**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Кафедра информационных систем управления

Лукашевич Юрий Юрьевич

**ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ ОБМЕНА ИНФОРМАЦИИ В ГРУППЕ РОБОТОВ С ИСПОЛЬЗОВАНИЕМ БЛОКЧЕЙН-ТЕХНОЛОГИИ**

Курсовой проект

студента 3 курса 2 группы

|  |  |
| --- | --- |
|  | **Руководитель:**  *Гутников Сергей Евгеньевич*  старший преподаватель  кафедры ИСУ |

**Минск 2020**

**Аннотация**

Лукашевич Ю.Ю. Обеспечение информационной безопасности обмена информации в группе роботов с использованием блокчейн-технологии. Курсовая работа / Минск: БГУ, 2021. – 22 с.

В курсовом проекте исследуются системы аутентификации пользователей в ОС Windows и Unix-Linux, используемые протоколы удаленной аутентификации. Подготовлено приложение на языке C++, аутентифицирующее пользователя в OC Windows и Linux

**Анатацыя**

Лукашэвiч Ю.Ю. Забеспячэнне інфармацыйнай бяспекі абмену інфармацыі ў групе робатаў з выкарыстаннем блокчейн-тэхналогіі. Курсавы праэкт / Мінск: БДУ, 2020. – 22 с.

У курсавым праэкце даследваюцца сістэмы аўтэнтыфікацыі карыстальнікаў у АС Windows і Unix-Linux, пратаколы аддаленай аўтэнтыфікацыі, якія у іх выкарыстоўваюцца. Падрыхтавана праграма на мове С++, якая аўтэнтыфіцыруе карыстальнікаў у АС Windows і Linux.

**Annotation**

Lukashevich Y.Y. Ensuring information security of information exchange in a group of robots using blockchain technology. Course project /Minsk: BSU, 2020 – 22 p.

In course project authentication systems of users in OS Windows and Unix-Linux, remote authentication protocols are researched. An application in C++, that authenticates user in OS Windows and Linux, is prepared.

**РЕФЕРАТ**

Курсовой проект, 22 с., 4 источника, 2 приложения

***Ключевые слова:*** РОЕВОЙ ИНТЕЛЛЕКТ, ТЕХНОЛОГИЯ БЛОКЧЕЙН, РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ, БЛОКЧЕЙН, УМНЫЕ КОНТРАКТЫ, SMART CONTRACTS.

***Объект исследования*** – системы удаленной аутентификации пользователей в ОС Windows и Unix-Linux.

***Цель работы*** – изучение концепций и особенностей реализации систем аутентификации пользователей и используемых протоколов в ОС Windows и Unix-Linux.

***Методы исследования*** – анализ, разработка приложений.

***Результат исследования*** – изученные концепции и особенности систем удаленной аутентификации и различия между ними в ОС Windows и Unix-Linux, разработанные приложения аутентификации пользователей.

***Областью применения*** является разработка и модификация систем удаленной аутентификации.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 5](#_Toc532510031)

[ГЛАВА 1. МЕТОДЫ АУТЕНТИФИКАЦИИ В СОВРЕМЕННЫХ ОС 6](#_Toc532510032)

[ГЛАВА 2. СИСТЕМА УДАЛЕННОЙ АУТЕНТИФИКАЦИИ В ОС WINDOWS 7](#_Toc532510033)

[2.1 Общая информация 7](#_Toc532510034)

[2.2 LANManager 8](#_Toc532510035)

[2.3 NTLANManager 9](#_Toc532510036)

[2.4 NTLMv2 9](#_Toc532510037)

[2.5 Kerberos 10](#_Toc532510038)

[ГЛАВА 3. СИСТЕМА УДАЛЕННОЙ АУТЕНТИФИКАЦИИ В ОС UNIX-LINUX 12](#_Toc532510039)

[3.1 Общая информация 12](#_Toc532510040)

[3.2 Хранение учетных данных 13](#_Toc532510041)

[3.3 SSH 14](#_Toc532510042)

