Технология JWT языка программирования Node JS
Васильев П. А.

Васильев Петр Алексеевич / Vasilyev Petr Alexeevich - студент,
кафедра информационных технологий,
институт математики и информатики
Северо-Восточный федеральный университет, г. Якутск

Аннотация: *в статье предлагаются сведения о технологии JWT. О его достоинствах и недостатках в использовании на web – приложениях. Основные методы и использование технологии в реальных условиях.*
Ключевые слова: *Node JS, JWT, Web - программирование, JavaScript, авторизация, аутентификация, SPA.*

На сегодняшний день web – технологии и интернет сильно развились и продолжают развиваться очень быстро. Развиваются способы реализации разных функций – передача данных, защита от сетевых атак, хостинг на сервере и т.д. Одним из таких способов является авторизация в современных сайтах.

В настоящее время в большинстве сайтов в основном для авторизации используются «сессии» и «cookie» (временное хранилище браузера). То есть пользователь регистрируется и входит в веб – приложение, на что приложение создает для него сессию и каждый раз проверяет, есть ли этот пользователь в сессии. Клиент сохраняет сессию у себя в браузере в cookie. Такой способ авторизации пользователя сейчас очень популярно используется, но с развитием мобильных приложений веб – сайты начали переходить на приложения, и разработчикам приходится создавать другую систему авторизации именно для мобильных приложений. А метод с сессиями очень сложен в реализации для них и требует объемной большой работы. Поэтому разработчики ищут альтернативные способы аутентификации. Одним из них является JSON Web Token (JWT)/

JWT (JSON WEB TOKEN) [1] – так называемый маркер в специальном формате, который содержит минимальную информацию в зашифрованном виде, необходимую для авторизации и получения доступа к приложению. Представьте JWT как ключ – карту, которую предоставили вам на работе. Каждый день вы ходите на работу и открываете им свой офис и можете находиться в здании компании. Чтобы работать, вы выдали свою личную информацию – имя, водительские права и т.п. Если бы вы решили уйти с работы или вас уволили (вышли из системы), компания может просто отключить ключ.

Как работает JWT:

1. Пользователь запрашивает доступ у вашего сервера, высылая ему логин и пароль.
2. Сервер проверяет пользователя и высылает ему access token, который имеет некий expiration date (время использования).
3. Пользователь использует этот access token для доступа к ресурсам на вашем сервере.
4. По наступлению expiration date (например, через 2 недели) пользователю придется вновь пройти процедуру аутентификации

Основным минусом такого подхода является то, что пользователь каждый раз должен проходить авторизацию при маленьком значении времени использования. Но если использовать большое значение, то возникают проблемы с безопасностью:

- В случае кражи ключа злоумышленник сможет получить доступ к ресурсу на длительный период.
- Если администратор захочет ограничить пользователя в правах или поменять его роль, то пользователю придется вновь пройти процедуру аутентификации, чтобы ключ обновился.

Для того чтобы решить проблемы, часто предлагается наряду с кратковременным ключом дополнительно использовать второй ключ для обновления основного ключа. При этом при авторизации пользователь получает ключ обновления (с длительностью, например, в 1 год) и ключ (например, с длительностью в 30 минут). И для доступа к ресурсам он по-прежнему использует основной ключ, но теперь через 30 минут для того, чтобы получить новый ключ, ему достаточно отправить ключ обновления и тот вышлет ему в ответ свежий ключ, при этом очередной раз проверив права пользователя. Но такой подход очень сильно усложняет программный код и требует большого объема работы. Поэтому для реальных проектов требуется хорошее знание этого способа авторизации и аутентификации.

В реальном проекте требуется обеспечить безопасность SSL и ключа синхронизации. Также требуется добавить https сертификацию в свой проект при запуске.

Литература

1. *Блинов А. М.* Информационная безопасность. СПб.: СПбГУЭФ, 2010. 96 с.