Санкт-Петербургский государственный университет

Кафедра Системного программирования

Танков Владислав Дмитриевич

Реализация единой точки аутентификации и регистрации на основе Spring

Курсовая работа

Научный руководитель:
к. т. н., старший преподаватель Брыксин Т. А.

Санкт-Петербург
2016

Оглавление

Введение

В последние годы в IT-компаниях усилилось движение по направлению ко взаимной интеграции корпоративных сервисов. Такие образующиеся группы сервисов принято называть экосистемами. Объединение группы программных продуктов в экосистему упрощает маркетинг сервисов, уменьшает отток клиентов и позволяет более гибко управлять процессом разработки в компании.

В процессе объединения группы веб-сервисов возникает задача объединения пользовательских баз и создания единой точки аутентификации и регистрации [1]. Если первая задача довольно специфична для каждой компании, то вторая почти не зависит от конкретной компании и её веб-сервисов. Тем не менее, как утилиты для объединения пользовательских баз (что вполне разумно) существуют лишь в недрах компаний, так и инструменты для быстрого создания единых точек аутентификации и регистрации очень редко доступны opensource (что совершенно непонятно).

Создание единой точки аутентификации и регистрации не кажется сложной задачей, однако в действительности требует предельной аккуратности при исполнении. Любая ошибка при проектировании или реализации может обернуться серьёзной уязвимостью в программе. Поэтому абсолютное большинство компаний используют известные и протестированные протоколы единой аутентификации, лишь реализуя их собственными силами.

В рамках проекта QReal:Web также существует несколько веб-сервисов, объединение которых в единую экосистему позволило бы пользователю не выходить из привычного окружения во время работы с конструктором TRIK. Было решено разработать инструментарий для быстрого проектирования единой точки аутентификации и регистрации.

[1] Далее в некоторых случаях будем сокращать полное название до просто "точка"

Постановка задачи

Целью моей курсовой работы является разработка безопасной и легко расширяемой единой точки аутентификации и регистрации для группы сервисов QReal:Web.

Для достижения данной задачи были поставлены следующие цели:

1. Провести исследование существующих подходов к организации единых точек аутентификации и регистрации
2. Реализовать единую точку аутентификации и регистрации для группы сервисов QReal:Web
3. Провести обзор безопасности полученного решения

1. Обзор существующих решений

PubCookie

Данный проект предоставляет отдельный сервер — единую точку аутентификации и регистрации и плагины для интеграции веб-серверов (например, Apache и IIS) с ней.

К сожалению, PubCookie не поддерживается с 2010 года. Более того, он не интегрируется с контейнерами сервлетов, что затрудняет полное тестирование сервисов в процессе разработки.

Jasig CAS

Данный проект предоставляет отдельный сервер — единую точку аутентификации и различные способы интеграции, включая поддержку из Spring посредством классов Spring Security.

CAS - это чрезвычайно мощное решение, позволяющее сконфигурировать точку аутентификации, удовлетворяющую любым возможным требованиям. Тем не менее, и здесь есть недостатки.

Дело в том, что CAS изначально спроектирован как точка именно аутентификации, а не аутентификации и регистрации. В результате его способности по оперированию группами пользователей совершенно недостаточны для нормальной работы. Расширение же данного проекта своими силами (его исходные коды доступны свободно) представляет собой нетривиальную задачу, сравнимую по сложности с полностью самостоятельной реализацией единой точки аутентификации и регистрации.

Уже при проектировании было выдвинуто требования возможности расширения системы до точки авторизации доступа к пользовательским ресурсам. CAS также не поддерживал бы эту возможность.

Итог

Так как ни одно из существующих решений не удовлетворяет задачам данной работы и требованиям, предъявляемым к создаваемой системе, было решено реализовать собственную точку единой аутентификации и регистрации.

2. Обзор существующих протоколов единой аутентификации

OpenID

Данный протокол изначально проектировался для единой аутентификации на множестве веб-сервисов.

К сожалению, первая версия OpenID не смогла завоевать всеобщую популярность, и лишь современный стандарт — OpenID Connect — начинает поддерживаться крупными игроками, и то далеко не всеми.

OAuth 2.0

В свою очередь, протокол OAuth изначально разрабатывался как протокол авторизации доступа к ресурсам, однако чрезвычайно популярен и как протокол единой аутентификации.