[ГЛАВА 4. ПРИЛОЖЕНИЕ АУТЕНТИФИКАЦИИ ПОЛЬЗОВАТЕЛЯ В ОС WINDOWS И LINUX 16](#_Toc532510043)

[4.1 Постановка задачи 16](#_Toc532510044)

[4.2 Реализация приложения на ОС Windows 16](#_Toc532510045)

[4.3 Реализация приложения на ОС Linux 17](#_Toc532510046)

[ЗАКЛЮЧЕНИЕ 18](#_Toc532510047)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc532510048)

[ПРИЛОЖЕНИЯ 20](#_Toc532510049)

[Приложение 1 20](#_Toc532510050)

[Приложение 2 21](#_Toc532510051)

# **ВВЕДЕНИЕ**

Аутентификация — это незаменимая процедура для каждого пользователя, которая играет основную роль в организации защиты от похищения персональных данных злоумышленниками. Вопрос о безопасности данных стал еще более актуальным с появлением новых технологий удаленного доступа, в частности доступа через интернет. Механизмы удаленной аутентификации постоянно развиваются и улучшаются, однако вместе с ними развиваются и способы взлома и кибератак, используемые хакерами, поэтому проблема безопасного доступа становится все более сложной.

Используемые сегодня механизмы аутентификации охватывают широкий диапазон методик и технологий. Разработчикам приходится уметь анализировать и выбирать тот способ, который будет наиболее подходящим и обеспечивать достаточную безопасность, поэтому важно знать о принципах и особенностях этих механизмов.

# **ГЛАВА 1. МЕТОДЫ АУТЕНТИФИКАЦИИ В СОВРЕМЕННЫХ ОС**

Аутентификация – процесс проверки подлинности. Аутентификация представляет собой один из компонентов любой компьютерной системы управления доступом, состоящей из: идентификации, аутентификации, авторизации и отчетности.

Существует несколько методов аутентификации, которые используются в современном мире. Самая распространенная на данный момент – это *аутентификация с помощью многоразового пароля*. Это самый простой и дешевый способ аутентификации, поэтому он и является наиболее популярным. Однако на сегодняшний день существует много различных способов перехвата и кражи паролей, из-за чего этот вид аутентификации является не самым безопасным. Паролевая аутентификация постоянно улучшается, разрабатываются все более надежные алгоритмы хэширования паролей и их передачи.

Более надежная система аутентификации на основе пароля – *аутентификация с помощью многоразовых паролей*. Этот метод безопаснее многоразовых паролей, т.к. если злоумышленник и сможет перехватить пароль, то не сможет использовать его в дальнейшем.

Еще одним методом аутентификации является *аутентификация на основе сертификатов.* Сертификат — блок информации, содержащий данные, уникально идентифицирующие пользователя и его публичный ключ. Сертификат должен быть подписан заверяющим лицом – специальной сертифицирующей организацией, пользующейся доверием со стороны пользователей.  Система аутентификации может проверить электронную подпись, используя открытый ключ клиента, а у клиента в свою очередь есть закрытый ключ, которым он подписывает запрос [сервера аутентификации](https://powersecurity.org/ru/support/wiki/index.php?title=%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%B0%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8).

Одним из самых сложных и дорогих методов аутентификации является *биометрическая аутентификация.* Этот метод обеспечивают почти 100 % идентификацию, т.к. использует уникальную биометрическую информацию человека (отпечаток пальца, радужная оболочка глаза и т.д.). Однако этот метод требует дополнительные аппаратные устройства, которые зачастую являются слишком дорогими и трудными в реализации. Кроме того, эти устройства должны храниться в безопасном месте, т.к. информацию можно подделать, например с помощью муляжа.

В последнее время распространенной стала многофакторная аутентификация, которая сочетает разные виды аутентификации. Например, популярная двухфакторная аутентификация с помощью многоразового и одноразового пароля.

Далее мы более подробно рассмотрим системы аутентификации, реализованные в OC Windows и Unix-Linux, а также разберем протоколы аутентификации, которые эти системы используют.

