

Использование совместно Refresh-токена и JWT для повышения надёжности системы

Вихляев Дмитрий Романович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Данная статья содержит описание системы использующей пару токенов в процессе верификации пользователей. Для представления Refresh и Access токенов используется стандарт JWT. В результате исследования будет рассмотрен способ создания системы авторизации поддерживающая два токена.

Ключевые слова: JWT, Refresh-токен, авторизация.

Using the Refresh token and JWT together to increase system reliability

Vikhlyayev Dmitry Romanovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article describes a system that uses a pair of tokens in the user verification process. The JWT standard is used to represent Refresh and Access tokens. As a result of the study, a way to create an authorization system that supports two tokens will be considered.

Keywords: JWT, Refresh token, authorization.

1 Введение

1.1 Актуальность

Использование Refresh-токена совместно с JSON Web Token (JWT) представляет собой современный подход к обеспечению надёжности и безопасности систем аутентификации и авторизации. Актуальность данного метода обусловлена растущими требованиями к защите данных в распределённых системах, где компрометация токенов может привести к несанкционированному доступу и утечке конфиденциальной информации. Совместное использование JWT и Refresh-токенов позволяет эффективно минимизировать риски, связанные с длительным сроком действия токенов, обеспечивая баланс между безопасностью и удобством пользователей. Refresh-токены обладают более продолжительным сроком действия и хранятся на сервере, что обеспечивает их защиту от атак типа «man-in-the-middle» и предотвращает их перехват. Они передаются только через защищённые каналы связи, такие как HTTPS, что снижает риск их

компрометации. В случае утраты или кражи JWT пользователь может получить новый токен, используя Refresh-токен, а старый токен становится недействительным, что минимизирует потенциальный ущерб от утечки.

1.2 Обзор исследований

В своей работе Д.С.Матюшин, А.В.Богорадникова описали способ генерации токена авторизации посредством библиотеки jwt [1]. Д.А.Макаров подробно изучил механизм авторизации с использованием технологии JWT [2]. И.С.Голубь создал приложение .NET core web api с использованием JWT-токенов для авторизации [3]. В.С.Попов реализовал процесс авторизации пользователя angular веб приложения с использованием JWT-токена [4]. А.О.Колесников исследовал идентификацию пользователей клиент-серверных приложений с помощью JWT-токена [5].

1.3 Цель исследования

Цель исследования – описать метод использования Refresh-токена совместно с JWT для повышения надёжности системы.

2 Материалы и методы

Для описания используются схемы и программные реализации, написанные на языке программирования JavaScript.

3 Результаты и обсуждения

В системах с авторизацией пользователей по токену, частой практикой является использование двух токенов вместо одного. Использование только одного JWT влечёт за собой уязвимости в безопасности, в случае похищения. Данные уязвимости можно исправить более гибко и эффективно с добавлением второго токена. В таком случае JWT (Access Token) используется для быстрого и безопасного доступа к ресурсам API. Обычно имеет короткий срок действия (например, 15 минут). Проверяется только по сигнатуре без обращения к базе данных. Refresh Token применяется для получения нового JWT, когда текущий токен истекает. Обычно имеет более длительный срок действия (дни или недели). Позволяет избежать необходимости повторной аутентификации пользователя.

Refresh токен отличается от JWT по своему назначению и способу использования, что обуславливает необходимость его хранения, а не проверки исключительно по сигнатуре, как в случае с JWT. JWT, или токен доступа, используется для быстрого и безопасного доступа к защищённым ресурсам API. Его ключевой особенностью является возможность проверки токена без обращения к серверу, поскольку вся необходимая информация содержится внутри токена и подписана. Это обеспечивает высокую скорость обработки запросов, но делает JWT менее подходящим для длительных сессий, так как истечение срока действия токена требует либо повторной аутентификации пользователя, либо использования механизма обновления токена.

В отличие от JWT, Refresh токен предназначен для получения новых токенов доступа после истечения срока действия текущего JWT. Этот токен имеет более длительный срок жизни и предоставляет возможность продления сессии пользователя без повторной аутентификации. Однако использование Refresh токена в виде само подписанного токена, аналогичного JWT, снижает безопасность системы, так как отсутствие контроля со стороны сервера исключает возможность оперативного отзыва или аннулирования токена в случае его компрометации.

Хранение Refresh токенов в базе данных или аналогичном серверном хранилище позволяет централизованно управлять сессиями. Например, сервер может реализовать механизмы ротации токенов, при которых старый Refresh токен становится недействительным после выдачи нового. Это значительно снижает риск злоупотреблений даже в случае утечки токена. Кроме того, серверное хранилище обеспечивает возможность отзыва токенов по требованию, что может быть актуально при выходе пользователя из системы или при обнаружении подозрительной активности.

