

Реализация и использование JSON Web Token для безопасной передачи информации

Вихляев Дмитрий Романович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье рассматривается внутренняя архитектура JSON Web Token и методика применения в системе авторизации. Реализации методов написаны на языке программирования JavaScript интерпретатора Nodejs. Результатом исследования станет описание создания и использования JSON Web Token на сервере.

Ключевые слова: JWT, авторизация, проверка подлинности.

Implementation and use of JSON Web Token for secure information transfer

Vikhlyayev Dmitry Romanovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article discusses the internal architecture of the JSON Web Token and the methodology of its application in the authorization system. The method implementations are written in the JavaScript programming language of the Node js interpreter. The result of the study will be a description of the creation and use of the JSON Web Token on the server.

Keywords: JWT, authorization, authentication.

1 Введение

1.1 Актуальность

Актуальность применения JWT обусловлена растущей потребностью в эффективных и безопасных методах передачи информации в распределенных системах, таких как веб-приложения, мобильные приложения и API. В условиях современной цифровой трансформации, характеризующейся увеличением объема обрабатываемых данных и числа подключенных устройств, разработка решений, обеспечивающих целостность, конфиденциальность и надежную аутентификацию, становится ключевым требованием. Важным аспектом является гибкость и универсальность JWT, позволяющая использовать его в различных сценариях, включая аутентификацию, авторизацию и обмен данными между сервисами. Токены легко интегрируются в существующие системы, обеспечивая поддержку множества стандартов безопасности, таких как OAuth 2.0 и OpenID Connect.

Вдобавок, благодаря статической структуре токены могут быть обработаны без обращения к серверу, что минимизирует задержки и снижает нагрузку на серверную инфраструктуру.

1.2 Обзор исследований

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 описал идентификацию пользователей клиент-серверных приложений с помощью JWT-токена [5]. Л.Н.Абдурайимов реализовал аутентификацию в REST-приложениях с использованием spring security и JWT [6].

1.3 Цель исследования

Цель исследования – описать процессы создания и применения JSON Web Token в системах авторизации.

2 Материалы и методы

Для работы используется формулы и блок схемы, представляющие теоритическую информацию, а также фрагменты программного кода для реализации методов.

3 Результаты и обсуждения

JSON Web Token (JWT) представляет собой стандарт для создания токенов, используемых для передачи информации между сторонами в зашифрованном и защищённом виде. JWT состоит из трёх частей: заголовка (header), полезной нагрузки (payload) и подписи (signature), объединённых точками. Каждая часть кодируется алгоритма Base64URL (eBase64), являющейся вариацией стандарта кодирования Base64, используемого для передачи в сети интернет внутри URL адреса.

$$JWT = eBase64(header).eBase64(payload).eBase64(signature)$$

Заголовок токена включает два параметра: тип токена (typ), который обычно равен JWT, и используемый алгоритм подписи (alg), например, HS256 для HMAC с использованием SHA-256 или RS256 для RSA с SHA-256. Заголовок JWT может включать дополнительные параметры, зависящие от конкретного использования.

Полный список поддерживаемых алгоритмов включает симметричные алгоритмы, такие как HS256, HS384, HS512, асимметричные алгоритмы RS256, RS384, RS512, PS256, PS384, PS512, а также алгоритмы эллиптической кривой ES256, ES384, ES512. Также существует алгоритм без подписи (none), который может использоваться только в специфических случаях и требует

особой осторожности. Существует метод взлома JWT-токена, в котором применяется алгоритм без подписи с целью узнать секретный ключ. Такая лазейка в защите может быть в библиотеках, предоставляющих готовые функции работы JWT, но с плохо реализованной логикой реализации безопасности.

Полезная нагрузка токена содержит утверждения (claims), которые разделяются на зарегистрированные, общие и частные. Зарегистрированные утверждения включают стандартные поля, такие как `iss` (issuer, идентификатор эмитента), `sub` (subject, идентификатор пользователя), `aud` (audience, идентификатор получателя), `exp` (expiration time, время истечения срока действия), `nbf` (not before, время начала действия токена), `iat` (issued at, время выпуска токена) и `jti` (JWT ID, уникальный идентификатор токена). Общие утверждения могут включать дополнительные данные, используемые в приложении, например, роли пользователя или идентификаторы. Частные утверждения добавляются пользователями для специфических целей и не стандартизированы. Например, можно хранить поле `ip` содержащее `ip`-адрес пользователя, что повышает безопасность системы для пользователей со статическим `ip`-адресом (рис.1).

Algorithm

HS256

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6I1Zpa2EiLCJlbWFnZCI6InZrMTU2QzQ1haWwucnUiLCJpYXQiOiJlE3MzY1NjIwNDg3NDksImV4cCI6MTczNjU2Mjk0ODc0X000.1EKN2uPFhHr_YyHD4M-FTsPXqvH0ggHIBslqCCBOUBM
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  "alg": "HS256",  "typ": "JWT"}
```

PAYLOAD: DATA

```
{  "username": "Vika",  "email": "vk156@mail.ru",  "iat": 1736562048749,  "exp": 1736562948749}
```

VERIFY SIGNATURE

```
HMACSHA256(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),  secret_key  )
```

secret_key

☒ secret base64 encoded

Рис. 1. Содержимое JWT-токена

Процесс генерации JWT состоит из следующих шагов. Сначала формируется заголовок, в котором указывается используемый алгоритм и тип токена. Затем создаётся полезная нагрузка с необходимыми утверждениями, такими как идентификатор пользователя и время действия токена. После этого заголовок и полезная нагрузка кодируются с использованием алгоритма `Base64Url`, и создаётся подпись на основе выбранного алгоритма с использованием секретного ключа или пары ключей. Итоговый токен

формируется путём объединения закодированного заголовка, полезной нагрузки и подписи через точки (рис.2).

