

Методы безопасного использования JWT для авторизации пользователей

Вихляев Дмитрий Романович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье приводятся примеры и рекомендации по безопасному использованию JSON Web Token. Описаны виды атак и способы защиты. Реализованы примеры функций на языке JavaScript. В результате исследования будут описаны дополнительные инструменты, которые следует использовать совместно с JSON Web Token и механизмы защиты ключей шифрования.

Ключевые слова: JWT, безопасность, SQL-инъекции, XSS и CSRF

Methods of safe use of JWT tokens for user authorization

Vikhlyayev Dmitry Romanovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article provides examples and recommendations for the safe use of JSON Web Token. The types of attacks and methods of protection are described. Examples of functions in JavaScript are implemented. As a result of the study, additional tools that should be used in conjunction with the JSON Web Token and encryption key protection mechanisms will be described.

Keywords: JWT, Security, SQL Injection, XSS and CSRF

1 Введение

1.1 Актуальность

Дополнительный контроль безопасности при использовании JSON Web Token (JWT) необходим из-за особенностей его архитектуры и способов применения в современных распределенных системах. JWT широко используется для передачи данных об аутентификации и авторизации, однако неправильное использование или недостаточная защита может привести к серьезным угрозам безопасности, включая утечку данных и компрометацию системы. JWT часто используется в распределенных системах с несколькими компонентами, что требует синхронизации политик безопасности между ними. Недостатки в настройках одного из компонентов могут привести к общему снижению безопасности всей системы. Например, неправильно

настроенный сервер может пропустить важные проверки, что делает систему уязвимой.

1.2 Обзор исследований

Г.Г.Шамян провёл сравнительный анализ методологий авторизации JWT и sessions [1]. С.Н.Девицына, П.В.Пилькевич, Е.В. Удод представили способы улучшения защищённости сервисов, использующих JWT-токены [2]. В своей работе М.В.Бабичева показал примеры атак на jwt токены [3]. А.О.Колесников описал идентификацию пользователей клиент-серверных приложений с помощью JWT-токена [4]. Л.Н.Абдурайимов реализовал аутентификацию в REST-приложениях с использованием spring security и JWT [5].

1.3 Цель исследования

Цель исследования – привести примеры уязвимостей при использовании JWT подписи и методы их решения с целью предотвращения атак злоумышленников.

2 Материалы и методы

В описываемых материалах описаны методы защиты связанные с SQL-инъекциями, XSS и CSRF атаками, а также подменой токена, приведены фрагменты программного кода.

3 Результаты и обсуждения

Обеспечение безопасности при использовании JWT для проверки валидности пользователя охватывает несколько аспектов, включая хранение ключей шифрования, защиту токена у клиента, выбор алгоритма шифрования, учёт известных атак, периодическую смену ключей, использование refresh-токенов и генерацию секретных ключей.

Выбор алгоритма шифрования имеет ключевое значение для устойчивости системы к атакам. Рекомендуется использовать только один надёжный алгоритм, например, асимметричный алгоритм RS256 или симметричный HS256, с обязательным игнорированием параметра alg из заголовка токена для предотвращения атак на основе изменения указанного алгоритма. Стандарт JWT допускает использование алгоритма без подписи (none). В связи с этим возможна атака злоумышленника. В поле заголовка «alg» подставляется none и генерируется новая подпись без шифрования. Полученный токен отправляется на сервер, где неправильно спроектированная библиотека читает заголовок и сравнивает подписи по алгоритму указанному в заголовке. Для предотвращения подобных атак следует утвердить только один алгоритм шифрования и не принимать другие.

Секретные ключи для симметричных алгоритмов должны быть сгенерированы с использованием криптографических надёжных генераторов случайных чисел с целью не допустить подбор ключа.

Передаваемый JWT используется для проверки подлинности пользователя, а не для шифрования информации. Поэтому следует избегать хранения личных данных пользователя внутри полезной нагрузки. Или использовать шифрование самого токена. Также не в заголовке не в полезной нагрузке не стоит хранить какие-либо идентификаторы из базы данных. В случае ошибок кодирования это поле может быть уязвимо к SQL-инъекции.

