемый компонент Microsoft Visual C++ для Visual Studio 2015, 2017 и 2019;

- .NET Core 3.1 или выше.

3 Работа с системой

3.1 Запуск системы

1 Для того, чтобы запустить клиент DataWall запустите соответствующий ему исполняемый файл;

2 Авторизуйтесь в системе;

3 Выберите интересующую продукт в вашей библиотеке и запустите его.

3.2 Проверка работоспособности системы

Для проверки работоспособности клиентского приложения и доступности сервиса DataWall в целом:

1 Запустите приложение;

2 Авторизироваться в системе;

3 Получить доступ к меню и библиотеке;

4 В библиотеке должны отображаться ваши приложения;

5 Запустить приложение.

Если какой-либо пункт не удается выполнить, обратитесь в раздел 5 “Аварийные ситуации” настоящего документа.

4 Описание операций

4.1 Наименование операции

Клиентское приложение DataWall в составе системы DRM-защиты выполняет функции, описанные в таблице 4.1.

Таблица 4.1 - Функции клиентского приложения DataWall

|  |  |
| --- | --- |
| Функции | Задачи |
| Взаимодействие клиента с сервером | Авторизация клиента на сервере |
| Получение от сервера секрета для генерации ключа шифрования |
| Загрузка приложений с сервера |
| Проверка целостности приложения |
| Работа с установленными приложениями | Запуск приложений |
| Удаление приложений |

4.2 Условия выполнения операции

Задача: «Авторизоваться в системе»

Условия, при соблюдении которых возможно выполнение операции:

- Имеется подключение к сети “Интернет”.

Подготовительные действия:

- Запустить клиентское приложение.

Основные действия в требуемой последовательности:

- Ввести логин и пароль

- Нажать кнопку «Вход»

Заключительные действия: Не требуются.

Задача: «Загрузить приложение»

Условия, при соблюдении которых возможно выполнение операции:

- Имеется подключение к сети “Интернет”;

- Пользователь авторизован.

Подготовительные действия:

- Запустить клиентское приложение;

- Авторизоваться.

Основные действия в требуемой последовательности:

- Выбрать приложение из библиотеки;

- Нажать кнопку «Загрузить».

Заключительные действия: Не требуются.

Задача: «Удалить приложение»

Условия, при соблюдении которых возможно выполнение операции:

- Имеется подключение к сети “Интернет”;

- Пользователь авторизован;

- Приложение установлено.

Подготовительные действия:

- Запустить клиентское приложение;

- Авторизоваться.

Основные действия в требуемой последовательности:

- Выбрать приложение из библиотеки;

- Нажать кнопку «Удалить».

Заключительные действия: Не требуются.

Задача: «Переупаковать приложение»

Условия, при соблюдении которых возможно выполнение операции:

- Имеется подключение к сети “Интернет”;

- Пользователь авторизован;

- Приложение установлено.

Подготовительные действия:

- Запустить клиентское приложение;

- Авторизоваться.

Основные действия в требуемой последовательности:

- Выбрать приложение из библиотеки;

- Нажать кнопку «Переупаковать».

Заключительные действия: Не требуются.

Задача: «Проверить целостность приложения»

Условия, при соблюдении которых возможно выполнение операции:

- Имеется подключение к сети “Интернет”;

- Пользователь авторизован;

- Приложение установлено.

Подготовительные действия:

- Запустить клиентское приложение;

- Авторизоваться.

Основные действия в требуемой последовательности:

- Выбрать приложение из библиотеки;

- Нажать кнопку «Проверить файлы».

Заключительные действия: Не требуются.

Задача: «Запустить приложение»

Условия, при соблюдении которых возможно выполнение операции:

- Имеется подключение к сети “Интернет”;

- Пользователь авторизован;

- Приложение установлено.

Подготовительные действия:

- Запустить клиентское приложение;

- Авторизоваться.

Основные действия в требуемой последовательности:

- Выбрать приложение из библиотеки;

- Нажать кнопку «Запустить».

Заключительные действия: Не требуются.

5 Аварийные ситуации

Аварийные ситуации и способы их разрешения представлены в таблице 5.1.

Таблица 5.1 – Ошибки

|  |  |  |  |
| --- | --- | --- | --- |
| Класс ошибки | Ошибка | Описание ошибки | Требуемые действия пользователя при возникновении ошибки |
| Ошибки программно-аппаратного окружения | Процессор не поддерживает требуемых функций | Установленный в системе процессор не поддерживает инструкции шифрования AES | Проверить наличие аппаратной поддержки заявленных функций  Использовать более современное устройство |
| Библиотека \*\*\* не найдена в системе | Приложение не смогло запуститься по причине отсутствия в системе необходимого исполняемого кода | Переустановить приложение  Попробовать вручную установить требуемый исполняемый код  Переустановить ОС |
| Ошибки авторизации | Auth failed | Пользователем был введен неверные логин и\или пароль при входе в приложение | Проверьте интернет соединение  Введите корректные данные |
| Ошибки запуска приложения | Check failed! Start cancelled! | Приложение повреждено или отсутствует подключение к серверу | Проверьте интернет соединение  Переупакуйте приложение |

6 Рекомендации по освоению

Для успешного освоения клиентского приложения DataWall необходимо иметь навыки работы с вычислительными устройствами и изучить следующее:

- Настоящее «Руководство пользователя».

