ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
2017 Управление, вычислительная техника и информатика № 41

УДК 004.75
DOI: 10.17223/19988605/41/7

М.К. Павликов

МЕТОД АВТОРИЗАЦИИ КЛИЕНТСКИХ ПРИЛОЖЕНИЙ В ГЕТЕРОГЕННОЙ РАСПРЕДЕЛЕННОЙ ПРОГРАММНОЙ СИСТЕМЕ, ПОСТРОЕННОЙ НА ОСНОВЕ ПРОТОКОЛА HDP

Предложен метод авторизации приложений в распределенной программной системе, построенной на основе протокола HDP. Данный метод позволил получить централизованное управление авторизацией для программной системы, построенной из микросервисов. Использование метода позволило повысить безопасность защищенных данных, приобрести толерантность системы к изменению механизма авторизации и снизить временные затраты на разработку микросервисов. Даны описание метода и его сравнение с другими подходами к авторизации клиентских приложений в распределенной программной системе.
Ключевые слова: HDP; распределенная программная система; авторизация в программной системе; OAuth.

Авторизация – это предоставление прав на выполнение определенной операции и проверка таких прав при выполнении операции. В программных системах авторизация необходима для доступа к защищенной информации или методам получения, агрегации защищенной информации. В случае монолитного приложения трудностей в реализации авторизации обычно не возникает, так как существует достаточно много готовых решений данной задачи. В различных языках программирования и фреймворках накопилось достаточно много библиотек, которые расширяют языковые возможности по авторизации клиента.

В случае с высоконагруженными проектами, часто построенными по микросервисной архитектуре, одно приложение разносится по многочисленным узлам программной системы, причем процедуру авторизации необходимо получать к большинству таких сервисов. Реализация стандартных решений по авторизации приводит к катастрофическому дублированию кода, реализующего однотипную логику. В случае потребности внесения изменений в данную логику этот процесс невозможно реализовать мгновенно, так как в один момент времени обновить все микросервисы без выключения системы из эксплуатации невозможно. Также возникают вопросы отзыва прав у пользователя, которые должны вступить в силу на всех микросервисах в данной системе, и возникают проблемы назначения определенному пользователю прав на несколько операций, которые выполняют различные микросервисы.

1. Протокол авторизации OAuth и особенности его применения

На сегодняшний день самый популярный протокол авторизации между двумя приложениями – это OAuth. Данный протокол позволяет приложению получить авторизацию у микросервиса, используя HTTP запросы. Основное преимущество подхода состоит в том, что клиент не передает доверенному приложению логин и пароль от своего аккаунта, а лишь назначает определенные права, которые отражаются в получаемом токене.

Как видно из рис. 1, OAuth использует взаимодействие через браузер пользователя и прямое взаимодействие между программными системами, чтобы выполнить полный процесс авторизации. Это приводит к тому, что необходимый для авторизации токен должен быть получен вручную, в протоколе не заложено возможности авторизации получения с целью обеспечения безопасности пользователя, так как последний должен подтвердить свое намерение передать определенные права определенному приложению. Это допустимо, когда пользователь хочет предоставить права своему доверенному приложению, но использование OAuth становится невозможным для автоматизации взаимодействия двух или

более приложений. Ко всему прочему, для взаимодействия по протоколу OAuth требуется наличие постоянно запущенного сервера авторизации на неком узле программной системы.