Многие исследователи отмечают некоторую двусмысленность направленности данного протокола [2], однако в условиях его широчайшей распространённости не использовать OAuth - значит лишиться возможности интеграции с большинством социальных сетей и других провайдеров аутентификации.

Итог

В качестве протокола единой аутентификации был выбран протокол OAuth 2.0. Как уже было отмечено, он поддерживается абсолютным большинством внешних поставщиков аутентификации и на данный момент де-факто является стандартом протоколов единой аутентификации.

[2] Статья о проблемах OAuth: `http://www.thread-safe.com/2012/01/problem-with-oauth-for-authentication.html`

3. Обзор существующих инструментов

Spring Security OAuth

Данный проект предоставляет библиотеку, реализующую все основные возможности OAuth и OAuth 2.0, описанные в стандарте. Библиотека обладает понятной документацией и регулярно обновляется сообществом разработчиков Spring. Приятным дополнением является удобная и простая интеграция со всеми основным службами Spring framework и системой безопасности Spring Security.

Важно отметить, что использование данной библиотеки без использования в проекте Spring Security довольно затруднительно, так как она фактически является расширением Spring Security, а не самостоятельным проектом.

Apache Oltu

Данный проект предоставляет библиотеку, реализующую основные возможности OAuth 2.0 и OpenID. Библиотека обладает обширной документацией и регулярно обновляется разработчиками Apache Foundation. В отличие от Spring Security OAuth, она не зависит от других проектов и предоставляет несколько более низкоуровневый доступ к управлению протоколом OAuth.

Итог

Так как на данный момент все веб-сервисы проекта QReal:Web используют Spring и в проекте есть несколько разработчиков, близко знакомых с данным фреймворком, для реализации серверной части была выбрана связка Spring + Spring Security с библиотекой Spring Security OAuth.

На клиентской стороне на данный момент также используется Spring Security OAuth, однако в случае миграции сервисов на другой фреймворк он будет заменён на Apache Oltu.

4. Реализация

4.1. Требования

Так как система изначально имеет прикладную направленность и будет использоваться простыми пользователями QReal:Web, были поставлены некоторые не функциональные, но обязательные для успешного внедрения системы требования.

На стадии реализации задачи данной курсовой работы так же были переформулированы в требования.

Требования к системе таковы:

1. Должен реализовываться один из механизмов единой аутентификации и регистрации
2. Должно быть возможно расширение системы до точки авторизации доступа к ресурсам пользователя
3. Система должна иметь основные возможности по управлению группами пользователей и работе с клиентами единой точки аутентификации и регистрации
4. Должна поддерживаться интеграция со внешними поставщиками аутентификации (к примеру, социальными сетями)
5. Система должна обеспечивать надлежащий уровень безопасности

4.2. Реализация протокола единой аутентификации

Для аутентификации используется Authorization Code Grant поток OAuth 2.0. Его использование даёт некоторые преимущества, такие как дополнительные гарантии безопасности пользователю, увеличивающееся время доступности ресурсов пользователя клиенту без переавторизации, возможность отзыва авториазации в любой момент [1].