# **ГЛАВА 2. СИСТЕМА УДАЛЕННОЙ АУТЕНТИФИКАЦИИ В ОС WINDOWS**

## **2.1 Общая информация**

За вход пользователя в систему Windows отвечает сервис Winlogon. Он запрашивает у пользователя его логин и пароль и затем протокол аутентификации шифрует его и передает запрос.

На каждом компьютере с OC Windows располагается база данных SAM (Security Accounts Manager).  В ней хранится вся информация, используемая для аутентификации пользователей Windows (имена, пароли) как при интерактивном входе в систему, так и при удаленном доступе к ней по сети. Информация в базе данных SAM хранится в основном в двоичном виде. По умолчанию изменять записи, находящиеся в базе данных SAM, при помощи программ нельзя, т. к. доступ к базе данных SAM запрещен для всех категорий пользователей операционной системы Windows. Доступ к ней обычно осуществляется через диспетчер учетных записей. Во время аутентификации введенные пользователем данные сравниваются с данными, находящимися в базе SAM на сервере.

В случае успешной аутентификации пользователя в ОС Windows создается токен (или маркер) доступа (access token). Токен доступа привязывается ко всем процессам, которые данный пользователь создает в течении сеанса работы.

Токен доступа, который включает:

- идентификатор пользователя (SID);

- идентификаторы всех групп, в которые входит пользователь;

- список прав пользователя на выполнение системных операций;

- другие данные о пользователе.

Протокол аутентификации должен выполнять по крайней мере две задачи. Во-первых, он должен безопасно передавать запросы от клиента к серверу. Во-вторых, он должен безопасно и надежно хранить пароль или маркер.

Далее рассмотрим протколы аутентификации, использующиеся в Windows.

## **2.2 LANManager**

Протокол LANManager (LM) или LANMANбыл разработан компанией IBM и применялся в ранних версиях Windows. На сегодняшний протокол день считается устаревшим и ненадежным, однако в целях совместимости все еще поддерживается. Разберемся, как он работает и почему является небезопасным.

После получения пароля, полученного при вводе пользователя, функция хэширования преобразует его по следующему правилу:

- Изменение пароля до вида 14-символьной цепочки добавлением или удалением символов;

- Преобразование символов в символы верхнего регистра;

- Разделение результата на 2 семибайтовых фрагмента;

- Шифрование по алгоритму DESASCII строки “KGS!@#$”с использованием каждого из двух фрагментов в качестве ключа;

- Результаты шифрования соединяются в одну цепочку, давая окончательный результат.

Полученный при попытке входа хэш пароля сравнивается с хэшем, хранящимся в базе данных SAM.

Этот алгоритм имеет несколько недостатков. Во-первых, пароль пользователя имеет ограничение в 14 символов. Во-вторых, все символы приводятся к верхнему регистру, что сокращает количество вариантов пароля, тем самым уменьшая пространство атаки. Кроме того, алгоритм разделяет пароль на два фрагмента, что дает возможность взлома каждого фрагмента по отдельности, и если пароль состоит из семи или менее символов, то второй фрагмент будет состоять из нулей. В некоторых случаях один фрагмент пароля может подсказать и второй(например, если мы нашли фрагмент пароля “sim”, то возможно, что вторая часть будет “salabim”), что еще больше упрощает задачу взлома.

Таким образом, сегодня протокол LANMAN является весьма ненадежным, и с помощью специальных инструментов, существующих в наши дни, пароли в лучшем случае (для взломщика) взламываются всего за несколько секунд.

## **2.3 NTLANManager**

Новый протокол аутентификации NTLANManager (или NTLM) появился в Windows NT и использовал более надежный метод хэшированияMD4. Более того, в отличие от LANMAN протокола, который использует только символы ASCII, NTLM совместим со всем наборов символов Unicode, что делает пароли более устойчивыми ко взлому, хотя их длина по-прежнему не превышает 14 символов.