Для обеспечения безопасности взаимодействия используются дополнительные меры, такие как короткий срок действия токенов, использование защищённых каналов передачи данных (например, HTTPS), периодическая ротация ключей подписи и внедрение механизма отзыва токенов. Токен становится недействительным только после истечения срока действия, обычно нет функций, которые позволят пользователю выйти и отменить действие токена. Тогда, если токен будет украден пользователь не сможет отозвать его заблокировав тем самым похитителя. Поэтому, в случае утраты или компрометации токена сервер аутентификации может включать идентификатор токена (jti) в список отозванных, что позволяет серверам приложений отклонять такие токены при последующей проверке (рис.1).

```
const jwt = require('jsonwebtoken');
const { v4: uuidv4 } = require('uuid');
const Redis = require('ioredis');

// Создаем подключение к Redis
const redis = new Redis({
  host: '127.0.0.1',
  port: 6379,
});

// Конфигурация JWT
const secretKey = 'your-very-secure-secret-key';
const tokenExpiry = '1h'; // Время жизни токена (часы, минуты, секунды)

// Модуль управления токенами
const tokenManager = {
  generateToken(payload) {
    const jti = uuidv4();
    const token = jwt.sign({ ...payload, jti }, secretKey, { expiresIn: tokenExpiry });
    return { token, jti };
  },
  async isTokenRevoked(jti) {
    const exists = await redis.exists(jti);
    return exists === 1;
  },
  async revokeToken(jti, expiryTime = 3600) {
    await redis.set(jti, 'revoked', 'EX', expiryTime);
    console.log(`Token with JTI ${jti} has been revoked.`);
  },
  async verifyToken(token) {
    try {
      const decoded = jwt.verify(token, secretKey);
      const isRevoked = await this.isTokenRevoked(decoded.jti);
      if (isRevoked) {
        throw new Error('Token has been revoked');
      }
      return decoded;
    } catch (err) {
      console.error('Token verification failed:', err.message);
      return null;
    }
  },
};
```

Рис. 1. Хранение списка отозванных токенов

Хранение JWT у клиента должно учитывать угрозы, связанные с атаками типа XSS и CSRF. Для обеспечения безопасности токен рекомендуется хранить в Cookies с атрибутами HttpOnly и Secure, что предотвращает доступ к токenu через JavaScript и обеспечивает его передачу только через защищённые соединения (HTTPS) (рис.2).

```
// Роут для входа (авторизации пользователя)
app.post('/login', (req, res) => {
  const { username, password } = req.body;
  const user = getUser(username);
  if (!user || !bcrypt.compareSync(password, user.password)) {
    res.status(401).json({ success: false, message: 'Invalid username or password' });
    return;
  }
  const token = generateToken({ username });

  // Сохраняем токен в cookie
  res.cookie('jwt', token, {
    httpOnly: true, // Доступ только для сервера
    secure: true, // Токен передается только по HTTPS (включите на продакшене)
    sameSite: 'strict', // Ограничивает доступ к cookie для CSRF-защиты
    maxAge: 3600000, // Время жизни cookie: 1 час (в миллисекундах)
  });

  res.json({ message: 'Logged in successfully!' });
});
```

Рис. 2. Сохранение токена в Cookie

Дополнительно следует минимизировать срок действия токена, задавая ограничение по времени с помощью параметра `exp`, что снижает риск его использования в случае компрометации. Тогда, чтобы пользователям не нужно было постоянно авторизоваться, нужно добавить Refresh-токен. Refresh-токены применяются для продления срока действия JWT, что позволяет минимизировать необходимость долгосрочной валидности токенов доступа. Refresh-токены должны иметь более длительный срок действия, но их использование должно быть ограничено строго определёнными условиями, например, запросами на сервер аутентификации.

Также важным аспектом безопасности является регулярная ротация ключей, что снижает вероятность компрометации системы. Периодическая замена ключей должна выполняться с использованием заранее продуманного механизма, например, через выпуск новых ключей и одновременную поддержку старых ключей для проверки уже выпущенных токенов в течение переходного периода. Это позволяет избежать отказа в обслуживании из-за недействительности токенов, выпущенных до обновления ключей. Выбор подхода зависит от требований системы к производительности, безопасности и масштабируемости. Предоставление публичного ключа всем сервисам рекомендуется для систем с высокой нагрузкой, централизованная проверка токенов подходит для упрощения управления и контроля, а хранение ключей в базе данных может быть полезно в средах с ограниченным числом сервисов.

4 Выводы

В результате исследования были рассмотрены основные требования к безопасному использованию JWT авторизации. Приведены примеры возможных уязвимостей в реализации библиотек и представлены виды атак злоумышленников.

Библиографический список

1. Шамян Г.Г. Сравнительный анализ методологий авторизации JWT и

- sessions // Студенческий вестник. 2022. № 19-16 (211). С. 15-17.
2. Девыцына С.Н., Пилькевич П.В., Удод Е.В. Способы улучшения защищённости сервисов, использующих JWT-токены // Экономика. Информатика. 2023. Т. 50. № 1. С. 144-151.
 3. Бабичева М.В. Атаки на jwt токены // В сборнике: Донецкие чтения 2022: образование, наука, инновации, культура и вызовы современности. Материалы VII Международной научной конференции, посвящённой 85-летию Донецкого национального университета. Под общей редакцией С.В. Беспаловой. Донецк, 2022. С. 176-178.
 4. 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.
 5. 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.
 6. 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.
 7. Макаров Д.А. Механизм авторизации с использованием технологии JWT // Теория и практика современной науки. 2020. № 1 (55). С. 474-476.
 8. Голубь И.С. Приложение .NET core web api с использованием JWT-токенов для авторизации // Постулат. 2020. № 1 (51). С. 91.