(a) Пользователь желает предоставить авторизацию клиенту

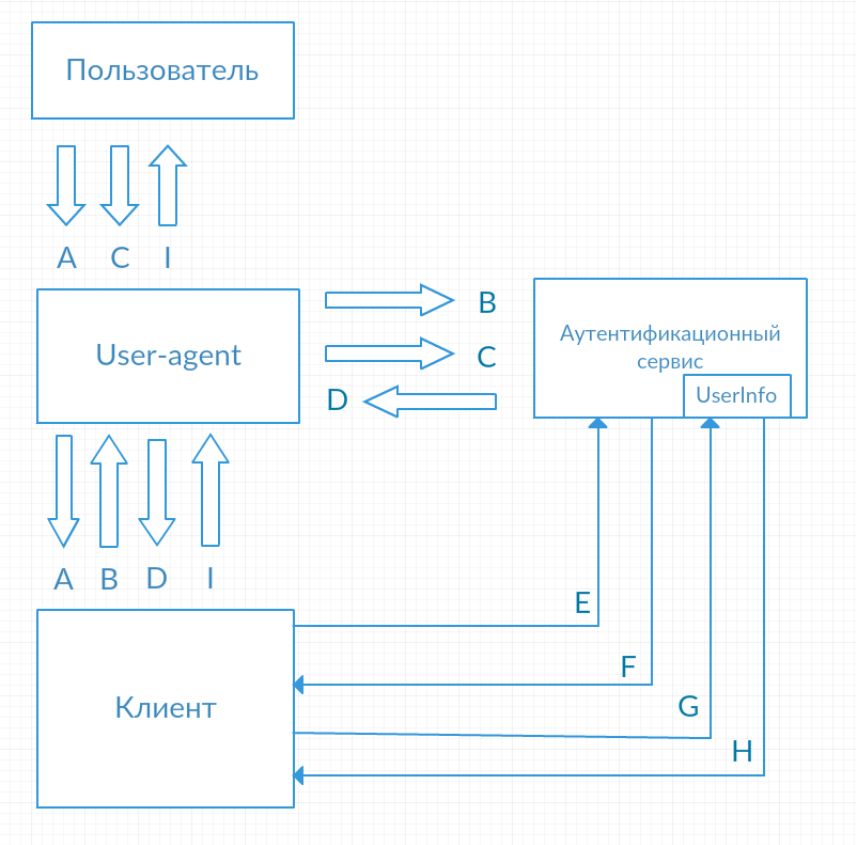

Рис. 1: Authorization code grant поток

(b) Клиент перенаправляет пользователя на аутентификационный сервис. В запросе передаются адрес обратного перенаправления и дополнительные сведения о клиенте

(c) Пользователь аутентифицируется на аутентификационном сервисе и авторизует клиент на доступ к некоторым из его ресурсов

(d) Аутентификационный сервис перенаправляет пользователя обратно на клиент по адресу обратного перенаправления. В запросе передаётся авторизационный код

(e) Клиент перехватывает пользователя на URI обратного перенаправления и забирает из запроса авторизационный код. После этого клиент направляет новый запрос на аутентификационный сервис с целью обменять авторизационный код на токен доступа и токен обновления

(f) В случае если авторизационный код верный, аутентификацион-

ный сервис возвращает клиенту токен доступа, токен обновления и некоторые дополнительные параметры

Это стандартный поток, его развёрнутое описание можно найти в RFC 6749[2].

Представляет интерес модификация данного потока для механизма единой аутентификации. После получения токена доступа (f) клиент с его помощью запрашивает у аутентификационного сервиса информацию о пользователе(g) (фактически получает доступ к ресурсу). Получив эту информацию (h) (как правило общий для всех сервисов id пользователя), клиент считает это доказательством верной аутентификации пользователя на точке единой аутентификации, загружает окружение пользователя с полученным с точки id и в свою очередь разрешает пользователю доступ к приложению (i).

Данная схема не лишена недостатков, однако наиболее распространена и исследована. Вопросы обеспечения безопасности при таком методе аутентификации будут рассмотрены в соответствующей главе.

Указанный протокол единой аутентификации реализуется раздельно - часть на единой точке аутентификации (серверная часть), часть на клиенте (клиентская часть).

Для реализации на сервере используется OAuth Spring Security с настройками сервера авторизации ресурсов. В качестве ресурса, авторизуемого пользователем, выступает точка доступа UserInfo. В ответ на авторизованный пользователем запрос (g) клиента данная точка возвращает ID пользователя и его права (т.н. роли) на сервисе авторизации (h).

Для реализации на клиенте используется OAuth Spring Security с переопределённым фильтром обработки неавторизованного доступа[3]. При попытке входа(a) пользователь перенаправляется на единую точку аутентификации и регистрации (b), в запрос добавляются сведения о клиенте - его id и секрет. На точке единой аутентификации и регистрации пользователь аутентифицируется и разрешает клиенту доступ

[3]Используется свободно доступная конфигурация (MIT license): `https://github.com/pwheel/spring-security-oauth2-client`

к своим данным(c). Далее он перенаправляется обратно на сервисный URI Spring Security OAuth (d). Здесь библиотека забирает из запроса авторизационный код, получает токен доступа и токен обновления с аутентификационного сервиса (e, f), после чего производит запрос к UserInfo точки аутентификации с использованием полученного токена доступа (g, h). В случае удачного запроса пользователь перенаправляется с сервисного URI на изначально запрошенную им страницу, ему передаются cookie с параметрами сессии, созданной для пользователя с ID полученным из UserInfo (i). В случае неудачного запроса пользователь перенаправляется на страницу ошибки (i).

