Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Рыбинский государственный авиационный технический университет

имени П.А. Соловьева»

Факультет радиоэлектроники и информатики Кафедра Математическое и программное обеспечение ЭВС

**ОТЧЕТ**

**по преддипломной практике**

Студент группы Козин О.И.

*(Код)* *(Подпись, дата)* *(Фамилия И. О.)*

Руководитель Черных Ю.А.

*(Уч. степень, звание)* *(Подпись, дата)* *(Фамилия И. О.)*

Оценка

*(Подпись, дата)* *(Фамилия И. О.)*

Рыбинск 2019

Оглавление

[Введение 3](#_Toc9463119)

[Раздел 1 6](#_Toc9463120)

[1.1 Тема задания 6](#_Toc9463121)

[1.2 Анализ предметной области 6](#_Toc9463122)

[1.3 Обзор и анализ литературы 18](#_Toc9463123)

[1.4 Обзор аналогов 18](#_Toc9463124)

[1.5 Формулировка проблемы и актуальность 19](#_Toc9463125)

[1.6 Цели и задачи 19](#_Toc9463126)

[Раздел 2 21](#_Toc9463127)

[2.1 Функциональные и технологические требования 21](#_Toc9463128)

[2.2 Обоснование выбора технологий и инструментов разработки 21](#_Toc9463129)

[2.3 Техническое задание на разработку и проектирование программного обеспечения 29](#_Toc9463130)

[Раздел 3 30](#_Toc9463131)

[3.1 Описание методики тестирования 30](#_Toc9463132)

[3.2 Набора тестов,программа и методикаиспытаний 31](#_Toc9463133)

[Раздел 4 35](#_Toc9463134)

[4.1 Описание алгоритма 35](#_Toc9463135)

[4.2 Результат тестирования 36](#_Toc9463136)

[Заключение 38](#_Toc9463137)

[Список используемой литературы 39](#_Toc9463138)

# Введение

Цель прохождения практики - получить профессиональные умения и опыт профессиональной деятельности, формирующие следующие компетенции для выполнения выпускной квалификационной работы:

1. Способность к формализации в своей предметной области с учетом ограниченийиспользуемых методов исследования;
2. Готовность к использованию методов и инструментальных средств исследованияобъектов профессиональной деятельности;
3. Готовность обосновать принимаемые проектные решения,осуществлятьпостановку и выполнение экспериментов по проверке их корректности и эффективности;
4. Способность готовить презентации,оформлять научно-технические отчеты порезультатам выполненной работы, публиковать результаты исследований в виде статей и докладов на научно-технических конференциях.

Задачи учебной практики могут быть сформулированы следующим образом.

Во время практики студент должен:

*изучить:*

* синтаксис выбранного языка программирования, особенности программирования на нем, стандартные библиотеки языка программирования;
* особенности используемой среды программирования и системы управления базами данных;
* компоненты программно-технических архитектур, существующие приложения и интерфейсы взаимодействия с ними;
* требования к структуре и форматам хранения тестовых наборов данных;
* внутренние нормативные документы, регламентирующие требования к программному коду, порядок отражения изменений в системе контроля версий;
* методологии разработки программного обеспечения и технологии программирования;
* методы и средства проектирования программного обеспечения;
* типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке программного обеспечения;
* действующие стандарты, технические условия, положения и инструкции по эксплуатации программных средств вычислительной техники, по про­граммам испытаний и оформлению технической документации;

*освоить:*

* выбранные языки программирования для написания программного кода;
* выбранную среду программирования и средства системы управления базами данных;
* возможности имеющейся технической и/или программной архитектуры;
* внутренние нормативные документы, регламентирующие требования к программному коду, порядок отражения изменений в системе контроля версий;
* методы использования выбранной среды программирования для разработки процедур проверки работоспособности программного обеспечения на выбранном языке программирования;
* методы и приемы отладки дефектного программного кода;
* типовые решения и шаблоны проектирования программного обеспечения;
* методы и средства проектирования программного обеспечения, структур данных, баз данных, программных интерфейсов

В процессе прохождения учебной практики необходимо изучить технологии: Kerberos, OpenID Connect, SSO, LDAP, Java. Затем, используя полученные знания, разработать систему централизованной авторизации на основе LDAP и современных протоколов аутентификации.

# Раздел 1

## 1.1 Тема задания

Разработка программной системы с функциями дополненной реальности для обучения конструированию и эксплуатации технологического оборудования.

## 1.2 Анализ предметной области

Процессы аутентификации и авторизации позволяют реализовать разделения прав доступа и защиту конфиденциальной информации от внешних угроз. Это необходимо в большинстве программных продуктов, особенно коммерческих. Также реализация конкретной системы может значительно повысит удобство работы пользователей, что зачастую ведёт ещё большему увеличению безопасности.

Для успешного прохождения авторизации необходимо пройти несколько этапов:

1) Идентификация – заявление о том, кем является пользователь, обычно выражается в предоставлении своего уникального идентификатора-имени.

