

Способы хранения ключей шифрования JWT

Вихляев Дмитрий Романович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье приводятся примеры способов хранения ключей шифрования токенов в распределённых системах. Описание примеров приводится с помощью схем. В результате исследования будут представлены преимущества и особенности каждого способа, а также сравнение использования симметричного и асимметричного ключа шифрования.

Ключевые слова: JWT, ключ шифрования, распределённая система.

Ways to store JWT encryption keys

Vikhlyaev Dmitry Romanovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article provides examples of ways to store token encryption keys in distributed systems. The examples are described using diagrams. As a result of the research, the advantages and features of each method will be presented, as well as a comparison of the use of symmetric and asymmetric encryption keys.

Keywords: JWT, encryption key, distributed system.

1 Введение

1.1 Актуальность

Обеспечение надежного хранения ключей шифрования JSON Web Token (JWT) является важной частью защиты систем аутентификации и авторизации от угроз, связанных с компрометацией ключей. В условиях активного использования JWT для передачи данных между клиентом и сервером безопасность хранения ключей является критическим фактором, определяющим общую устойчивость системы к атакам. Учитывая рост числа распределенных систем и сервисов, ключи шифрования становятся основной целью злоумышленников, так как их компрометация может привести к полному обходу механизмов аутентификации и авторизации. Эффективные методы хранения включают использование аппаратных и программных хранилищ, ротацию ключей, применение современных алгоритмов шифрования и строгие политики управления доступом. Комплексный подход к защите ключей способствует минимизации рисков компрометации и повышает общую устойчивость системы к внешним угрозам.

1.2 Обзор исследований

А.Д.Жестков провёл анализ угроз и уязвимостей при хранении закрытого ключа [1]. К.А.Бирюков сравнил средства безопасного хранения ключей [2]. С.С.Агафьин представил функции безопасности внешних аппаратных модулей, используемых для хранения криптографических ключей [3]. В.П.Гергель, Д.С.Скатов реализовали эффективную структуру хранения словаря строковых ключей с ассоциированными значениями [4]. В.В.Банит описал способ хранения закрытого ключа криптосистемы цифровой подписи [5].

1.3 Цель исследования

Цель исследования – привести примеры способов хранения ключей шифрования для токенов авторизации с целью повышения защиты системы.

2 Материалы и методы

Для описания процессов работы применяются графические схемы, каждый метод поясняется механизмом реализации и преимуществом безопасности.

3 Результаты и обсуждения

Для обеспечения безопасной работы с JWT в распределённой системе используется сервер аутентификации, который отвечает за генерацию и проверку токенов. Процесс взаимодействия начинается с этапа аутентификации клиента. Клиент, представляющий собой приложение или веб-интерфейс, отправляет запрос на сервер аутентификации с предоставлением учётных данных, таких как логин и пароль, либо с использованием альтернативных методов, включая OAuth или OpenID Connect. Сервер аутентификации проверяет корректность предоставленных данных, используя внутренние механизмы верификации, такие как сверка с базой данных пользователей, и в случае успешной проверки генерирует JWT, содержащий утверждения (claims), определяющие идентификационные данные пользователя, роли и срок действия токена. JWT подписывается сервером аутентификации с использованием заранее определённого ключа, который может быть симметричным или асимметричным в зависимости от выбранного алгоритма. Полученный токен передаётся клиенту в ответе, после чего клиент сохраняет его, например, в cookie или локальном хранилище, для последующего использования.

На этапе взаимодействия клиента с сервером приложения JWT используется для аутентификации запросов. Клиент передаёт токен в заголовке HTTP-запроса Authorization, используя схему Bearer, что позволяет серверу приложения идентифицировать пользователя без необходимости повторного запроса учётных данных. Сервер приложения, получив токен, извлекает его и передаёт для проверки на сервер аутентификации либо, при использовании асимметричной подписи, выполняет проверку локально, используя публичный ключ, полученный из доверенного источника. Процесс

проверки токена включает в себя проверку подписи с использованием соответствующего ключа, декодирование содержимого и проверку временных утверждений, таких как `exp` (время истечения токена) и `nbf` (время начала действия). Если токен валиден, сервер приложения извлекает данные из полезной нагрузки и использует их для предоставления доступа к защищённым ресурсам или выполнения запрашиваемой операции (рис.1).