Реализация позволяет легко расширить точку аутентификации и регистрации и передать ей дополнительные обязанности точки авторизации доступа к ресурсам пользователя. Использование точки доступа UserInfo - это уже фактически использование ресурсов пользователя, к которым был разрешён доступ. Добавление новых точек доступа к ресурсам в таком случае является тривиальной задачей.

4.3. Реализация веб-приложения единой точки аутентификации и регистрации

Для построения веб-интерфейса управления единой точкой аутентификации и регистрации используется SpringMVC.

Реализована панель управления правами пользователей, с помощью которой можно дать пользователю привилегии администратора или же их отнять их. На данный момент иные роли для пользователей веб-сервисов QReal:Web не требуются, однако при необходимости новые роли легко могут быть добавлены.

Реализована панель управления клиентами. С помощью неё можно создать нового клиента, выбрать для него id, секрет и права на доступ к ресурсам (к примеру, read и write), которые он будет запрашивать у пользователя.

Хранение записей пользователей и клиентов осуществляется в базе SQLite, доступ к ней производится через JDBC, а сериализацией-

десериализацией объектов занимается Hibernate. За счёт использования JDBC возможна смена базы данных на более производительный вариант, такой как MySQL или PostgreSQL. Конфигурация с SQLite используется лишь на время разработки.

Говоря о базе данных, используемой в системе, стоит отметить, что выпущенные токены доступа и обновления, а также выданные авторизационные коды хранятся в оперативной памяти. В абсолютном большинстве случаев они не являются критически важной информацией и могут быть перевыпущены после перезагрузки сервера.

4.4. Реализация интеграции со внешними поставщиками аутентификации

Для реализации аутентификации и регистрации через внешних поставщиков аутентификации используется Spring Security OAuth, однако в данном случае точка единой аутентификации и регистрации предстаёт не как сервер, а как клиент ко внешнему аутентификационному сервису.

В данном случае механизм единой аутентификации на основе протокола OAuth 2.0, описанный ранее, просто используется два раза. Клиент перенаправляет пользователя для прохождения аутентификации на единую точку аутентификации и регистрации, пользователь выбирает аутентификацию с помощью внешнего поставщика аутентификации и протокол единой аутентификации запускается снова, но уже между точкой и внешним поставщиком аутентификации. В результате точка получает информацию о пользователе от внешнего поставщика аутентификации (к примеру, google.com). После этого она либо аутентифицирует данного пользователя, в случае если запись пользователя с id, полученным от внешнего поставщика аутентификации, уже существует, либо создаёт новую запись пользователя. Далее точка расценивает получение данных со внешнего поставщика аутентификации как удачную аутентификации пользователя и продолжает исполнение протокола единой аутентификации между точкой и клиентом.

Для реализации данного механизма на единой точке аутентификации и регистрации используются почти те же приёмы, что и для реализации аутентификации через единый механизм аутентификации на клиентах. Однако, имеются некоторые тонкости.

Так как на точке уже используется контекст безопасности, реализующий стандартную систему аутентификации и регистрации Spring Security, приходится выделять отдельные URI, не входящие в общий контекст безопасности, а имеющие собственный контекст, закрытый переопределённым фильтром обработки неавторизованного доступа[4]. Для каждого URI создаётся свой фильтр с соответствующими параметрами доступа ко внешнему поставщику аутентификации (такими как - id приложения, секрет, требуемые права и т.д.).

Таким образом, для интеграции со внешними поставщиками аутентификации - Google и Github требуется создать два не закрытых общим контекстом безопасности URI (/oauth/google, /oauth/github) и для каждого из них создать свой контекст безопасности с собственным фильтром. Несмотря на кажущуюся сложность данной задачи, процесс этот довольно механический и не представляет трудности. Возможно реализовать изменение параметров внешних поставщиков аутентификации и добавление новых поставщиков без внесения изменений в код, однако, как правило, список внешних поставщиков аутентификации определяется при старте веб-сервисов и мало меняется со временем, так что от данной возможности пока было решено отказаться.