Если Refresh токены хранятся на стороне клиента, например, в HTTP-only cookie, сервер по-прежнему должен иметь возможность контролировать их валидность. В этом случае используется сочетание клиентского хранения и серверного контроля, что обеспечивает баланс между безопасностью и удобством использования (рис.1).

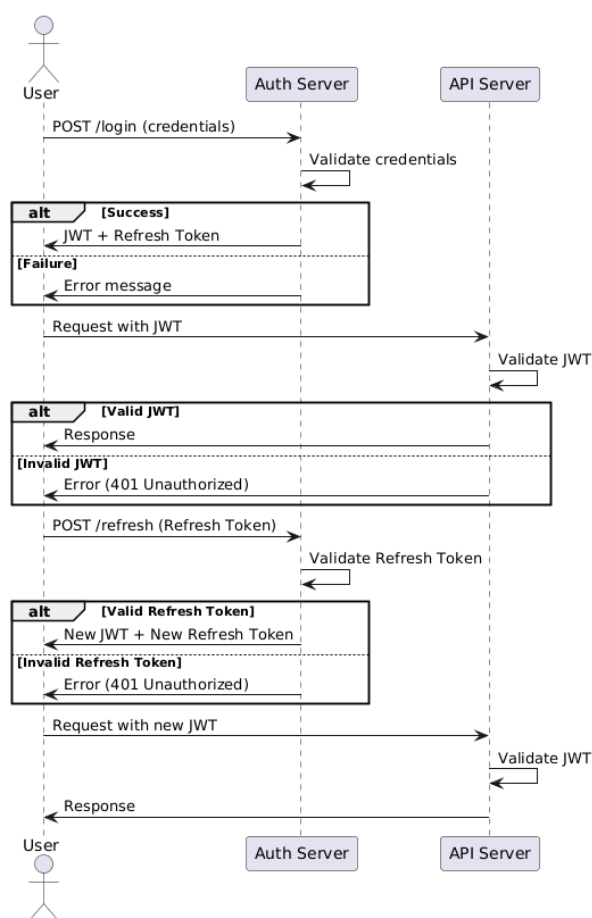


Рис. 1. Последовательность функций авторизации пользователя

Управление несколькими Refresh токенами для одного пользователя представляет собой важный аспект обеспечения безопасности и удобства использования в системах аутентификации. Хранение до пяти токенов на одного пользователя позволяет поддерживать одновременную работу с нескольких устройств, обеспечивая при этом эффективный контроль за использованием токенов. При каждом запросе обновления токена клиент отправляет текущий Refresh токен на сервер аутентификации, который проверяет его валидность, обращаясь к хранилищу токенов, например, базе данных. В случае успешной проверки токен помечается как недействительный, чтобы предотвратить его повторное использование, что снижает риск компрометации. После успешной проверки сервер генерирует новый JWT для доступа к защищённым ресурсам и создаёт новый Refresh токен. Новый токен добавляется в хранилище токенов пользователя с указанием времени его создания. При этом сервер контролирует общее количество токенов, хранящихся для данного пользователя. Если количество токенов превышает установленный лимит, например, пять токенов, сервер удаляет самый старый из них на основании времени создания. Такой подход позволяет ограничивать количество одновременно активных токенов, предотвращая перегрузку системы и снижая риск несанкционированного доступа (рис.2).