2) Аутентификация – проверка подлинности пользователя. То есть действительно ли пользователь тот, за кого себя выдаёт. Самый распространённый способ — это подтверждение с помощью секретного пароля. Кроме того, аутентификация может быть односторонней и двусторонней (взаимной). При односторонней, проверки требует только тот, кто запрашивает доступ. При двусторонней запрашиваемый ресурс тоже должен подтвердить себя.

3) Авторизация – выдача соответствующих прав доступа к запрашиваемому ресурсу. Этот процесс позволяет использовать результаты работы аутентификации и выдать пользователю только то, что ему разрешается согласно его статусу.

Успешное последовательное прохождение этих этапов позволяет гарантировать защищённость информации, к которой осуществляется доступ.

Но, так как информация имеет ценность, то найдутся люди, желающие ей завладеть. При этом они могут использовать различные методы чтобы нарушить нормальное функционирование этапов авторизации или получить доступ к конфиденциальной информации напрямую. Поэтому появляется необходимость в полной защите данных: как при хранении, так и при передаче.

На практике процессы авторизации обычно происходят с использованием компьютерной сети, которая не может быть полностью ограждена от физического доступа злоумышленника. В таком случае приходится учитывать, что любая информация, передаваемая по сети может быть известна кому угодно. Следовательно, необходимо использовать методы, позволяющие преобразить данные так, что даже при их перехвате неавторизированным пользователем конфиденциальность не будет нарушена.

Такие методы предоставляет криптография. Криптография - это наука об обеспечении безопасности данных. Она занимается поисками решений четырех важных проблем безопасности - конфиденциальности, аутентификации, целостности и контроля участников взаимодействия. Основным способом решения этих проблем является шифрование. Шифрование.

Шифрование - это преобразование данных определённым способом-алгоритмом в нечитабельную форму, с возможностью возвращения исходного состояния, используя специальные ключи шифрования.

На практике приходится считаться со сложными структурами данных и с аппаратные ограничениями, поэтому отдельных алгоритмов шифрования недостаточно. Нужны комплексные решения описывающие все этапы передачи данных, способные надёжно и эффективно решать проблему защиты информации. Такими решениями являются криптосистемы.

Они включают в себя набор правил и алгоритмов, регламентирующих манипуляции с данными, а также способы создания и распространения ключей. Криптографические системы можно классифицировать по типу используемых алгоритмов шифрования:

1. На основе симметричных алгоритмов.
2. На основе асимметричных алгоритмов.

Симметричные криптосистемы, также называемые системами с секретным ключом – способ шифрования, в котором один ключ используется как для шифрования, так и для расшифровывания. Является первым подходом к сокрытию информации изобретённым человеком.

Главной особенностью симметричной криптосистемы является то, что что ключ алгоритма шифрования должен оставаться в секрете от посторонних. Поэтому должна обеспечиваться защита ключа на протяжении всего его существования.

Кроме возможности потери конфиденциальности ключа, также существует опасность применения криптоанализа к зашифрованным данным. Поэтому выбранный алгоритм шифрования должен быть достаточно стойким, для недопущения расшифровки информации за приемлемое время.

Для симметричных алгоритмов уровень криптостойкости, обычно, определяется размером ключа шифрования, что эквивалентно полному перебору его значений.

Для современных систем желателен ключ длинной не менее 128 двоичных разрядов (128 бит). (про крипто стойкость уточнить)

Асимметричные криптосистемы в своей работе используют пару ключей: закрытый и открытый. Открытый ключ применяется для шифрования и может передаваться по незащищённому каналу связи. Расшифровка производится с помощью закрытого ключа, который должен оставаться в секрете.

Реализация асимметричных алгоритмов шифрования основана на применении односторонних функций, то есть таких функций, что, зная несложно найти значение F(x), но вычисление x по F(x) невозможно за приемлемое время. Теоретически, зная F(x), x можно подобрать, перебирая все возможные значения x и применять к ним функцию, до тех пор, пока результат не совпадёт с F(x). Однако практически, при достаточной размерности множества X, выполнить такой перебор, за приемлемое время, невозможно.

Классический пример такой функции – это функция основанные на принципе обратного модульного возведения в степень, также называемые задачей дискретного логарифмирования.

Но получается, что односторонняя функция не может использоваться в качестве функции шифрования. Так как хоть она и позволяет надёжно зашифровать данные, но расшифровать их некто не сможет. Для решения этой проблемы используются односторонние функции с секретом. Это особый вид функций, которые имеют некоторый секрет, позволяющий быстро вычислить обратное значение функции. Именно на этом принципе основаны большинство асимметричных криптографических систем.

В итоге оба подхода имеют свои преимущества и недостатки. Симметричные системы проще в реализации, за счёт более простых операций, а скорость шифрования в среднем на несколько порядков выше, чем у аналогичных по защищённости систем с открытым ключом. Единственным серьёзным недостатком является сложность обмена секретного ключа. Поэтому для применения симметричных систем на практике необходимо решить проблему надёжного распределения ключей между участниками процесса передачи данных.

При работе в публичных сетях, требуется быть уверенным в подлинности ключа. А именно в том, что полученный от пользователя ключ действительно сгенерирован им. Такой функционал может предоставить механизм электронной подписи.