[4]Используется та же свободно доступная конфигурация (MIT license): `https://github.com/pwheel/spring-security-oauth2-client`

5. Безопасность полученного решения

Важной характеристикой всякого решения единой аутентификации является его безопасность. Как уже упоминалось во введении, при проектировании собственного протокола единой аутентификации легко допустить ошибку, поэтому использовался стандартный протокол OAuth 2.0 и его распространённая модификация для проведения аутентификации [3].

Важно отметить, что, так как сервис проектируется для использования в рамках единой экосистемы, используемый протокол несколько упрощён. Как упоминается в [1] использование дополнительного токена ID Token обусловлено возможностью кражи токенов доступа одним из клиентов. Защита токенов на всём пути от единой точки аутентификации и регистрации до клиента осуществляется посредством TLS. Так как все клиенты расположены в рамках единой экосистемы, то шанс кражи одним из клиентов токенов чрезвычайно мал.

Более того, так как используется Authorization Code Grant поток OAuth 2.0 со случайно генерируемыми state (что защищает от атаки повторением), и токен доступа клиент получает при обращении напрямую к единой точке аутентификации и регистрации, подмена украденного токена доступа возможна лишь на путь от клиента к точке, а данный путь защищён TLS. Данный довод также приводится в [1]. Таким образом, при наличии сети доверенных клиентов и реализованным в соответствии со спецификацией Authorization Code Grant потоке OAuth 2.0 используемый механизм аутентификации может быть существенно упрощён.

Кроме общих требований к безопасности структуры системы существуют также требования, выдвигаемые [2] к материальной базе системы. Без исполнения данных требований система не может считаться безопасной.

Первым требованием является обязательное и повсеместное использование TLS. Использование данного протокола шифрования поможет предотвратить атаки man-in-the-middle. В случае отсутствия TLS воз-

можна кража токенов доступа и внедрение в канал клиент - единая точка аутентификации и регистрации украденных токенов. В таком случае злоумышленник без особых сложностей сможет войти в любой сервис компании под именем пользователя, у которого были украдены токены.

Вторым требованием является обязательное соответствие всех используемых клиентами и единой точкой аутентификации и регистрации DNS-серверов спецификации DNSSEC. В ином случае возможна атака на клиента подменой DNS-записей и соответственно единой точки аутентификации и регистрации. Описание атаки DNS spoofing можно найти в [4]

Если все клиенты являются доверенными и выполнены все вышеперечисленные требования, то данную систему можно считать вполне надёжной для целей проекта QReal:Web.

Заключение

В рамках работы был проведён всесторонний обзор существующих инструментов, протоколов и готовых решений. В итоге были обоснованно отвергнуты существующие готовые решения, выбран подходящий стандарт протокола аутентификации и необходимые инструменты.

Была реализована система, удовлетворяющая задаче, поставленной в работе, а также всем не функциональным требованиям, предъявляемым к её работе со стороны проекта QReal:Web. Проект системы был обоснован существующими стандартами данной предметной области. Система была интегрирована с системой QReal:Web - Robots diagrams и показала свою работоспособность.

Был проведён обзор безопасности данной системы и сформулированы основные требования к условиям её работы. Данные требования обоснованы существующими стандартами и ссылками на литературу предметной области. В случае исполнения всех требований ожидается достаточная для проекта QReal:Web безопасность системы.

Дальнейшие перспективы

На данный момент проект QReal:Web планируется перепроектировать в новом архитектурном стиле - микросервисном. В рамках продолжения данной работы будет создан клиентский плагин к единой точке аутентификации и регистрации на Apache Oltu.

Также есть планы добавить OpenID Connect как дополнительный поддерживаемый механизм аутентификации. Данный механизм лишён некоторых недостатков используемого OAuth 2.0 и набирает популярность среди внешних поставщиков аутентификации.

Список литературы

[1] Boyd Ryan. Getting Started with OAuth 2.0. — O'Reilly Media, 2012. — ISBN: 978-1-4493-1160-5.

[2] IETF. The OAuth 2.0 Authorization Framework. — 2012. — URL: `https://tools.ietf.org/html/rfc6749` (online; accessed: 25.04.2016).

[3] OpenID. OpenID Connect Basic. — 2015. — URL: `http://openid.net/specs/openid-connect-basic-1_0.html` (online; accessed: 26.04.2016).

[4] Таненбаум Эндрю Уэзеролл Дэвид. Компьютерные сети. — Питер, 2013. — ISBN: 978-5-4461-0068-2.

[2] [1] [3] [4]