Приложение В

(обязательное)

Руководство программиста

1 Условия применения программы

1.1. Требования к техническим средствам

Требования к техническим средствам системы DRM-защиты DataWall. Для разработки приложений, использующих функционал системы необходима следующая минимальная конфигурация:

- процессор x86-64 с поддержкой аппаратного шифрования AES-128;

- оперативная память – не менее 1 Гб;

- свободного места на накопителе не менее 100 Мб.

1.2. Требования к общему программному обеспечению (ОПО)

Требования к общему программному обеспечению (ОПО), необходимому для разработки приложений, использующих функционал системы DRM-защиты DataWall:

- операционная система Windows 7 с пакетом обновления 1 или выше;

- OpenSSL;

- Распространяемый компонент Microsoft Visual C++ для Visual Studio 2015, 2017 и 2019.

2 Характеристики SDK

Основными пользователями системы DRM-защиты DataWall являются не конечные потребители, а разработчики, желающие защитить свое ПО от недобросовестного использования. С целью удовлетворить их потребности система предоставляет SDK-разработчика. Данное SDK позволяет разрабатывать защищенную архитектуру приложения, при использовании которой возможно хранить в защищенном виде весь исполняемый код и ресурсы защищаемого ПО. SDK включает в себя две библиотеки динамической компоновки: DataWallEngine ил DataWallLoader, заголовочные файлы к ним и данное руководство.

По умолчанию все необходимые функции для реализации защищенного приложения содержатся в библиотеке DataWallLoader, а использование библиотеки DataWallEngine необязательна и потребуется только в том случае если при разработке возникла необходимость в более тонкой настроке ПО или дополнительном тестировании.

2 Обращение к SDK

API программиста DataWallEngine: см. пункт 5.1.

API программиста DataWallLoader: см. пункт 5.2.

3 Сообщения

Сообщения движка DataWallEngine записываются в лог-файл, создаваемый при инициализации движка и/или в консольный вывод по запросу программиста. Формат сообщений: дата в формате дд.мм.гггг, время в формате чч:мм:сс, имя модуля и функция, в которых было сгенерировано сообщения и сообщение. Возможные сообщения:

«Logging initialized» – инициализирован новый лог;

«Initialize COM» - инициализация системных COM-объектов

«Set general COM security levels» - запущена установка уровня защиты COM-взаимодействий;

«Obtain the initial locator to WMI» - запущено получение стартового указателя на службу WMI для получения системной конфигурации;

«Connect to WMI through the IWbemLocator::ConnectServer method» - запущено подключение к службе WMI;

«Set security levels on the proxy» - Настройка взаимодействия со службой WMI;

«Initialize WinSockets» - Запущена инициализация WSA;

«Initialize socket» - Создается новый сокет;

«Connect with server» - Попытка подключится к серверу;

«Init SSL» - Запущена инициализация безопасного сетевого транспорта;

«Connect over SSL» - Выполняется TLS-рукопожание;

«Clear SSL» - Запущена разинициализация безопасного сетевого транспорта;

«Close socket» - Закрывается сокет;

«Query for motherboards quantity» - Выполняется запрос к службе WMI для получения информации о материнских платах;

«Motherboards count» - Определяется количество материнский плат в системе;

«Motherboards quantity = <количество>» - Получено количество материнский плат в системе;

«Query for CPUs quantity» - Выполняется запрос к службе WMI для получения информации о процессорах;

«CPUs count» - Определяется количество процессоров в системе;

«CPUs quantity = <количество>» - Получено количество процессоров в системе;

«Query for GPUs quantity» - Выполняется запрос к службе WMI для получения информации о графических ускорителях;

«GPUs count» - Определяется количество графических ускорителей в системе;

«GPUs quantity = <количество>» - Получено количество графических ускорителей в системе;

«Query for data» - Выполняется запрос к WMI на получение данных о конкретном устройстве;

«Read INFO» - Считываются данные, полученные от WMI;

«Start network authentication» - Запущена аутентификация на сервере;

«Failed to send authentication info» - Не удалось отправит аутентификационную информацию;

«Failed to recv authentication result» - Не удалось получить результат аутентификации;

«Authentication successfull with message: <код шифрования для пользователя>» - пользователь авторизован и получен код шифрования для данного пользователя;

«Authentication failed with code <код ошибки> and message: <сообщение от сервера>» - Аутентификация провалена с кодом ошибки и получено сообщение, описывающее ошибку;

«Start send device config» - Запущена отправка конфигурации устройства пользователя на сервер;

«Failed to send device config» - Не удалось отправить конфигурацию устройства пользователя на сервер;

«Start requst library» - Отправлен запрос на получение пользовательской библиотеки;