Обычно, в большинстве реальных систем, электронная подпись основана на принципах асимметричного шифрования, только применяет их иначе. Так же генерируется пара ключей: открытый, закрытый. Но для шифрования (подписи) данных применяется закрытый, соответственно для расшифровки (проверки подписи) открытый. Такой подход позволяет любому владельцу открытого ключа проверять подпись, а возможность подписать, остаётся только за хранителем закрытого ключа. Таким образом проведя подписание каких-либо информации, мы генерируем электронную подпись и можем отправлять её вместе с исходными данными. Далее, зная отправителя и его открытый ключ, можно применить его к подписи и получить исходный файл.

Однако, получается, что каждый обладатель открытого ключа может получить из подписи оригинальные данные, а они должны быть секретны. Кроме того, подписанный файл по размеру будет сопоставим с оригинальным, что удвоит нагрузку на передающую среду. Поэтому в электронная подпись невозможна без хеш-функций.

Хеш-функция – осуществляет преобразование (хеширование) набора входных данных произвольной длинны, в выходную последовательность определённой длинны. При этом для каждого набора результат хеширования будет уникальным, а провести обратное преобразование невозможно.

Таким образом подписывая не сами данные, а их хеш, общедоступным окажется именно хеш, что безопасно. И размер подписи теперь возможно уменьшить, выбрав подходящую хещ-функцию.

Одной только проверки подлинности ключей недостаточно для сохранения конфиденциальности информации. Кроме этого, очевидно, необходимо скрывать сами значения ключей. Это можно сделать если использовать асимметричное шифрование на ключи. Так как размеры ключей небольшие, то потеря ресурсов на зашифровку-расшифровку будет незначительной.

Используя цифровую подпись и асимметричное шифрование можно реализовать безопасное распределение ключей для симметричных криптосистем.

Но такая система связи остаётся уязвимой для злоумышленника, который представляется доверенным пользователем, но отдаёт свой открытый ключ асимметричного шифрования, таким образом подменяя его.

Для решения этой проблемы открытый ключ доверенного пользователя вместе с сопроводительной информацией: именем, сроком действия и прочим подписывается центром сертификации. Предполагается, что центр сертификации честный и не подпишет ключ злоумышленника. А также то, что центр сертификации распространяет свой открытый ключ максимально широко, и желающие пройти аутентификацию, ещё до инициализации процесса обмена ключами, будут иметь открытый ключ такого центра, и злоумышленник ничего не сможет с этим поделать.

Этот процесс называется сертификацией. А подписанные центром сертификации открытые ключи, включающие в себя также прочие данные о владельце, называются цифровыми сертификатами. (добавить про центры сертификации)

Таким образом современные протоколы аутентификации и авторизации должны включать в себя множество механизмов обеспечения безопасности.

Протоколы в общем можно классифицировать по методу аутентификации:

1. По паролю
2. По сертификатам
3. По одноразовым паролям
4. По токенам

Самым простым методом является простая авторизация по паролю. Она предполагает наличие у пользователя и у сервиса пары логин-пароль, с помощью которых и происходит подтверждение личности. Является наиболее не безопасным, так как перехват пароля ведёт к полной потере конфиденциальности. Поэтому все коммуникации должны происходить в защищённой среде. Например, используя HTTPS. Одним из протоколов реализующих такой метод аутентификации является HTTP authentication.

Клиент

Сервер

Запрос ресурса

Запрос на аутентификацию

Учётные данные

Отчёт о выполнении

Рисунок 1. Пример работы HTTP аутентификации со схемой Basic

1. Клиент запрашивает ресурс, используя GET запрос
2. Сервер отвечает о необходимости аутентификации кодом 401
3. Клиент снова запрашивает ресурс и передаёю свои данные в заголовке Authorization в открытом виде.
4. Сервер проводит проверку данных пользователя и в случае успеха возвращает код 200.

Аутентификация по сертификатам предполагает наличие доверенного сервера certificate authority (CA). Этот сервер выступает гарантом подлинности сертификатов, которые он подписал. Сертификаты содержат данные идентифицирующие конкретного пользователя. Если сервис доверяет CA который подписал сертификат, то он принимает его для аутентификации.

Клиент

Сервис

Подпись сертификата

Подписанный сертификат

Рисунок 2. Аутентификация по сертификатам

Аутентификация по одноразовым паролям обычно является дополнительным способом повышения безопасности. С помощью её реализуется концепция two-factor authentication (2FA). Согласно ей, пользователь должен использовать два типа учётных данных. Например, обычный пароль к учётной записи и код из смс-сообщения. Для реализации одноразовых паролей могут использоваться как программные средства, так и специальные устройства.

Аутентификация по токенам чаще всего используется для построения больших распределённых систем. В ней функционал по аутентификации делегируется сервисами отдельному приложению – серверу аутентификации или identity provider (IdP). Это позволяет упростить структуру сервисов. А также открывает возможность реализовать технологию единого входа Single Sign-On (SSO). Она позволяет однажды аутентифицирующемуся пользователю иметь доступ ко всем сервисам, находящимся в зоне ответственности IdP. По такому же принципу работает вход через учётные записи социальных сетей.