```
// Конфигурация
const SECRET_KEY = 'your-secret-key';
const TOKEN_EXPIRATION = 15 * 60 * 1000; // 15 минут в миллисекундах

// Функция для кодирования в Base64URL
function base64urlEncode(data) {
  return Buffer.from(JSON.stringify(data))
    .toString('base64')
    .replace(/=/g, '')
    .replace(/\+/g, '-')
    .replace(/\//g, '_');
}

// Функция для создания подписи
function createSignature(header, payload, secret) {
  const data = `${header}.${payload}`;
  return crypto.createHmac('sha256', secret).update(data).digest('base64url');
}

// Генерация токена
function generateToken(username) {
  const header = base64urlEncode({ alg: 'HS256', typ: 'JWT' });
  const payload = base64urlEncode({
    username,
    iat: Date.now(),
    exp: Date.now() + TOKEN_EXPIRATION,
  });
  const signature = createSignature(header, payload, SECRET_KEY);
  return `${header}.${payload}.${signature}`;
}
```

Рис. 2. Реализация программы для генерации JWT-токена

Подпись формируется путём шифрования двух строк. Первой строкой является часть итогового токена, закодированные заголовок и полезная нагрузка. Второй строкой является секретный ключ, который должен храниться в безопасном месте на сервере или в базе данных.

$$signature = Hash(eBase64(header).eBase64(payload), SECRET_KEY))$$

Процесс проверки JWT включает в себя верификацию подписи, декодирование и проверку полезной нагрузки. Сначала токен разделяется на три части. Затем, используя указанный в заголовке алгоритм и доступный ключ (секретный для симметричных алгоритмов или публичный для асимметричных), вычисляется подпись и сравнивается с предоставленной в токене. Если подпись валидна, декодируется полезная нагрузка. После этого проверяются временные утверждения, такие как exp, nbf и iat, чтобы убедиться, что токен не истёк, ещё не действителен или не скомпрометирован. Также проверяются утверждения о получателе (aud) и эмитенте (iss), если они используются. В случае успешного выполнения всех проверок токен считается валидным, и данные из полезной нагрузки могут быть использованы (рис.3,4).

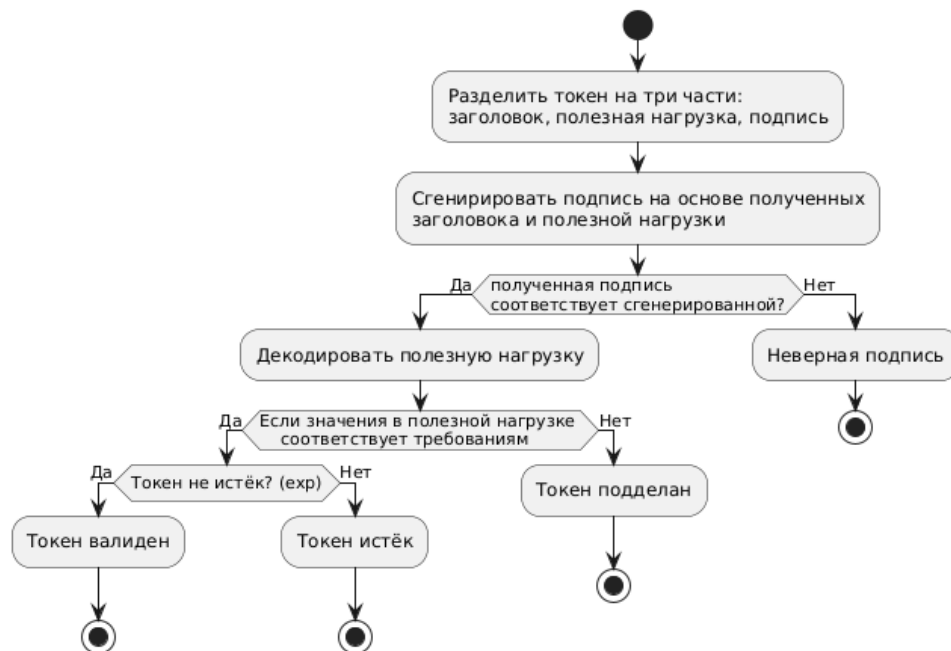


Рис. 3. Блок схема результатов проверки токена

```

// Функция для создания подписи
function createSignature(header, payload, secret) {
  const data = `${header}.${payload}`;
  return crypto.createHmac('sha256', secret).update(data).digest('base64url');
}

// Проверка токена
function verifyToken(token) {
  const [header, payload, signature] = token.split('.');
  if (!header || !payload || !signature) {
    return { valid: false, error: 'Malformed token' };
  }

  const expectedSignature = createSignature(header, payload, SECRET_KEY);
  if (expectedSignature !== signature) {
    return { valid: false, error: 'Invalid signature' };
  }

  const decodedPayload = JSON.parse(Buffer.from(payload, 'base64').toString('utf-8'));
  if (decodedPayload.exp < Date.now()) {
    return { valid: false, error: 'Token expired' };
  }

  return { valid: true, payload: decodedPayload };
}

```

Рис. 4. Реализация программы для проверки JWT-токена

4 Выводы

Таким образом, была рассмотрена внутренняя организация JSON Web Token. В результате исследования разобраны алгоритмы генерации и проверки токена. Представлена программная реализация создания и вариации токена. Описаны поддерживаемые методы шифрования, и параметры для заголовка и полезной нагрузки.

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.