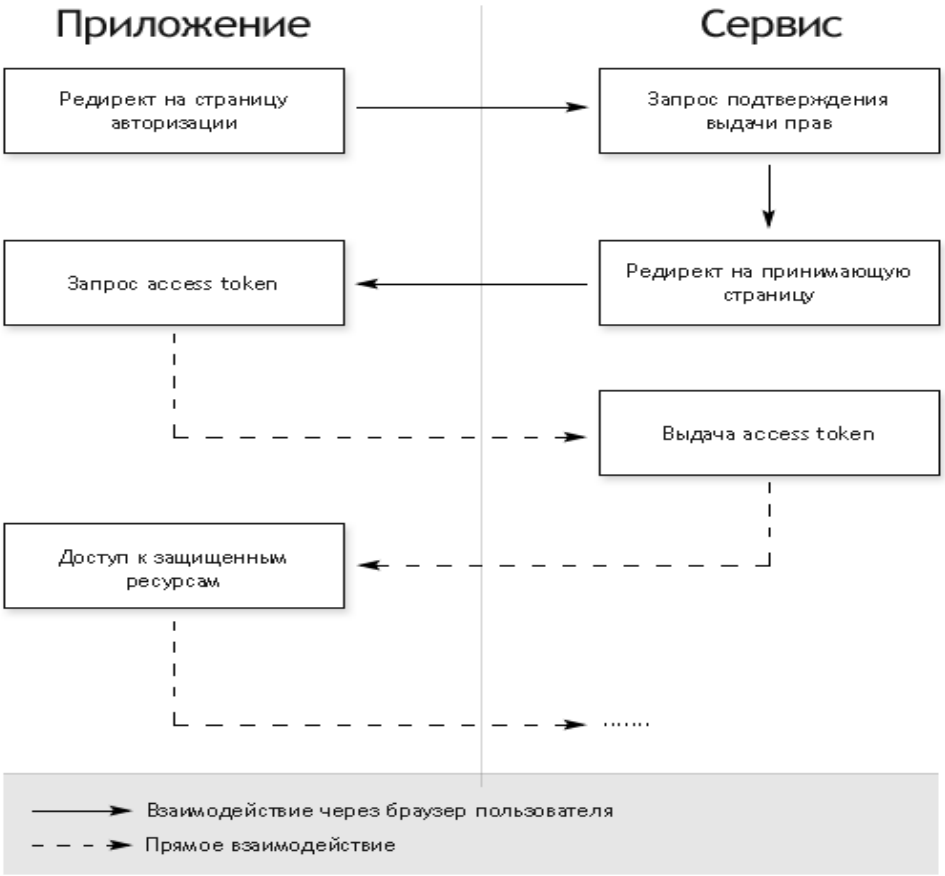

Рис. 1. Принцип взаимодействия приложений при авторизации с помощью протокола OAuth

Стоит упомянуть про механизм аутентификации OpenID, который, в свою очередь, является отдельным протоколом, не связанным с OAuth, тем не менее в случае с OpenID пользователь удостоверяет приложение, что выдает себя за того, кем и является, но данный протокол не позволяет назначать определенные права и контролировать их. Помимо отсутствия автоматизированной авторизации из недостатков OAuth стоит отметить то, что безопасность OAuth основана на SSL, а это серьезно повышает требования к ресурсам и порой становится препятствием к использованию протокола в высоконагруженных системах. Несмотря на то, что OAuth 2.0 был принят в 2006 г., в спецификацию еще вносятся изменения, поэтому невозможно быть уверенным, что в будущем не произойдет изменений без поддержки обратной совместимости.

2. Метод авторизации клиентских приложений в HDP

HDP (Heterogenous Data Protocol) представляет собой протокол для обмена гетерогенными данными в распределенной программной среде. HDP использует жестко структурированные данные в формате JSON со специальными служебными метками согласно спецификации протокола.

Узел в распределенной системе, который желает предоставить доступ остальным узлам по HDP, обязан сформировать файл HDP-сервиса, в котором будут перечислены доступные клиентам функции с указанием входных параметров и выходной схемы данных. По каждому входному параметру допустимо указывать значения по умолчанию, правила проверки данных этого параметра и др.

Протокол позволяет гетерогенным программным системам осуществить обмен данными с гарантией целостности данных, возможностью их проверки, построения распределенной сервисной архитек-

туры на стороне провайдера данных. Протокол имеет собственную спецификацию, поэтому обмен данными между различными приложениями не вызывает противоречий. Допускается произвольное количество форматов сериализации данных, предлагая JSON как основной. Но независимо от протокола и его спецификации провайдер данных может выбрать желаемый формат сериализации данных, в котором он будет получать данные от клиентов.