«Failed to send requst» - Не удалось отправить запрос на получение пользовательской библиотеки;

«Failed to recv library from server» - Не удалось получить ответ сервера, содержащий пользовательскую библиотеку;

«User have <количество> units in library» - Сервер прислал ответ с пользовательской библиотекой, в которой имеется выведенное число программ;

«Start generate key» - Запущен процесс генерации ключа;

«Start install software» - Отправлен запрос на получение ПО с сервера;

«file type <тип файла> name <имяо> – downloading» - Запущена загрузка файла с сервера;

«Start calculate soft hash» - Запущено вычисление хэша;

«Error when calculate hash» - Ошибка в процессе вычисления хэша;

«Server sent an error code: <сообщение>» - Сервер отверг запрос с сообщением;

«Successfull» - функция или действие в теле функции завершились успешно;

«Error, code = <код ошибки>» - функция или действие в теле функции завершились с ошибкой.

4 Рекомендации по использованию

Разработчику рекомендуется по возможности весь исполняемый код хранить в безопасных контейнерах, а в незащищенном исполняемом файле располагать исключительно загрузку данных из безопасных контейнеров.

При необходимости возможно использовать загрузку безопасного контейнера из безопасного контейнера.

Поскольку система подразумевает низкоуровневую загрузку исполняемого кода, использовать систему необходимо на максимально низком уровне.

5 Пример программы, использующей SDK

см. приложение Г.

Приложение Г

(справочное)

Листинг кода примера использования системы DataWall

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <stdlib.h>

#include <string.h>

#include <tchar.h>

#include <stdio.h>

#include <string>

#include "DataWallLoader.h"

void print(BYTE\* data, int size)

{

for (int i = 0; i < size; i++)

printf("%02X ", data[i]);

printf("\n");

}

typedef void (CALLBACK\* LOADLOGODATA)(BYTE\*, TCHAR\*);

typedef int (CALLBACK\* INITWIND)(HINSTANCE, HINSTANCE, LPSTR, int);

int CALLBACK WinMain(

\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPSTR lpCmdLine,

\_In\_ int nCmdShow)

{

BYTE\* key = DataWallLoader::LoadKey("TestSoft");

HMEMORYMODULE hDLL;

INITWIND InitializeWindow;

LOADLOGODATA LoadLogoData;

HRESULT hr;

char path[1024];

snprintf(path, 1024, "%s\\TestDLL.pak", lpCmdLine);

hDLL = DataWallLoader::LoadEncryptedLibrary(path, key);

if (!hDLL)

{

MessageBox(NULL,

\_T("Error when load dll"),

\_T("Error"),

NULL);

return -1;

}

LoadLogoData = (LOADLOGODATA)DataWallLoader::LoadEncryptedFunction(hDLL, "LoadLogoData");

if (!LoadLogoData)

{

MessageBox(NULL,

\_T("Error when load function"),

\_T("Error"),

NULL);

DataWallLoader::FreeEncryptedLibrary(hDLL);

return -1;

}

BYTE\* bLogo = NULL;

int bLogoSize = 0;

ContentType logoType = 0;

snprintf(path, 1024, "%s\\logo.pak", lpCmdLine);

hr = DataWallLoader::ReadFromContainer(

path,

key,

bLogo,

bLogoSize,

logoType);

if (FAILED(hr) || logoType != CONTENT\_IMAGE)

{

MessageBox(NULL,

\_T("Error when load logo"),

\_T("Error"),

NULL);

DataWallLoader::FreeEncryptedLibrary(hDLL);

return -1;

}

BYTE\* bText = NULL;

int bTextSize = 0;

ContentType textType = 0;

snprintf(path, 1024, "%s\\text.pak", lpCmdLine);

hr = DataWallLoader::ReadFromContainer(

path,

key,

bText,

bTextSize,

textType);

if (FAILED(hr) || textType != CONTENT\_TEXT)

{

MessageBox(NULL,

\_T("Error when load text"),

\_T("Error"),

NULL);

delete[] bLogo;

DataWallLoader::FreeEncryptedLibrary(hDLL);

return -1;

}

LoadLogoData(bLogo, (TCHAR\*)bText);

InitializeWindow = (INITWIND)DataWallLoader::LoadEncryptedFunction(hDLL, "InitializeWindow");

if (!InitializeWindow)

{

MessageBox(NULL,

\_T("Error when load function"),

\_T("Error"),

NULL);

delete[] bLogo;

delete[] bText;

DataWallLoader::FreeEncryptedLibrary(hDLL);

return -1;

}

// call the function

int result = InitializeWindow(hInstance, hPrevInstance,

lpCmdLine, nCmdShow);

if (result)

{

MessageBox(NULL,

\_T("Error when exec function"),

\_T("Error"),

NULL);

delete[] bLogo;

delete[] bText;

DataWallLoader::FreeEncryptedLibrary(hDLL);

return -1;

}

DataWallLoader::FreeEncryptedLibrary(hDLL);

delete[] bLogo;

delete[] bText;

printf("Success!\n");

return 0;// (int)msg.wParam;

}