Б.Е. УРАЗАКАЕВ

Идентификация сеанса пользователя в веб-приложениях

ФГБОУ ВПО
«Владимирский
государственный
университет имени
А.Г. и Н.Г. Столетовых»,
г. Владимир

УДК 004.77

В данной статье рассматриваются два способа аутентификации и авторизации запросов приложения-клиента к серверу. Показано сравнение двух способов, достоинства и недостатки. Проведены границы области применения обоих методов идентификации.

Веб-приложение - это клиент-серверное приложение, в котором обмен данными между клиентом (веб-браузером) и сервером (веб-сервером) происходит посредством протокола HTTP. Данный протокол изначально не предусматривал сеансовость - способность поддерживать сеанс связи с клиентом. Эту проблемы решили при помощи специальных заголовков cookie, которые и сейчас являются наиболее распространенным способом идентификации сеанса (далее, сессии) пользователя[3].

Cookie

Cookie используются практически повсеместно, и не только для идентификации пользователя, но и для сохранения пользовательских настроек на сайте или ведения статистики. Однако в данной статье они будут рассмотрены именно в качестве способа идентификации сессии пользователя. Для того, чтобы понять, как это происходит на данный момент, рассмотрим типовой процесс аутентификации с использованием cookie:

1. Пользователь вводит логин и пароль в поля формы и отправляет их на сервер;

2. Сервер сравнивает присланные данные с теми, что хранятся в хранилище и идентифицирует пользователя;

3. На сервере создается объект сессии - чаще всего это хранилище типа “ключ-значение”, в котором могут храниться разные атрибуты, необходимые для работы с пользователем (например, список товаров в корзине);

4. Для идентификации - однозначной ассоциации - объекта сессии с пользователем сервер посылает клиенту сгенерированный идентификатор сессии (чаще всего значение хэш-фукнции для случайной последовательности байтов). Этот идентификатор сессии затем пересылается клиентом - браузером - при каждом запросе к странице[2].

Вышеописанный процесс предоставляет удобный способ идентификации сессии пользователя для приложений: пользователя не нужно вводить свои логин и пароль для того чтобы сервер мог удостовериться в том, что перед ним аутентифицированный пользователь.

Проблем не возникает до тех пор, пока сервер один. Что делать, если серверов несколько? Рассмотрим проблему на следующем примере.

Имеется балансировщик нагрузки - сервер, принимающий запросы пользователей и переадресовывающий их на наименее загруженные серверы в его списке. Допустим, аутентификация пользователя, а следовательно и создание объекта сессии, произошло на одном сервере, а несколько следующих запросов пользователя были переадресованы другому серверу. Другой сервер не сможет идентифицировать объект сессии, т.к. этот объект у него отсутствует. Вопрос: как распределить объекты сессий между серверами? Ответов на этот вопрос существует несколько. В частности, можно использовать отдельный сервер для хранения сессий (например, Redis или Apache Cassandra). Однако все это требует дополнительных затрат и увеличивает нагрузку на серверную часть.

Еще одной проблема является достоверность полученного сервером cookie. Если в них хранятся настройки приложения, как приложению удостовериться, что они не изменились на клиентской стороне? Ведь изменять Cookie должен только сервер посредством специального заголовка в ответе[3].

JSON Web Token

С приходом одностраничных (single-page) приложений на смену традиционных веб-приложений, в которых браузер при переходе на страницу пересылает отправляет запрос, обрабатывая который, сервер генерирует HTML-код страницы и присылает его в качестве ответа на запрос клиента, способ разработки и взаимодействия клиента с сервером изменился. Теперь все чаще создаются так называемые RESTful веб-сервисы, которые не генерируют HTML код, а работают только с данными. Весь HTML-код уже содержится на клиенте, нужно только получать актуальные данные с сервера. Традиционно в подобных веб-сервисах использовались маркеры (tokens) для авторизации пользователя. Однако подобного рода сервисы используются не только в веб, но и в мобильных приложениях, где использование Cookie вызывает затруднения. Для унификации процесса идентификации сеанса пользователя был создан стандарт JSON Web Token [1].

Что представляет стобой JSON Web Token? Это структура, состоящая из 3-х частей:

1. Заголовок

Содержит в себе метаданные маркера (token). Состоит из как минимум двух частей: поля “typ”, определяющего тип подписи (по стандарту здесь должно быть значение “jwt”), и алгоритм шифрования “alg”. Данные хранятся в формате JSON и закодированы алгоритмом Base64;

2. Полезная нагрузка

Тут могут храниться данные, которые обычно содержатся в сессии: товары в корзине, email пользователя и т.д.

3. Цифровая подпись

Эта часть представляет собой цифровую подпись маркера. Она вычисляется отправляющей стороной и использует для проверки целостности сообщения, содержащегося в маркере. Вычисление происходит при помощи алгоритма, описанного в заголовке. Если алгоритм требует секретный пароль, то он так же сообщается в параметрах вызова хэш-функции.

Маркер такого формата пересылается клиентом на сервер каждый раз, таким образом сервер может понять, кто именно послал запрос.

Сравнение двух способов

JSON Web Token решает две проблемы, связанные с использованием Cookies. Во-первых, все данные, которые раньше хранились в сессии, теперь хранятся в JWT, т.е. нет необходимости серверу хранить у себя объект сессии и ассоциировать его с определенным пользователем. Запрос может приходить на разные серверы, однако сервер всегда способен понять, какой именно пользователь прислал запрос. Во-вторых, JWT исключает возможность подделать маркер благодаря цифровой подписи. Что значит подделка подписи? Допустим такую ситуацию с интернет-магазином. Злоумышленник проанализировал, каким образом проходит передача Cookies, и начал совершать запросы, подставляя различные значения в передаваемый заголовок Cookies. Рано или поздно, найдутся такие Cookies, которые соответствуют какому-либо пользователю. Маркеры же подписываются на стороне сервера с использованием пароля, который никак не передается клиентской стороне. Здесь можно применить метод полного перебора, но из-за того, что в JWT используется цифровая подпись, количество возможных вариантов возрастает экспоненциально, и вероятность успешности такой атаки сводится практически к нулю.

У обоих подходов есть один существенный недостаток: отсутствие защиты от атаки MiTM (Man in The Middle - человек посередине), суть которой заключается в том, что злоумышленник может считывать обмен трафиком, а следовательно украсть как Cookie, так и JWT. Для того, чтобы устранить такой вид атаки, необходимо использовать SSL (подходит для Cookie и JWT), либо использовать спецификацию JWE[4] для защиты (подходит только для JWT).

Разработка программных продуктов, спецификаций и прочего всегда проходила итеративно: создавался стандарт, находились уязвимости, эти уязвимости устранялись в следующей версии стандарта, либо же предлагалась альтернатива, как в случае в JWT, которая больше отвечает потребностям современного мира разработки. Вполне возможно, что JWT станет стандартом для обмена не только для веб-сервисов, но так же и для обычных веб-приложений, или же для децентрализованного обмена.

Литература

1. RFC 7519 «JSON Web Token (JWT)»
2. RFC 6265 «HTTP State Management Mechanism»
3. Д. Скембрей, М. Шема, Й.-М. Чен, Д. Вонг Секреты хакеров. Безопасность Web-приложений — готовые решения. М: Вильямс, 2003. 384 с.
4. RFC 7516 «JSON Web Encryption (JWE)»

ТЕЛ. ДОМАШНИЙ (902) 883-29-56

ЭЛ. ПОЧТА
BOGDAN.URAZAKAEW@GMAIL.COM