УДК 621

АУТЕНТИФИКАЦИЯ ПРИ ПОМОЩИ JSONWEBTOKEN(JWT). СТРУКТУРА JWT

Курьянович Д.Ю.

Научный руководитель – Прибыльская Н.М., ст. преподаватель

Цель работы – рассмотреть процесс аутентификации при помощи JsonWebToken и его структуру.

JSON WebToken (JWT) — это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях.

JWT состоит из трех основных частей: заголовка (header), нагрузки (payload) и подписи (signature). Заголовок и нагрузка формируются отдельно, а затем на их основе вычисляется подпись.

Заголовок состоит из двух полей: типа токена (в данном случае JWT) и алгоритма хэширования подписи. Пример представлен на рисунке 1.

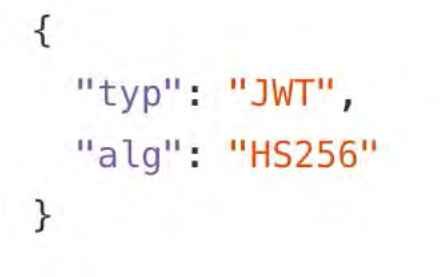

Рисунок 1 – заголовок токена

Официальный сайт jwt.io предлагает два алгоритма хэширования: HS256 и RS256. Но на деле можно использовать любой алгоритм с приватным ключом.

Нагрузка (payload) — это любые данные, которые вы хотите передать в токене. Но стандарт предусматривает несколько зарезервированных полей. Все эти поля не являются обязательными, но их использование не по назначению может привести к коллизиям.

Любые другие данные можно передавать по договоренности между сторонами, использующими токен. Пример payload представлен на рисунке 2.

Рисунок 2 – поля нагрузки токена

Payload не шифруется при использовании токена, поэтому не стоит передавать в нем данные, которые не должны попасть в открытый доступ.
Подпись вычисляется на основе заголовка и нагрузки. Таким образом, если кто-то попытается изменить данные в токене, он не сможет изменить подпись, не зная приватного ключа. При аутентификации приватным ключом может выступать пароль пользователя (или хеш от пароля).
Сначала header и payload приводятся к формату JSON, а затем переводятся в base64. Результат представлен на рисунке 3.

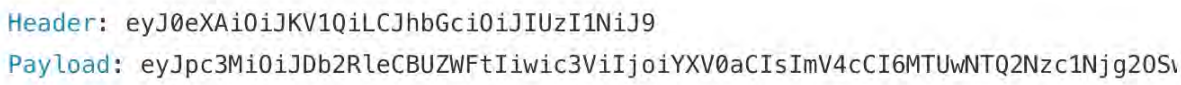

Рисунок 3 – примеры заголовка и нагрузки токена

Затем, две эти строки соединяются через точку и хэшируются указанным в заголовке алгоритмом. Допустим, пользователь использует пароль: HS256('eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9' + '.' + 'eyJpc3MiOiJDb2RleCBUZWFtIiwic3ViIjoiYXV0aCIsImV4cCI6MTUwNTQ2Nzc1Njg2OSwiaWF0IjoxNTA1NDY3MTUyMDY5LCJ1c2VyIjoxfQ', 'password') = '0ynjTRZT9Uk77TnGy_g9Mxi1decLBjKxQK6e2dVzDJo'
Результат работы алгоритма и есть подпись. Теперь осталось только сформировать сам токен, для этого нужно через точку соединить заголовок и нагрузку в base64 и подпись: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJDb2RleCBUZWFtIiwic3ViIjoiYXV0aCIsImV4cCI6MTUwNTQ2Nzc1Njg2OSwiaWF0IjoxNTA1NDY3MTUyMDY5LCJ1c2VyIjoxfQ.0ynjTRZT9Uk77TnGy_g9Mxi1decLBjKxQK6e2dVzDJo
Токен готов. Проверить его можно на jwt.io.

