Реализация единой точки аутентификации и регистрации на основе Spring

Танков В. Д., математико-механический факультет СПбГУ, 244 группа, vdtankov@gmail.com

Аннотация

При объединении нескольких веб-сервисов в единую экосистему компании часто сталкиваются с необходимостью создания механизмов единой аутентификации пользователей во всей группе веб-сервисов. Несмотря на то, что задача эта чрезвычайно распространённая, большая часть компаний решаёт её самостоятельно и не делится наработками с сообществом. Данная работа описывает процесс построения единой точки аутентификации и регистрации в проекте QReal:Web от выбора механизма единой аутентификации до реализации серверной логики.

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 работу пользователя с сервисами и позволяет более гибко управлять процессом разработки веб-сервисов в компании, за счёт переиспользования сервисов аутентификации, оплаты и т.д. [1].

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 мало инструментов для быстрого создания единых точек аутентификации и регистрации доступны opensource.

Проект QReal:Web[2] в течение нескольких лет разрабатывается на математико-механическом факультете СПбГУ. В его состав входят сервис Robots diagram, позволяющий создавать диаграммы для роботов

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

[2] Проект QReal:Web на Github: https://github.com/qreal/qreal-web

TRIK, и сервис Robots Store, позволяющий закачивать программы, сгенерированные по диаграммам, прямиком на робота TRIK. Объединение двух данных веб-сервисов в единую экосистему избавило бы пользователя от необходимости аутентифицироваться дважды при выполнении стандартного сценария работы — создания диаграммы и закачки её на робота. Чтобы такое объединение стало возможным, необходимо было разработать единую точку аутентификации и регистрации.

Выбор протокола единой аутентификации

В качестве механизма аутентификации было решено использовать OAuth 2.0 [3] и его расширение OpenID Connect 1.0 [4] (с некоторыми упрощениями).

OAuth 2.0 и его расширение OpenID Connect 1.0 были выбраны как наиболее популярный и изученный механизм единой аутентификации (в отличие, например, от OpenID который до недавнего времени был мало распространён). Дополнительным плюсом была возможность интеграции со внешними сервисами, предоставляющими аутентификацию, без смены механизма аутентификации.

Стандарт OAuth 2.0 описывает несколько потоков (видов) авторизации. Был выбран Authorization Code Grant поток, кратко описанный ниже. Развёрнутое описание данного потока можно найти в RFC 6749 [3].

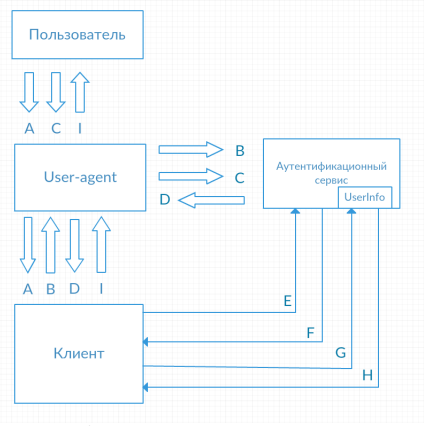

Рис. 1: Authorization code grant поток

(a) Пользователь желает предоставить авторизацию клиенту.

(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, аутентификационный сервис возвращает клиенту токен доступа, токен обновления и некоторые дополнительные параметры.

По сравнению с другими потоками, Authorization Code Grant даёт некоторые преимущества, такие как дополнительные гарантии безопасности пользователю, увеличивающееся время доступности ресурсов пользователя для клиента без переавторизации, возможность отзыва авторизации в любой момент [2]. Также важно отметить, что его использование позволяет отказаться от некоторых дополнительных мер безопасности в OpenID Connect 1.0.

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 [3], однако именно она наиболее

[3] О некоторых из которых можно прочесть здесь: http://www.thread-safe.com/2012/01/problem-with-oauth-for-authentication.html

исследована и стандартизована [4]. Вопросы обеспечения безопасности при таком методе аутентификации будут рассмотрены далее.

Реализация протокола единой аутентификации

Протокол единой аутентификации реализовывался раздельно — часть на единой точке аутентификации (серверная часть), часть на клиенте (клиентская часть).

В качестве фреймворка реализации был выбран Spring[4]. Он обладает обширной библиотекой, мощной моделью безопасности и готовыми библиотеками по работе с OAuth (Spring Security OAuth[5]). К тому же в проекте QReal:Web уже были разработчики, знакомые со Spring.

В случае, если использование Spring на клиентской части по каким-либо причинам невозможно, его библиотеки по работе с OAuth могут быть заменены библиотекой Apache Oltu[6].

На сервере используется Spring Security OAuth с настройками сервера авторизации ресурсов. В качестве ресурса, авторизуемого пользователем, выступает HTTP endpoint, которую мы будем называть UserInfo. В ответ на авторизованный пользователем запрос клиента (g), UserInfo возвращает ID пользователя и его права (т.н. роли) на сервисе авторизации (h).

На клиенте используется Spring Security с переопределённым фильтром обработки неавторизованного доступа [7]. 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 к своим данным (c). Далее он перенаправляется обратно на сервисный URI Spring Security (d). Здесь клиент извлекает из запроса авторизационный код, после чего клиент получает токен доступа и токен обновления с аутентификационного сервиса (e, f) и 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 па-

[4]Фреймворк Spring: https://spring.io/

[5]Библиотека Spring Security OAuth: http://projects.spring.io/spring-security-oauth/

[6]Библиотека Apache Oltu: https://oltu.apache.org/

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

раметрами сессии, созданной для пользователя с ID полученным из UserInfo (i). В случае неудачного запроса пользователь перенаправляется на страницу ошибки (i).

Безопасность полученного решения

Важной характеристикой всякого решения единой аутентификации является его безопасность.

При развёртывании единой точки аутентификации и регистрации должны соблюдаться все требования стандарта OAuth 2.0, в частности: использование только TLS (Transport Layer Security — протокол защищённой передачи данных, надстройка над TCP) для взаимодействий с точкой, соответствие всех используемых DNS-серверов спецификации DNSSEC (данная спецификация описывает дополнительные меры по защите DNS-серверов). Несмотря на то, что DNSSEC не требуется стандартом OAuth 2.0, без выполнения этого требования возможна атака DNS spoofing, описанная в [5].

Важно отметить, что используемый протокол несколько упрощён. 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 пути от клиента к точке, а данный путь защищён TLS. Данный довод также приводится в [2].

Таким образом, если выполнены все вышеперечисленные требования, то данную систему можно считать вполне надёжной для целей проекта QReal:Web.

Заключение

В данной статье были рассмотрены основные моменты проектирования и реализации единых точек аутентификации и регистрации на примере такой точки, созданной для проекта QReal:Web.

Литература

[1] Alistair Barros, Marlon Dumas and Peter Bruza. The Move to Web Service Ecosystems – BPTrends, Ноябрь 2005. – `http://www.`

bptrends.com/the-move-to-web-service-ecosystems/

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

[3] IETF. The OAuth 2.0 Authorization Framework. — 2012. — https://tools.ietf.org/html/rfc6749

[4] OpenID. OpenID Connect Basic. — 2015. — http://openid.net/specs/openid-connect-basic-1_0.html

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