Существует несколько типов NTLM протоколов. Одни используют LM-хэш, который был рассмотрен в предыдущем пункте, другие – NT-хэш на основе алгоритма MD4. Для совместимости с LANMAN протоколом NTLM требует вычисления и хранения в базе данных SAM обоих хэшированных значений пароля.

Рассмотрим принцип работы протокола:

- В ответ на запрос клиента сервер передает ему случайное число - запрос сервера;

- Клиент вычисляет 2 ответа на данный запрос по алгоритму DES: один на основе LM-хэша, второй на основе NT-хэша. Первый случай был рассмотрен ранее. Во втором случае хэш делится на три части по 56 бит, и каждая часть шифрует запрос сервера отдельно;

- Ответ передается обратно серверу, который в свою очередь также вычисляет значения хэша пароля, хранящегося в базе данных SAM;

- Хэши, вычисленные клиентом и сервером сравниваются, и аутентификация считается успешной, если они совпадают.

Огромной проблемой протокола NTLM является то, что в базе данных хранится небезопасный LM-хэш, который сводит на нет повышенную безопасность нового алгоритма шифрования. Взлом пароля происходит в 2 этапа: сначала взломщик перехватывает LM-хэш, из которого находится пароль в верхнем регистре, а затем из NT-хэш подбираются регистры символов пароля, что является не такой трудной задачей.

## **2.4 NTLMv2**

Однако вскоре выяснилось, что и NTLM протокол слишком уязвим, поэтому с выходом  [Windows NT 4.0](https://ru.wikipedia.org/wiki/Windows_NT_4.0)Microsoft представила вторую версию протокола NTLMv2, который был серьезно доработан в плане улучшений криптографической стойкости и противодействия распространенным типам атак. NTLMv2 до сих пор считается достаточно надежным протоколом, в отличие от NTLM.

Рассмотрим принцип работы:

- Как и в NTLM, в ответ на запрос клиента сервер передает ему случайное число - запрос сервера;

- В ответ клиент также генерирует случайное число, куда добавляется метка времени - запрос клиента;

- Запрос сервера объединяется с запросом клиента и от этой последовательности вычисляется HMAC-MD5 хэш;

- От полученного хэша берется еще один HMAC-MD5 хэш, ключом в котором является NT-хэш пароля пользователя. Полученный результат отправляется серверу вместе с запросом клиента;

- Сервер, получив ответ и запрос клиента, объединяет последний с запросом сервера и проделывает те же шаги алгоритма. При совпадении результатов аутентификация успешна.

Метка времени в запросе клиента передается для того, чтобы предотвратить атаки, в ходе которых взломщик записывает учетные данные и впоследствии их использует.

## **2.5 Kerberos**

С версии Windows 2000 протоколом аутентификации по умолчанию становится Kerberos, который был разработан в Массачусетском технологическом институте в 1983 году. Основным отличием от NTLM является то, что в отличие от NTLM протокола Kerberos предусматривает взаимную аутентификацию между клиентом и сервером, что позволяет защититься от фишинг-атак.

Основным компонентом Kerberos является **Центр распространения ключей** (KDC), который является доверенным центром аутентификации для всех участников сети и хранит их **долговременные ключи, которые** формируются на основе пароля. При этом долговременные ключи никогда не передаются по сети и располагаются в защищенных хранилищах.

Рассмотрим принцип работы протокола Kerberos:

- Клиент передает KDC открытым текстом свое имя, имя домена и метку текущего времени, зашифрованное долговременным ключом клиента.

- Получив эти данные KDC извлекает долговременный ключ данного пользователя и расшифровывает метку времени, которую сравнивает с собственным текущим временем, если оно отличается не более чем на заданное небольшое время, то метка считается действительной. Эта проверка является дополнительной защитой, так как не позволяет использовать для атаки перехваченные и сохраненные данные.

- KDC выдает клиенту **сеансовый ключ** и **билет на получение билета** (*ticket granting ticket, TGT*), который содержит копию сеансового ключа и сведения о клиенте, TGT шифруется с помощью долговременного ключа KDC, и никто кроме него билет расшифровать не может. Сеансовый ключ шифруется с помощью долговременного ключа клиента, а полученная от клиента метка времени возвращается обратно, зашифрованная уже сеансовым ключом. Билет на получение билета является действительным в течении 8 часов или до завершения сеанса пользователя.