Метод авторизации клиентских приложений в распределенной программной системе, построенной на основе протокола HDP, представляет собой заложенную в ядро протокола возможность хранения и проверки токена: уникальная строка, идентифицирующая заданный набор доступных ролей. Согласно спецификации протокола HDP вся функциональность провайдера данных делится на отдельные функции микросервисов. Каждый микросервис вправе самостоятельно генерировать и выдавать токены с учетом их срока жизни. При запросе к функции сервиса, которая требует авторизации, HDP сначала проверяет наличие такого токена в уже используемых и сравнивает его со сроком годности. В случае если действительный токен найден, запрос будет направлен к микросервису, иначе будет выдан ответ с HTTP заголовком 401 и запрос на микросервис отправлен не будет.

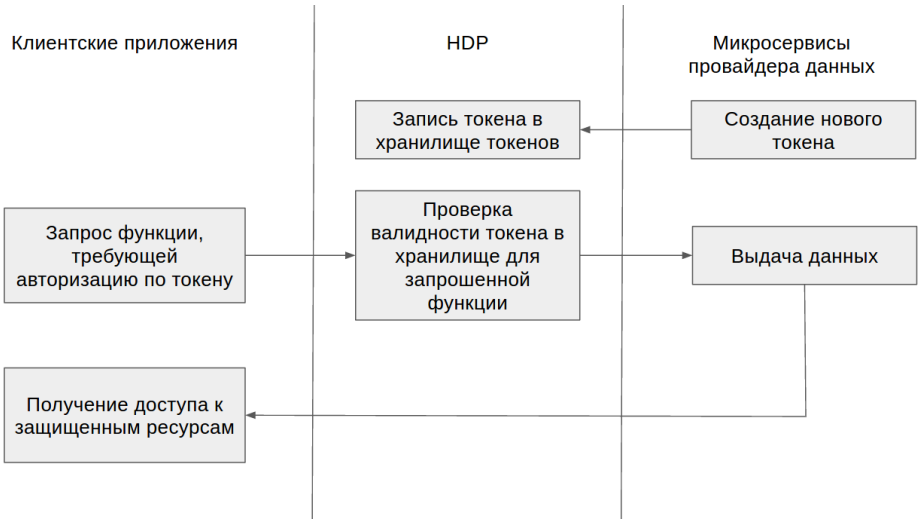

Рис. 2. Принцип взаимодействия приложений при авторизации с помощью метода авторизации HDP Auth

Как видно из рис. 2, каждый микросервис в произвольный момент может заносить и управлять токенами в хранилище сервиса HDP, для этого необходимо отправить токен с указанием срока годности на определенный протоколом адрес. HDP, получая данный запрос, складывает поступивший токен в свое хранилище, и последующие от клиентов запросы с данными токенами признаются корректными.

Данная логика позволяет снизить вычислительные сложности, объемы и количество HTTP запросов к микросервисам, что положительным образом сказывается на общей нагрузке, которую испытывает программная система. Пользователь полностью отсутствует в данном методе авторизации, что позволяет автоматизировать добавление и удаление новых микросервисов. Каждый токен может быть выдан на ограниченный или неограниченный срок действия как для одной, так и для нескольких функций, что привнесет гибкость в данный метод авторизации. Микросервис, выдавший токен в любой момент, может отозвать его, даже если его срок годности не изменился; допустимо менять и срок годности для токена, который уже находится в хранилище токенов.

Описание метода взаимодействия программных узлов при авторизации через HDP авторизационный модуль.

1. Поставщик данных регистрирует правила соответствия токена на сервере HDP.

2. Микросервис поставщика данных создает специальный токен для выбранного клиента или группы клиентов и / или пользователей.

3. Микросервис поставщика данных регистрирует данный токен на сервере HDP посредством отправки POST запроса на адрес /hdpauth/setToken.

4. Модуль авторизации, получив токен, проводит его проверку, согласно правилам, зарегистрированным в пункте 1.

4.1. В случае если проверка была успешно пройдена, HDP Auth выдает HTTP ответ с кодом 200 и уведомляет о том, что токен был успешно зарегистрирован.