Клиент

IdP

Сервис

Учётные данные

Токен сервиса

Токен сервиса

Отчёт о выполнении

Рисунок 3. Общая схема аутентификации по токену

В общем виде процесс аутентификации по токену выглядит следующим образом:

1. Клиент проводит аутентификацию с IdP. Способом может быть любым, но обычно это протоколы Kerberos, OpenID Connect и аналогичные.
2. Теперь Клиент может запрашивать к IdP токен для конкретного сервиса.
3. Используя токен, Клиент аутентифицируется в сервисе.

В некоторых ситуациях, например, когда клиентом является браузер невозможно выполнять сложные последовательности запросов. Для таких случаев разработана схема с пассивным клиентом.

Клиент

IdP

Сервис

Запрос учётных данных

Учётные данные

Запрос ресурса

Запрошенный ресурс

Токен

1

2

Перенаправление запроса

3

4

5

6

7

Рисунок 4. Схема аутентификации с пассивным клиентом, с перенаправлением запросов

Формат токенов очень важен при проведении аутентификации, поэтому стандартизирован. Основные используемые форматы токенов:

1. JSON Web Token (JWT)
2. Simple Web Token (SWT)

JWT – это открытии стандарт, который описывает формат сообщений в виде JSON файлов. Согласно нему токен состоит из заголовка, хранящего собственное описание, основной части, в которой хранится пользовательская информация, а также электронной подписи, удостоверяющей весь файл. В стандарте определены специальные имена, определяющие конкретные значения: iss, exp и т.д.

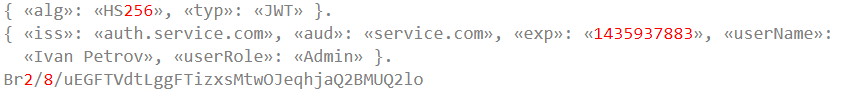


Рисунок 5. Пример не зашифрованного токена формата JWT

SWT – открытый стандарт, имеет простую структуру. Что позволяет просто использовать для передачи заголовки протоколов тапа HTTP. Имеет зарезервированные имена. Использует симметричное шифрование и цифровую подпись.



Рисунок 6. Пример не зашифрованного токена формата SWT

Из существующих протоколов аутентификации с токеном можно выделить наиболее популярные. Одним из таких является Kerberos.

Kerberos может предоставлять как сетевую аутентификацию, так и безопасную авторизацию без необходимости повторного ввода пароля или предоставления других удостоверяющих данных.

Особенностью Kerberos является то, что пароли или иные удостоверяющие данные никогда не пересылаются по сети. Это достигается благодаря наличию третьей доверенной стороны – центра распространения ключей (key distribution center (KDC)). KDC – хранит все учётные данные пользователей и сервисов, и предоставляет функции аутентификации. Для обеспечения безопасности протоколом предусмотрен следующий формат проведения аутентификации:

1. Пользователь, желая аутентифицироваться, обращается к KDC и передаёт ему в открытом виде данные о себе и о желаемом ресурсе.
2. KDC, посылает в ответ разрешение на получение разрешения (TGT), которое интерпретируется клиентом как неразбираемый набор бит, а также ключ сессии. Этот ключ генерируется случайным образом и зашифровывается секретным ключом пользователя, хранящимся в базе данных KDC.
3. Теперь пользователь, применяя свой закрытый ключ может получить сессионный ключ. И зашифровав им TGT, отправляет обратно на KDC.
4. KDC применяет сохранённый сессионный ключ получает TGT и если он соответствует исходному, значит пользователь тот за кого себя выдаёт. Генерируется новое сообщение. В него записываются данные необходимые для аутентификации и авторизации (опционально) пользователя сервисом – Service Ticket (ST). А также генерируется новый сессионный ключ, но на этот раз для взаимодействия сервиса и пользователя. Этот ключ вместе с ST шифруется закрытым ключом это сервиса. А чтобы пользователь мог получить доступ к нему, то его копия шифруется сессионным ключом пользователя. Всё вместе отправляется пользователю.
5. Пользователь, расшифровав ключ для общения с сервисом запоминает его. И посылает ST сервису.
6. Сервис расшифровывает ST своим секретным ключом, оттуда же берёт и сессионный ключ. По полученным данным сервис может решать какой доступ предоставить.

OpenId Connect – это надстройка над протоколом авторизации OAuth 2.0, предоставляющая функции аутентификации. OIDC предполагает наличие отдельного доверенного сервера аутентификации, благодаря чему поддерживает технологию SSO, позволяет отделить функционал по аутентификации от конкретных приложений и гарантирует безопасность. Очень популярный протокол, поддерживается практически всеми крупными компаниями, например, Google, Facebook.

Для передач данных в процессе аутентификации используется стандарт токенов JWT или SWT. Процесс авторизации, в общем случае, выглядит следующим образом:

1. Пользователь запрашивает ресурс.
2. Ресурс перенаправляет пользователя на сервер аутентификации.
3. Пользователь вводит свои учётные данные на форме сервера аутентификации.
4. Сервер аутентификации проверяет данные пользователя по своей базе, и вы даёт токен доступа (access token).

Имея access token пользователь может получить доступ к ресурсу.