- Клиент расшифровывает сеансовый ключ, а затем при помощи этого ключа метку времени и сравнивает ее с той, что он отправил KDC, если метка совпала, значит KDC тот, за кого себя выдает, так как расшифровать метку времени мог только тот, кто обладает долговременным ключом. В этом случае клиент принимает TGT и помещает его в свое хранилище.

- Когда клиент хочет обратиться к серверу, он снова обращается к KDC и посылает ему TGT, зашифрованную сеансовым ключом метку времени и имя сервера. KDC расшифровывает предоставленный ему TGT и извлекает оттуда данные о клиенте и его сеансовый ключ, обратите внимание, что сам KDC сеансовые ключи не хранит. Затем сеансовым ключом он расшифровывает данные от клиента и сравнивает метку времени с текущим. Если расхождения нет, то KDC формирует общий сеансовый ключ для клиента и сервера.

Теоретически теперь данный ключ следует передать как клиенту, так и серверу. Но KDC имеет защищенный канал и установленные доверительные отношения только с клиентом, поэтому он поступает по-другому. Экземпляр сеансового ключа для клиента он шифрует сессионным ключом, а копию сеансового ключа для сервера он объединяет с информацией о клиенте в **сеансовый билет** (*session ticket*), который шифрует закрытым ключом сервера, после чего также отправляет клиенту, дополнительно зашифровав сессионным ключом.

Таким образом клиент получает сессионный ключ для работы с сервером и сессионный билет. Получить содержимое билета, как и TGT, он не может, так как не располагает нужными долговременными ключами.

- Клиент предъявляет серверу сеансовый билет и метку времени, зашифрованную сессионным ключом. Сервер расшифровывает билет, извлекает оттуда свой экземпляр ключа и сведения о клиенте, затем расшифровывает метку времени и сравнивает ее с текущим. Если все нормально, то он шифрует полученную метку своим экземпляром сессионного ключа и посылает назад клиенту. Клиент расшифровывает ее своим сеансовым ключом и сравнивает с тем, что было послано серверу. Совпадение данных свидетельствует о том, что сервер тот, за кого себя выдает.

# **ГЛАВА 3. СИСТЕМА УДАЛЕННОЙ АУТЕНТИФИКАЦИИ В ОС UNIX-LINUX**

## **3.1 Общая информация**

Основу системы аутентификации в OC unix-linux составляют PAM – подключаемые модули аутентификации (Pluggable authentication modules). Они представляют собой библиотеки, хранящиеся в каталоге /lib/security, которые по-своему реализуют механизм аутентификации.

Чтобы внедрить аутентификацию в приложение, достаточно просто выбрать и подключить соответствующий PAM модуль. Это позволяет отделить систему аутентификации и при изменении метода аутентификации не нужно будет изменять остальное приложение, достаточно изменить только PAM модуль.

Тип PAM - модуля должен быть одним из следующих:

*- auth*: Такой модуль проверяет наличие пользователя в системе, спрашивает его имя, разрешает или нет доступ в ту или иную группу (независимо от записей в файле /etc/groups) и вообще способен давать привилегии (конечно специально предназначенные для этого).

*- account:* Этот модуль не занимается аутентификацией, а позволяет контролировать распределение ресурсов системы для тех или иных пользовательских бюджетов.

*- session:* Связан с вещами, которые могут происходить перед тем, как пользователь получит доступ к той или иной службе. Например, ведение записей в системных журналах.

*- password:* Модуль, занимающийся непосредственно проверкой паролей на подлинность, на слабость и т.д.