4.2. Иначе HDP Auth выдает HTTP ответ с кодом 400 и уведомляет о том, что токен не был зарегистрирован. В этом случае микросервис заново генерирует токен согласно правилам, зарегистрированным в рамках пункта 1, и далее действует аналогично, начиная с пункта 2.

5. Каждый поступающий клиентский запрос к защищенным токенами ресурсам HDP Auth выполняет проверку:

5.1. Если пришедший токен найден в хранилище токенов и у данного токена не истек срок годности, запрос считается корректным и продолжается его обработка согласно спецификации протокола HDP.

5.2. Если пришедший токен не был найден в хранилище токенов или у него истек срок годности, то HDP прекращает обработку запроса, убирает истекший токен из хранилища и возвращает HTTP ответ с кодом 401.

В любой момент эксплуатации системы микросервис имеет возможность запросить список всех действующих токенов по некой защищенной функции и удалить токен, который утратил необходимость в нахождении в хранилище.

Пример регистрации корректного токена:

```
POST /hdpauth/setToken HTTP/1.1
Host: auth.hdpexample.com
Content-Type: aaplication/x-www-form-urlencoded
token=8un1847d5jyy5m0guf5pqk4l0ljg5ab953l7&function=f&expires_in=0
```

Ответ

```
HTTP/1.1. 200 OK
Content-Type: application/json
{
   “status” : “ok”
}
```

Пример регистрации некорректного токена:

```
POST /hdpauth/setToken HTTP/1.1
Host: auth.hdpexample.com
Content-Type: aaplication/x-www-form-urlencoded
token=l7&function=f&expires_in=0
```

Ответ

```
HTTP/1.1. 400 Bad request
Content-Type: application/json
{
   “status” : “error”,
   “message” : “Insufficient token length, must be greater than 10”
}
```

Пример отзыва токена:

```
POST /hdpauth/removeToken HTTP/1.1
Host: auth.hdpexample.com
Content-Type: aaplication/x-www-form-urlencoded
token=8un1847d5jyy5m0guf5pqk4l0ljg5ab9wc4r53l7&function=f
```

Ответ

```
HTTP/1.1. 200 OK
Content-Type: application/json
{
   “status” : “ok”
}
```

Ответ, если данный токен не был найден

```
HTTP/1.1. 404 OK
Content-Type: application/json
```

Пример запроса токенов по функции:

```
POST /hdpauth/getToken HTTP/1.1
Host: auth.hdpexample.com
Content-Type: aaplication/x-www-form-urlencoded
function=f
```

Ответ

```
HTTP/1.1. 200 OK
Content-Type: application/json
{
   “status” : “ok”,
   “tokens” : [“8un1847d5jyy5m0guf5pqk4l0ljg5ab9wc4r53l7”,
“dwq83d5jyy5m0guf5pqk4l0ljg5ab9wc4r53l1”]
}
```

Пример клиентского запроса к функции в случае корректного токена:

```
POST /authclosed/function HTTP/1.1
Host: clientapp.com
Content-Type: aaplication/x-www-form-urlencoded
updatedparam=newvalue&token=8un1847d5jyy5m0guf5pqk4l0ljg5abc4r53l7
```

Ответ

```
HTTP/1.1. 200 OK
Content-Type: application/json
{
   “status” : “ok”,
}
```

Пример клиентского запроса к функции в случае некорректного токена:

```
POST /authclosed/function HTTP/1.1
Host: clientapp.com
Content-Type: aaplication/x-www-form-urlencoded
updatedparam=newvalue&token=8u0ljg5ab9wc4r53l7
```

Ответ

```
HTTP/1.1. 401 Unauthorized
```

Заключение

В работе был проведен анализ проблемы авторизации клиентских приложений в распределенной высоконагруженной программной системе, указаны достоинства и недостатки существующих методов, предложен метод централизованного управления авторизационной функциональностью. Данный метод позволяет предоставить всем запущенным микросервисам единую входную точку авторизации, что положительным образом сказывается на скорости разработки высоконагруженной системы, снижает возможное число ошибок в авторизационной функциональности и повышает толерантность системы к изменению авторизационной логики. В будущем данный метод может подстраиваться под новые требования к механизму авторизации, например, различные усовершенствования для повышения уровня безопасности при доступе к данным. Предложенный метод разумно использовать в распределенных высоконагруженных системах, построенных на основе протокола HDP, где имеется внушительный объем защищенной информации.