## 1.3 Обзор и анализ литературы

При разработке использовалась литература, связанная с методами криптографии, теоретическими обоснованиями алгоритмов аутентификации и с описанием конкретных протоколов.

## 1.4 Обзор аналогов

При постановке задачи были выявлено несколько существующих аналогов. Они схожи по функционалу, но предоставляемые ими возможности не до конца удовлетворяют требования к системе.

KeyCloak - это open-source сервер аутентификации и управления учетными записями, построенный на базе спецификаций OAuth 2.0, Open ID Connect, JSON Web Token и SAML 2.0. Разрабатывается JBoss.

Преимущества:

* Встроенный механизм управления учетными записями
* Поддержка большого количества платформ
* Поддержка каталогов LDAP
* Поддержка протоколов SAML 2 и OpenID Connect

Недостатки:

* Не поддерживает Kerberos
* Ограниченные возможности по расширению функционала
* Сложности интеграции с существующими базами данных

OpenidDict – это open-sour библиотека, позволяющая разворачивать сервер аутентификации. Поддерживает OAuth 2.0, Open ID Connect.

Преимущества:

* Легко расширяется и дополняется
* Поддержка механизма управления учётными записями

Недостатки:

* Поддерживает только OpenID Connect
* Поддерживает только платформу ASP.NET

## 1.5 Формулировка проблемы и актуальность

Проблематика заключается в необходимость ООО «БиАйВи» в агрегации различных протоколов аутентификации в одну систему. С предоставлением пользователям этих протоколов единого интерфейса аутентификации.

Актуальность состоит в потребности ООО «БиАйВи» в повышении гибкости системы аутентификации. А также повышенные безопасности путём внедрения современных протоколов аутентификации.

## 1.6 Цели и задачи

Цель – основной целью создания системы является упрощение процесса аутентификации для пользователей сервисов компании ООО «БиАйВи».

Для реализации поставленных целей система должна решать следующие задачи:

* Предоставление общего интерфейса для пользователей, использующих разные протоколы аутентификации
* Поддержка работы с протоколом Kerberos
* Поддержка работы с протоколом OpenID Connect
* Поддержка работы с LDAP аутентификацией
* Реализация технологии Single Sign-On

# Раздел 2

## 2.1 Функциональные и технологические требования

Система должна проводить аутентификацию в доверенных центрах KDC с протоколами Kerberos, OpenID Connect.

Система должна обеспечивать хранение полученных от разных центров аутентификации тикетов, соответственно, для каждого зарегистрированного пользователя.

Система должна обеспечивать технологию Single Sign-On, используя полученные тикеты.

Система должна предоставлять гибкую структуру базы данных, для сохранения возможности по расширению перечня используемых протоколов.

Система по полученным тикетам должна понимать какой протокол использовался для их генерации.

Система не должна передавать по сети учётные данные пользователей в открытом виде.

## 2.2 Обоснование выбора технологий и инструментов разработки

Были использованы такие технологии, как:

Java — объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения.

Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной java-машине (JVM) — программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор, но с тем отличием, что байт-код, в отличие от текста, обрабатывается значительно быстрее.

Язык Java потребовался для создания интерактивных продуктов для сети Internet. Фактически, большинство архитектурных решений, принятых при создании Java, было продиктовано желанием предоставить синтаксис, сходный с C и C++. В Java используются практически идентичные соглашения для объявления переменных, передачи параметров, операторов и для управления потоком выполнением кода. В Java добавлены все хорошие черты C++.

Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов

SSO (Single Sign-On - Технология единого входа) — технология, при использовании которой пользователь переходит из одного раздела портала в другой без повторной аутентификации.

Например, если на веб-портале существует несколько обширных независимых разделов (форум, чат, блог и т. д.) то, пройдя процедуру аутентификации в одном из сервисов, пользователь автоматически получает доступ ко всем остальным, что избавляет его от многократного ввода данных своей учётной записи.

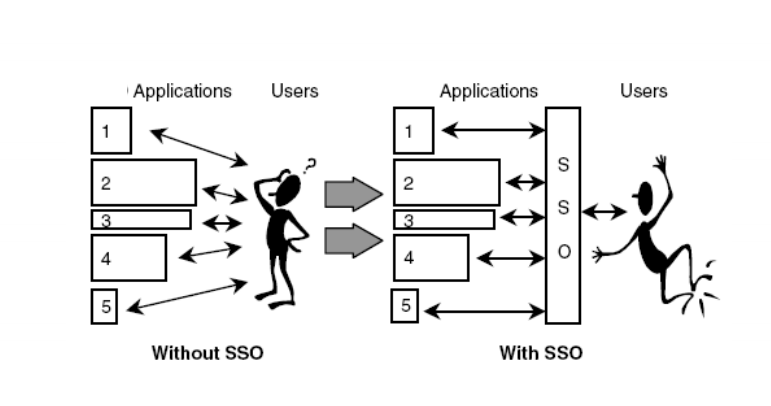


Рисунок 7. Принцип работы SSO

Таким образом, схема авторизации изменяется:

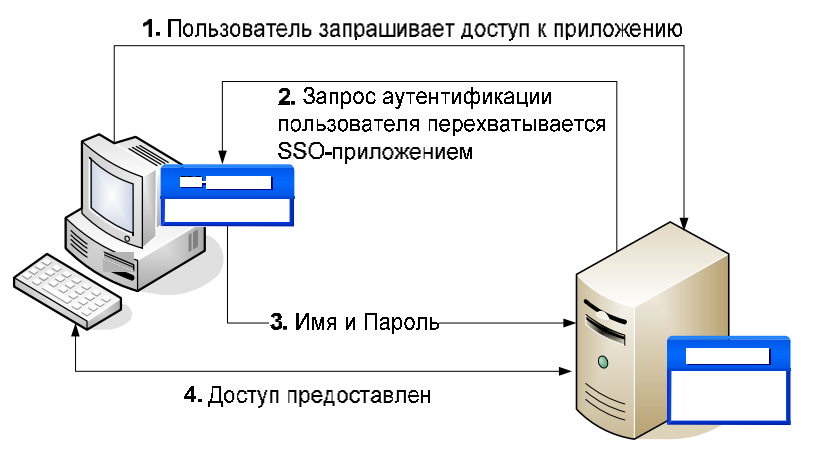


Рисунок 8. Схема авторизации без SSO

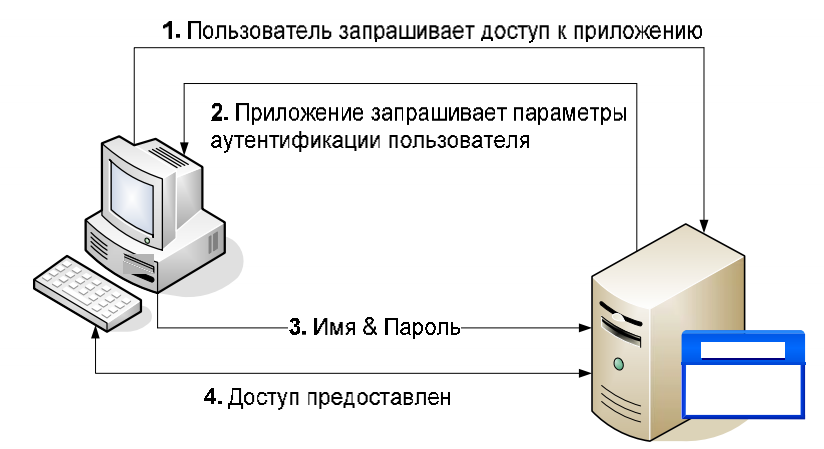


Рисунок 9. Схема авторизации с SSO

Таким образом, можно выделить преимущества технологии:

уменьшение парольного хаоса между различными комбинациями имени пользователя и пароля;

уменьшение времени на повторный ввод пароля для одной и той же учетной записи;

поддержка традиционных механизмов аутентификации, таких как имя пользователя и пароль;

снижение расходов на IT-службу за счёт уменьшения количества запросов по восстановлению забытых паролей;

обеспечение безопасности на каждом уровне входа/выхода/доступа к системе без причинения неудобств пользователям.

В технологии единого входа применяются централизованные серверы аутентификации, используемые другими приложениями и системами, которые обеспечивают ввод пользователем своих учётных данных только один раз.

LDAP (Lightweight Directory Access Protocol — «облегчённый протокол доступа к каталогам») — протокол прикладного уровня для доступа к службе каталогов X.500, разработанный IETF как облегчённый вариант разработанного ITU-T протокола DAP.

Технически, LDAP — это всего лишь протокол, определяющий методы, посредством которых осуществляется доступ к данным каталога. Он также определяет и описывает, как данные представлены в службе каталогов (Модель данных или Информационная модель). Наконец, он определяет, каким образом данные загружаются (импортируются) и выгружаются (экспортируются) из службы каталогов (с использованием LDIF). LDAP не определяет, как происходит хранение и манипулирование данными. С точки зрения стандарта хранилище данных и методы доступа к нему — это "чёрный ящик", за который, как правило, отвечают модули back-end (механизмы манипуляции данными) какой-либо конкретной реализации LDAP (обычно в них используется некоторая форма транзакционной базы данных).

LDAP определяет четыре модели. Информационная модель: Модель данных (или информационная модель) определяет, каким образом информация или данные представлены в системе LDAP. Это может совпадать или не совпадать с фактическим методом представления данных в хранилище на физическом носителе. Как упоминалось выше, вопрос хранилищ данных лежит за пределами стандартов LDAP.

Модель именования: определяет принципы наименования данных в иерархической структуре, наподобие 'dc=example,dc=com'.

Функциональная модель: чтение, поиск, запись или модификация данных в LDAP.

Модель безопасности: LDAP позволяет контролировать, причём весьма детально, кто, что и с какими именно данными может сделать. Это сложная, но мощная система. Эта модель также включает в себя защиту данных при передаче по сети, такую как TLS/SSL.

LDAP позволяет проводить аутентификацию напрямую, или использоваться как база данных.