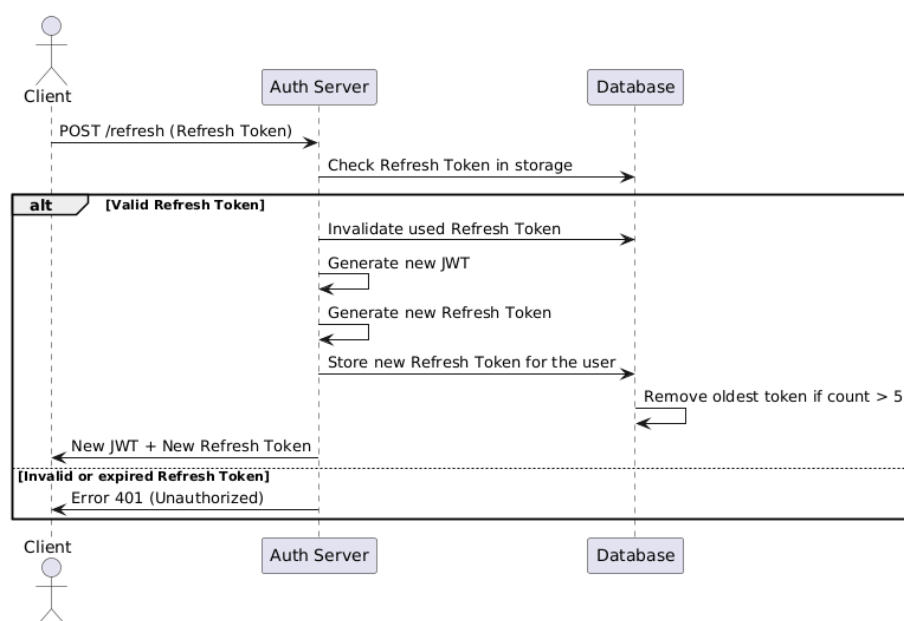


Рис. 2. Проверка Refresh-токена

Клиент сохраняет JWT в памяти приложения, refresh размещается в Cookie. Cookies с токенами должны быть настроены с параметрами httpOnly, secure и sameSite для повышения безопасности. Параметр httpOnly гарантирует, что cookies будут недоступны через JavaScript, тем самым защищая от XSS-атак. Параметр «secure» разрешает отправку cookies только через HTTPS-соединения. Что предотвращает перехват токенов при передаче через незашифрованные каналы. Параметр sameSite позволяет ограничить

отправку cookies только на тот же сайт, что снижает риски атак типа CSRF (рис.3).

```
// Аутентификация
app.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const user = users.find(user => user.username === username);
  if (!user) {
    return res.status(400).json({ message: 'Invalid username or password' });
  }

  // Проверка пароля
  const isPasswordValid = await bcrypt.compare(password, user.password);
  if (!isPasswordValid) {
    return res.status(400).json({ message: 'Invalid username or password' });
  }

  // Генерация токенов
  const accessToken = jwt.sign({ username }, SECRET_KEY, { expiresIn: '15m' });
  const refreshToken = jwt.sign({ username }, REFRESH_SECRET_KEY, { expiresIn: '7d' });
  REFRESH_TOKENS.push(refreshToken);

  res.cookie('refreshToken', refreshToken, {
    httpOnly: true,
    path: '/api/refresh', // Accessible only on /api/refresh
    domain: 'ex.com', // Specify the domain
    maxAge: 7 * 24 * 60 * 60 * 1000, // 7 days
  });

  res.json({ "accessToken": accessToken });
});

// Обновление токенов
app.post('/refresh', (req, res) => {
  const token = req.cookies.refreshToken;
  if (!token || !REFRESH_TOKENS.includes(token)) {
    return res.status(403).json({ message: 'Forbidden' });
  }

  try {
    const userData = jwt.verify(token, REFRESH_SECRET_KEY);
    const accessToken = jwt.sign({ username: userData.username }, SECRET_KEY, { expiresIn: '15m' });
    res.json({ accessToken });
  } catch {
    res.status(403).json({ message: 'Forbidden1' });
  }
});
```

Рис. 3. Реализация методов получения и отправки токенов на auth-сервере

В результате данной статьи, был описан подход с использованием Refresh-токена как дополнение к авторизации с использованием JWT. Продемонстрированы преимущества и особенности метода в сфере безопасности и практичности использования. Приведены примеры упрощённой программной реализации.

Библиографический список

1. Матюшин Д.С., Богорадникова А.В. Генерация токена авторизации посредством библиотеки jwt // В сборнике: Актуальные вопросы в науке и практике. Сборник статей по материалам XII международной научно-практической конференции. 2018. С. 91-94.
2. Макаров Д.А. Механизм авторизации с использованием технологии JWT // Теория и практика современной науки. 2020. № 1 (55). С. 474-476.
3. Голубь И.С. Приложение .NET core web api с использованием JWT-токенов

- для авторизации // Постулат. 2020. № 1 (51). С. 91.
4. Попов В.С. Авторизация пользователя angular веб приложения с использованием JWT-токена // В сборнике: Научное сообщество студентов XXI столетия. технические науки. сборник статей по материалам LXXXIX студенческой международной научно-практической конференции. Новосибирск, 2020. С. 133-138.
 5. Колесников А.О. Идентификация пользователей клиент-серверных приложений с помощью JWT-токена // В сборнике: EurasiaScience. Сборник статей XXXVI международной научно-практической конференции. Москва, 2021. С. 42-43.
 6. Абдурайимов Л.Н. Аутентификация в REST-приложениях с использованием spring security и JWT // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2021. № 4 (34). С. 5-13.
 7. Филатов К.Ю. Использование токенов на примере JWT в spring boot REST api приложении // Информационные технологии моделирования и управления. 2022. Т. 128. № 2. С. 153-157.
 8. Козлов С.В., Воробьев М.С. Реализация аутентификации пользователя в web-приложении с использованием стандарта JWT // Системы компьютерной математики и их приложения. 2022. № 23. С. 362-366.