Схема аутентификации в приложении, использующем PAM, может быть представлена следующим образом:

1. Приложение инициализирует библиотеку PAM;

2. PAM в соответствии с конфигурационным файлом для приложения обращается к требуемым модулям

3. Модули выполняют возложенные на них действия

4. Приложению возвращается результат операции

## **3.2 Хранение учетных данных**

Учетные данные пользователей и хэши их паролей хранятся в специальном файле /etc/passwd, доступ для чтения, к которому разрешен всем пользователям, а для записи – только root пользователю. Запись пользователя в этом файле выглядить следующим образом:

*example:hash:100:200:Ivan Ivanov:/home/example:/bin/bash*

Эта запись состоит из полей, разделенных двоеточиями:

* Логин пользователя (example);
* Хэш пароля (hash);
* Идентификатор (100);
* Идентификатор группы (200);
* Полное имя пользователя (IvanIvanov);
* Домашний каталог (/home/example/);
* Версия оболочки Unix, которая запускается при входе (/bin/bash/);

Однако хранение хэшей в открытом виде небезопасно и дает возможность злоумышленникам без особого труда восстановить пароль. Тем не менее, если запретить доступ для чтения для непривилегированных пользователей, может нарушиться работа приложений, которые используют другие данные помимо пароля из этого файла. В некоторых версиях ОС для решения этой проблемы пароли переносятся в отдельный файл /etc/shadow/, который закрыт обычным пользователям как для записи, так и для чтения.

Записи файла shadow, соответствующие записям файла passwd из приведенного выше примера, будут такими:

*test:!!:11111::::::*

Формат записи файла shadow (на примере последней записи) следующий:

·       логическое имя пользователя (test);

·       поле хэша пароля, которое состоит из трех подполей, разделённых знаком «$»: подполя условного номера односторонней функции, подполя криптографической «соли», добавленной к паролю при его хешировании, и подполя, собственно, хэша. Если пароль не задан, то вместо хэша ставится “!!”, как в данном пример;

·       количество дней, прошедших с 1 января 1970 года (момент начала отсчета времени Unix – Unixepoch) до последнего изменения пароля (11111);

·       количество дней до разрешенной смены пароля;

·       количество дней до необходимой смены пароля;

·       количество дней до предупреждения об истечении пароля;

·       количество дней до неактивности учетной записи;

·       количество дней с 1 января 1970 года до истечения действия учетной записи;

В ОС Unix-Linux одним из основных протоколов аутентификации является Kerberos, однако помимо него, широко распространен протокол SSH.

## **3.3 SSH**

Аутентификация с помощью SSH может осуществляться различными способами:

1. *Аутентификация на основе имен хостов.*

Удаленный хост успешно аутентифицирует пользователя, если выполняется два условия: имя пользователя и хоста, с которого этот пользователь осуществляет логический вход, содержится в определенных файлах на удаленном хосте, и удаленный хост успешно верифицировал открытый ключ клиентского компьютера, предъявленный в процессе данной сессии аутентификации. Верификация выполняется путем сравнения этого открытого ключа с   хранящимися на удаленном хосте ключами, которые передавались клиентским хостом во время других, предыдущих успешных логических входов данного пользователя, в том числе использовавших другие методы аутентификации. Если же предъявленный ключ клиента не совпадает ни с одним из имеющихся у удаленного хоста ключей, то он выдает предупреждение о возможной фальсификации и аутентификация не считается успешной.  Ключи хостов для этого вида аутентификации генерируются автоматически при установке пакета SSH в операционной системе хоста, их основным назначением является создание защищенного канала.  Поэтому сам по себе ключ клиента не может однозначно его аутентифицировать - для этого и требуется дополнительная информация его аутентичности в виде факта успешной аутентификации другим способом.

1. *Аутентификация с помощью публичных ключей.*

Используя утилиту ssh-keygen, пользователь должен вручную сгенерировать открытый и закрытый ключи по алгоритму RSA, поместить каждый из них в отдельный файл на своем клиентском хосте, а затем скопировать файл с открытым ключом в свой домашний каталог на хосте-сервере, в который он хочет удаленно входить.

- Когда клиент хочет аутентифицироваться, он передает серверу свое имя пользователя и публичный ключ.

- Сервер проверяет, совпадает ли переданный ключ пользователя с хранящимся у него, если совпадает, то отправляет клиенту сообщение, зашифрованное публичным ключом.