Kerberos. Протокол Kerberos был создан более десяти лет назад в Массачусетском технологическом институте в рамках проекта Athena. Однако общедоступным этот протокол стал, начиная с версии 4. После того, как специалисты изучили новый протокол, авторы разработали и предложили очередную версию — Kerberos 5, которая была принята в качестве стандарта IETF. Требования реализации протокола изложены в документе RFC 1510, кроме того, в спецификации RFC 1964 описывается механизм и формат передачи жетонов безопасности в сообщениях Kerberos.

Протокол Kerberos предлагает механизм взаимной аутентификации клиента и сервера перед установлением связи между ними, причём в протоколе учтён тот факт, что начальный обмен информацией между клиентом и сервером происходит в незащищённой среде, а передаваемые пакеты могут быть перехвачены и модифицированы. Другими словами, протокол идеально подходит для применения в Интернет и аналогичных сетях.

Основная концепция протокола Kerberos — если есть секрет, известный только двоим, то любой из его хранителей может с лёгкостью удостовериться, что имеет дело со своим напарником. Для этого ему достаточно проверить, знает ли его собеседник общий секрет.

OpenID — открытый стандарт децентрализованной системы аутентификации, предоставляющей пользователю возможность создать единую учётную запись для аутентификации на множестве не связанных друг с другом интернет-ресурсов, используя услуги третьих лиц.

Базовой функцией OpenID является предоставление портативного, клиент-ориентированного, цифрового идентификатора для свободного и децентрализованного использования.

Стандарт описывает процесс коммуникации интернет-ресурсов (Relying Parties), требующих аутентификации, и провайдеров OpenID (OpenID Providers). Существует несколько OpenID-провайдеров, которые предоставляют хостинг OpenID URL. Аутентификацию OpenID используют в том числе Google, Yahoo!, LiveJournal, MySpace, IBM. Расширение стандарта (the OpenID Attribute Exchange) облегчает передачу пользовательских данных, таких как имя или пол, от OpenID-провайдера до интернет-ресурса.

На декабрь 2009 года существовало более 1 миллиарда аккаунтов OpenID и около 9 миллионов сайтов, поддерживающих технологию OpenID.

Текущая версия стандарта, OpenID Connect 1.0, вышла в феврале 2014 года и была обновлена в ноябре 2014 года.

В качестве инструментов разработки были выбраны:

IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

Первая версия появилась в январе 2001 года и быстро приобрела популярность как первая среда для Java с широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с версии 9.0, среда доступна в двух редакциях: Community Edition и Ultimate Edition. Community Edition является полностью свободной версией, доступной под лицензией Apache 2.0, в ней реализована полная поддержка Java SE, Kotlin, Groovy, Scala, а также интеграция с наиболее популярными системами управления версиями. В редакции Ultimate Edition, доступной под коммерческой лицензией, реализована поддержка Java EE, UML-диаграмм, подсчёт покрытия кода, а также поддержка других систем управления версиями, языков и фреймворков.

Git — распределённая система контроля версий, которая даёт возможность разработчикам отслеживать изменения в файлах и работать совместно с другими разработчиками.

Системы контроля версий (СКВ, VCS, Version Control Systems) позволяют разработчикам сохранять все изменения, внесённые в код. Поэтому они могут просто откатить код до рабочего состояния вместо того, чтобы тратить часы на поиски маленькой ошибки или ошибок, ломающих весь код. СКВ также дают возможность нескольким разработчикам работать над одним проектом и сохранять внесённые изменения, чтобы убедиться, что все могут следить за тем, над чем они работают

## 2.3 Техническое задание на разработку и проектирование программного обеспечения

Система должна представлять из себя серверное приложение. Поддерживать работу с центрами аутентификации основанными на протоколах Kerberos, OpenID Connect, LDAP.

В систему должны входить:

1. Подсистема хранения данных
2. Подсистема обработки запросов
3. Подсистема работы с серверами аутентификации

Система должна сохранять работоспособность и обеспечивать восстановление своих функций при сбоях в системе электроснабжения аппаратной части, приводящих к перезагрузке сервера, восстановление программы должно происходить после перезапуска ОС сервера и повторного запуска

Система должна обеспечивать недопущение передачи данных пользователей в открытом виде и минимизировать передачу таких данных вовсе.

Система должна разворачиваться на сервере под управление операционной системы Linux.

# Раздел 3

## 3.1 Описание методики тестирования

При испытании модуля будут применены две методики тестирования:

1. Модульное тестирование, или юнит-тестирование — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.  
   Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.
2. Системное тестирование – это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям, как функциональным, так и нефункциональным.

Как правило выделяют 2 подхода к системному тестированию:

1. На базе требований. Тестирование проводится в соответствии с функциональными или нефункциональными требованиями, для каждого из которых пишется testcase (тестовые прецеденты).
2. На базе случаев использования. Тестирование происходит в соответствии с вариантами использования продукта, на основе которых создаются usercases (пользовательские прецеденты). Для каждого из данных пользовательских прецедентов создаются персональные тестовые прецеденты.