ЛИТЕРАТУРА

1. Павликов М.К. Протокол HDP // Вестник компьютерных и информационных технологий. 2016. № 8. С. 52–56.
2. Richer J., Sanso A. OAuth 2 in Action. Manning Publications, 2017. 375 p.

3. Bihis C. Mastering OAuth 2.0. Packt Publishing, 2015. 238 p.
4. Siriwardena P. Advanced API Security. Apress, 2014. 230 p.
5. Spasovski M. OAuth 2.0 Identity and Access Management Patterns. Packt Publishing, 2013. 128 p.
6. Parecki A. OAuth 2.0: The Definitive Guide. O'Reilly Media, 2014. 400 p.

Павликов Максим Константинович. E-mail: severemax@yandex.ru
Московский авиационный институт

Поступила в редакцию 14 апреля 2017 г.

Pavlikov Maxim K. (Moscow Aviation Institute, Russian Federation).
Authorization method of client applications in a heterogeneous distributed software system built on the basis of the HDP protocol.
Keywords: HDP; a distributed software system; the authorization system in distributed software system; OAuth.

DOI: 10.17223/19988605/41/7

In this paper, the authorization of client applications in a software system built on the basis of the HDP protocol is considered. Authorization is the granting of rights to perform a certain operation and checking such rights when performing an operation. In computer systems, authorization is necessary to access protected information or methods of obtaining, aggregating secure information. In the case of a monolithic application, there are usually no difficulties in implementing authorization, because there are many ready-made solutions to this problem. In various programming languages and frameworks, a lot of libraries have accumulated, which expand the language capabilities for client authorization.

To date, the most popular authentication protocol between two applications is OAuth. This protocol allows the application to get authorization from the service using HTTP requests. The main advantage of the approach is that the client does not transfer the login and password from its account to the trusted application, but only assigns certain rights that are reflected in the received token. OAuth uses interaction through the user's browser and direct interaction between the software systems to perform the full authorization process. This leads to the fact that the token necessary for authorization must be received manually, the protocol does not provide for the possibility of authorization of receipt in order to ensure the security of the user, as the latter must confirm its intention to transfer certain rights to a particular application. This is acceptable when the user wants to grant rights to his trusted application, but using OAuth becomes impossible to automate the interaction of two or more applications. In addition, OAuth requires a constantly running authorization server on a certain node of the software system.

The method of authorization of client applications in a distributed system built on the basis of the HDP protocol is the possibility of storing and checking a token embedded in the core of the protocol. According to the specification of the HDP protocol, all the functionality of the data provider is divided into separate functions of the microservices. Each microservice may independently generate and issue tokens with regard to their lifetime. When requesting a service function that requires authentication, HDP first checks the availability of such a token in the ones already used and compares it to the expiration date. In the event that a valid token is found, the request will be forwarded to the microservice, otherwise a response will be issued with the HTTP header 401 and the request for the microservice will not be sent.

REFERENCES

1. Pavlikov, M.K. (2016) HDP protocol. *Vestnik komp'yuternykh i informatsionnykh tekhnologiy – Herald of Computer and Information Technologies*. 8. pp.52–56. (In Russian). DOI: 10.14489/vkit.2016.08.pp.052-056
2. Richer, J. & Sanso, A. (2017) *OAuth 2 in Action*. Manning Publications.
3. Bihis, C. (2015) *Mastering OAuth 2.0*. Packt Publishing.
4. Siriwardena, P. (2014) *Advanced API Security*. Apress.
5. Spasovski, M. (2013) *OAuth 2.0 Identity and Access Management Patterns*. Packt Publishing.
6. Parecki, A. (2014) *OAuth 2.0: The Definitive Guide*. O'Reilly Media.