- Клиент должен расшифровать сообщение с помощью своего приватного ключа и отправить серверу расшифрованное сообщение.

- Если сообщение расшифровано верно, аутентификация считается успешной.

Как мы видим, приватный ключ никогда не передается и хранится исключительно у клиента. В качестве дополнительной безопасности его можно защитить паролем.

1. *Аутентификация на основе слова-вызова.*

Передача слова-вызова от удаленного сервера и хеш-результата от хоста-клиента происходит по защищенному каналу, который образуется на основе автоматически сгенерированных ключей хоста и сервера.

1. *Аутентификация на основе пароля.*

Отличается от аутентификации на основе слова-вызова тем, что клиент передает по сети пароль, а не хэш от него. Поскольку пароль и в этом случае передается по защищенному каналу, то этот способ также считается защищенным, хотя и менее защищенным, чем предыдущие.

Клиент SSH пытается использовать различные способы аутентификации в том порядке, в котором они описаны, то есть начиная с аутентификации на основе хостов и кончая аутентификацией на основе паролей. Переход к более низкоприоритетному способу аутентификации происходит в том случае, если очередной способ аутентификации не поддерживается обеими сторонами.  Но пользователь также может задать предпочтительный способ аутентификации принудительно.

# **ГЛАВА 4. ПРИЛОЖЕНИЕ АУТЕНТИФИКАЦИИ ПОЛЬЗОВАТЕЛЯ В ОС WINDOWS И LINUX**

## **4.1 Постановка задачи**

Для демонстрации реализаций систем аутентификации в разны операционных системах требуется подготовить приложение на языке C++ на каждой из рассмотренных ОС, которое будет аутентифицировать пользователя.

## **4.2 Реализация приложения на ОС Windows**

Для реализации задачи использовался LogonUser API. Для его использования необходимо подключить заголовочный файл windows.h.

Выполнение приложения начинается с запроса имени пользователя и пароля. Полученные данные записываются в массивы символов.

Затем функция LogonUser пытается аутентифицировать пользователя на компьютере. Пользователь определяется при помощи имени пользователя (логин) и подтверждает подлинность пользователя паролем. Если функция вызвана успешно, то мы получаем дескриптор, которая представляет вошедшего пользователя.

При успешном выполнении функции LogonUser, происходит вызов следующей функции – ImpersonateLoggedOnUser, в качестве аргумента которой мы передаём дескриптор hToken. Эта функция изменяет security context вызывающего потока на security context нашего нового пользователя. Убедиться в этом можно вызвав функцию GetUserName. Она вернет имя текущего владельца потока.

Функцией RevertToSelf возвращаем назад security context и снова выводим имя владельца потока – выведется первоначальный пользователь.

## **4.3 Реализация приложения на ОС Linux**

Для реализации данной задачи использовалась библиотека linux-pam.

Перед началом работы с системой аутентификации получаем имя пользователя из аргументов командной строки.

Затем инициализируем приложение посредством вызова функции pam\_start. Получаем дескриптор pamh.

pam\_set\_item позволяет задать необходимые для аутентификации данные.

pam\_authenticate производит аутентификацию. После этого с помощью функции pam\_acct\_mgmt определяем валидность аккаунта. Если аккаунт прошёл валидацию, то вызываются следующие функции:

pam\_setcred для получения полномочий данного пользователя;

pam\_open\_session для открытия PAM сессии;

pam\_getenvlist для экспорта PAM окружения. Таким образом мы получаем uid и groups данного пользователя. Используя эти параметры, мы можем запустить терминал от имени пользователя, и воспользоваться его полномочиями.

После совершения необходимых действий в терминале, освобождаем ресурсы.

Для компиляции используется команда gcc -o autt au.c -lpam -lpam\_misc

Для запуска используется команда sudo. /autt <username>, где <username> – имя пользователя для аутентификации.