## 3.2 Набора тестов,программа и методикаиспытаний

Таблица 1 - Тестирование функциональных требований. Системный способ.

|  |  |
| --- | --- |
| Тестирование функциональных требований | |
| Функциональность | Методика тестирования |
| Запрос к тестовому сервису уже аутентифицированного в системе пользователя | 1. Открыть страницу сервиса в браузере   Результат: предоставление доступа к ресурсу |
| Запрос к тестовому сервису ещё не аутентифицированного в системе пользователя, но аутентифицированного в доверенном KDC | 1. Открыть страницу сервиса в браузере 2. Выбрать метод входа Kerberos   Результат: предоставление доступа к ресурсу |
| Запрос к тестовому сервису не аутентифицированного в системе пользователя, и не аутентифицированного в доверенном KDC | 1. Открыть страницу сервиса в браузере 2. Выбрать метод входа Kerberos 3. В появившейся форме ввести учётные данные тестового пользователя.   Результат: предоставление доступа к ресурсу |

Таблица 2 - Тестирование функциональных требований. Модульный способ.

|  |  |  |
| --- | --- | --- |
| Тестирование функциональных требований | | |
| Функциональность | Модульный способ тестирования | |
| Входные данные | Ожидаемый результат |
| Выполнение запроса аутентификации к KDC Kerberos | Идентификатор пользователя, идентификатор запрашиваемого сервиса | Отправка сообщения KRB\_AS\_REQ с данными пользователя |
| Проверка подписи корректного токена JWT | Токен JWT, Открытый ключ IdP | Токен действителен |

# Раздел 4

## 4.1 Описание алгоритма

Для реализации приложения был разработан алгоритм выполнения входа.

1. Пользователь запрашивает сервис и передаёт ему свой токен.
2. Сервис обращается к системе централизованной ау
3. На этом шаге приложение начинает воспроизводит задание на экране, но требует подтверждения. Это специальная метка, по которой будет отображена 3D – модель в видеопотоке.
4. Приложение сканирует окружающее пространство, через камеру устройства, в поисках метки, на которую в видеопотоке будет крепится 3D – модель макета ветряной мельницы.
5. Если метка обнаружена, то модель прикрепляется к ней и становится не подвижной. Это дает пользователю возможность обойти вокруг реального макета и посмотреть на него со всех сторон через приложение, так как можно увидеть модель в разрезе.
6. После предыдущего шага появляются вопросы с заданием на которые необходимо ответить или предоставляется возможность управления реальным макетом. Для это создается отдельный процесс на общение с сервером по API (для отправки команд на CPS - сервер).
7. После окончания работ формируется отсчет с ошибками студента и отправляется преподавателю в личный кабинет на сервер.

## 4.2 Результат тестирования

Испытания проводились в соответствии с планом и методикой тестирования, описанными в разделе 2.4. В результате выполнения испытаний методом ручного тестирования были проверены требования, результаты проверки которых приведены в таблице:

Таблица 3 – Результаты тестирования функциональных требований. Системный способ.

|  |  |
| --- | --- |
| Тестирование функциональных требований | |
| Функциональность | Результат |
| Запрос к тестовому сервису уже аутентифицированного в системе пользователя | Корректно |
| Запрос к тестовому сервису ещё не аутентифицированного в системе пользователя, но аутентифицированного в доверенном KDC | Корректно |
| Запрос к тестовому сервису не аутентифицированного в системе пользователя, и не аутентифицированного в доверенном KDC | Корректно |

Таблица 2 – Результаты тестирования функциональных требований. Модульный способ.

|  |  |  |
| --- | --- | --- |
| Тестирование функциональных требований | | |
| Функциональность | Модульный способ тестирования | |
| Ожидаемый результат | Фактический результат |
| Выполнение запроса аутентификации к KDC Kerberos | Отправка сообщения KRB\_AS\_REQ с данными пользователя | Отправка сообщения KRB\_AS\_REQ с данными пользователя |
| Проверка подписи корректного токена JWT | Токен действителен | Токен действителен |

# Заключение

В ходе выполнения практики был проведен обзор предметной области современных методов аутентификации, также проведён анализ имеющихся аналогов. Кроме того, были изучены некоторые из основных протоколов аутентификации и авторизации. И связанные с ними технологии. Получен опыт работы в IT-компании.

# Список используемой литературы

1. Хорев П.Б. Методы и средства защиты информации в компьютерных системах: Учеб. пособие для студ. высш. учеб. заведений -- М.: Издательский центр «Академия», 2005. -- 256 с.
2. Адаменко, Михаил Основы классической криптологии. Секреты шифров и кодов / Михаил Адаменко. - Москва: Машиностроение, 2014. - 256 c.
3. Бабаш, А. В. История криптографии. Часть I / А.В. Бабаш, Г.П. Шанкин. - М.: Гелиос АРВ, 2002. - 240 c.
4. Гупта, Арун Java EE 7. Основы / Арун Гупта. - М.: Вильямс, 2014. - 336 c
5. RFC 4120 – Стандарт службы сетевой аутентификации Kerberos (версия 5)
6. RFC 6113 - Обобщенный механизм предварительной аутентификации для Kerberos.
7. Инфраструктуры открытых ключей / Полянская О.Ю., Горбатов В.С. - M.: Интернет-университет информационных технологий - ИНТУИТ.ру, 2007
8. Обзор протокола Kerberos – режим доступа:  
   <https://pro-ldap.ru/tr/zytrax/tech/kerberos.html>