Рассмотрим аутентификацию. После первого логина, клиенту возвращается сгенерированный сервером JWT. При каждом следующем запросе, клиент должен передавать JWT установленным API способом (например, через заголовок или как параметр запроса). Сервер декодирует header и payload и проверяет зарезервированные поля. Если все в порядке, по указанному в header алгоритму составляется подпись. Если полученная подпись совпадает с переданной, пользователя авторизуют. Можно реализовать всю эту схему вручную, а можно использовать одну из библиотек указанных на jwt.io.

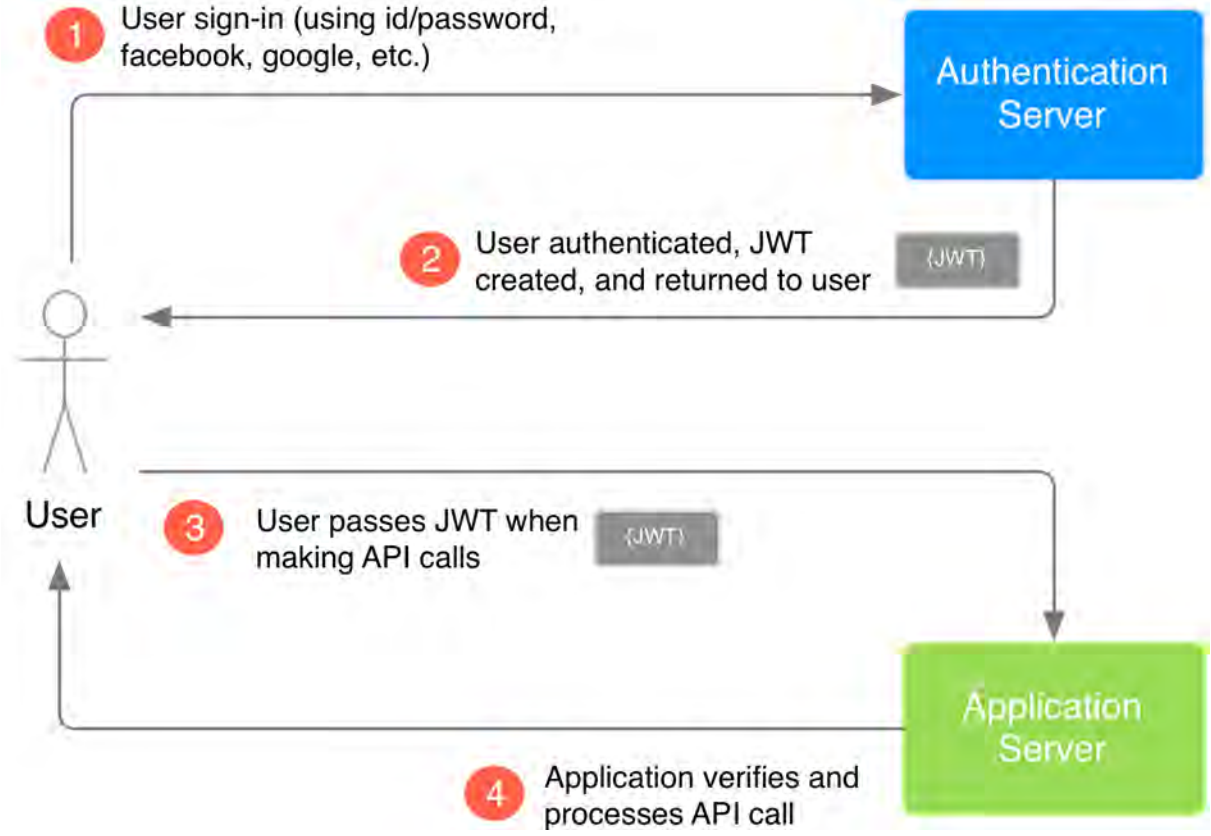

Рисунок 4 – схема аутентификации при помощи JWT