# **ЗАКЛЮЧЕНИЕ**

В процессе выполнения данного курсового проекта:

* Изучены основные методы аутентификации пользователей в современных ОС;
* Рассмотрены особенности реализации системы удаленной аутентификации в ОС Windows и Unix-Linux и проведен их анализ;
* Разработаны приложения аутентификации на языке С++ в ОС Windows и Linux.

Итак, мы выяснили, что в Windows и Unix-Linux используется разный подход аутентификации пользователей и хранения их данных. В ОС Unix-Linux используется менее безопасная система хранения учетных данных пользователей, однако реализована более удобная система внедрения аутентификации в приложения с помощью PAM – модулей.

В обоих операционных системах самым распространенным и безопасным протоколом удаленной аутентификации на сегодняшний день является протокол Kerberos, однако могут использоваться и другие в зависимости от ОС.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1.Уилльям Бозуэл, Внутренний мир Windows Server 2003, SP1 и R2., ИД «Вильямс», 2006 г.,

2.Вахалия Ю., Unix изнутри, СПб, 2003г.

3.Таненбаум Э., М. ван Стеен. Распределенные системы. Принципы и парадигмы, СПб, 2003г.

4.Ричард Э. Смит. Аутентификация: от паролей до открытых ключей. — М.: Вильямс, 2002.

# **ПРИЛОЖЕНИЯ**

## **Приложение 1**

Часть кода приложения для Windows:

while (true) {

if (!LogonUser(user, "USER-PC", pass,

LOGON32\_LOGON\_NETWORK, LOGON32\_PROVIDER\_DEFAULT, &hToken))

{

printf("Login error...\n");

break;

}

if (!ImpersonateLoggedOnUser(hToken))

{

printf("Impersonation error...\n");

break;

}

success = TRUE;

ZeroMemory(currUser, sizeof(currUser));

if (!GetUserName(currUser, &nSize))

{

wprintf(L"error...\n");

break;

}

printf("Current user is %s\n\n", currUser);

break;

}

SecureZeroMemory(pass, sizeof(pass));

if (success)

{

printf("Undo the impersonation ...\n");

if (!RevertToSelf())

{

printf("error...\n");

}

ZeroMemory(currUser, sizeof(currUser));

nSize = ARRAYSIZE(currUser);

if (!GetUserName(currUser, &nSize))

{

printf("error...\n");

}

printf("Current user is %s\n\n", currUser);

}

## **Приложение 2**

Часть кода приложения для Linux:

/\* initialize PAM \*/

pamc.conv = &converse;

pam\_start("sudo", user, &pamc, &pamh);

/\* set some items \*/

gethostname(hostname, sizeof(hostname));

if ((pam\_err = pam\_set\_item(pamh, PAM\_RHOST, hostname)) != PAM\_SUCCESS){

printf("error host\n");

}

user = getlogin();

if ((pam\_err = pam\_set\_item(pamh, PAM\_RUSER, user)) != PAM\_SUCCESS){

printf("error user\n");

}

tty = ttyname(STDERR\_FILENO);

if ((pam\_err = pam\_set\_item(pamh, PAM\_TTY, tty)) != PAM\_SUCCESS){

printf("error tty\n");

}

/\* authenticate the applicant \*/

if (pam\_err == PAM\_SUCCESS){

pam\_err = pam\_authenticate(pamh, 0);

}

if (pam\_err != PAM\_SUCCESS){

printf("authentification failed\n");

}

/\* establish the requested credentials \*/

if ((pam\_err = pam\_setcred(pamh, PAM\_ESTABLISH\_CRED)) != PAM\_SUCCESS){

printf("establish the requested credentials failed\n");

}

/\* authentication succeeded; open a session \*/

if ((pam\_err = pam\_open\_session(pamh, 0)) != PAM\_SUCCESS){

printf("open a session failed\n");

}

/\* get mapped user name; PAM may have changed it \*/

pam\_err = pam\_get\_item(pamh, PAM\_USER, (const void \*\*)&user);

if (pam\_err != PAM\_SUCCESS || (pwd = getpwnam(user)) == NULL){

printf("get mapped user name failed\n");

}