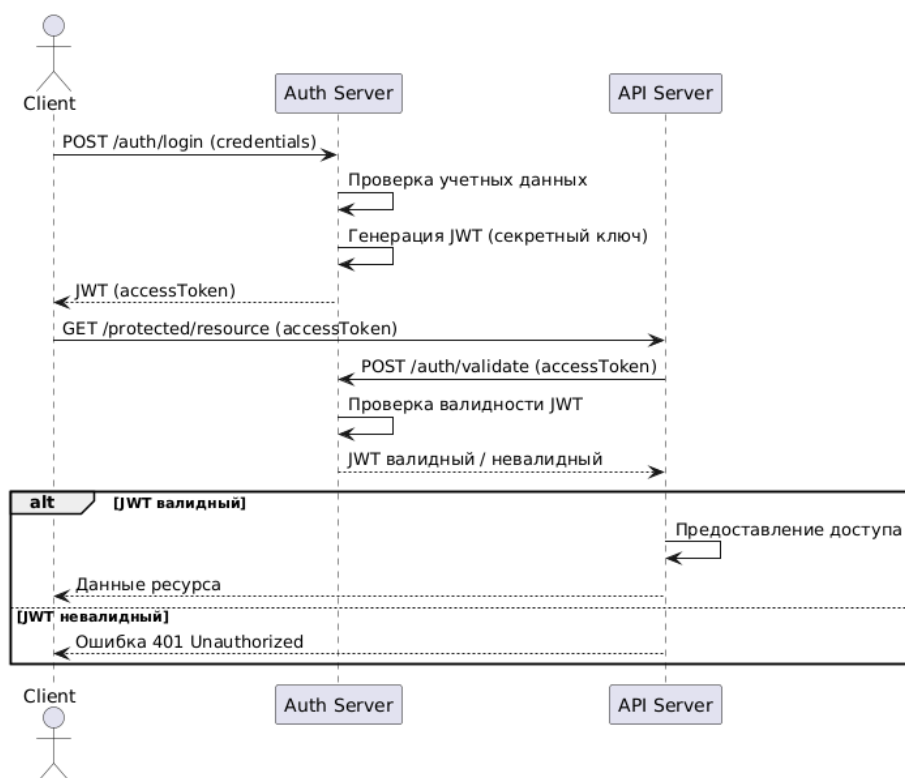


Рис. 1. Последовательность функций получение данных с api сервера, при хранении ключа на auth-сервере

Использование симметричного ключа для шифрования навязывает проблему передачи, данного ключа по не защищённому каналу передачи. Другим вариантом является использование асимметричного ключа.

Использование асимметричного ключа для работы с JWT в распределённой системе предполагает несколько подходов к организации проверки токенов. В первом подходе сервер аутентификации генерирует пару ключей, где приватный ключ используется для подписи токенов, а публичный ключ для их проверки. Централизованное хранение ключей на сервере аутентификации и предоставление метода верификации токенов другим сервисам через API. В этом случае серверы приложений передают токен в запросе на сервер аутентификации, который проверяет его подпись и возвращает результат проверки или декодированное содержимое токена. Этот подход снижает вероятность ошибки при обработке токенов и упрощает ротацию ключей, поскольку все операции выполняются в одном месте. Данный метод также подойдёт при использовании симметричного шифрования. Однако централизованная проверка токенов увеличивает

нагрузку на сервер аутентификации и задержки в обработке запросов из-за сетевых взаимодействий.

Во втором подходе сервер аутентификации передаёт публичный ключ всем другим сервисам для их проверки. Этот метод обеспечивает децентрализацию процесса верификации, поскольку каждый сервис может локально проверять подпись токена, используя публичный ключ. Данный подход снижает задержки и нагрузку на сервер аутентификации, поскольку исключает необходимость удалённых запросов для проверки токенов. В случае симметричного шифрования, необходимо надёжное распространение секретного ключа. Например, с использованием защищённых каналов или статически размещённого ключа в доверенном хранилище, а также реализация механизма ротации ключей и их обновления на всех сервисах. В случае асимметричного шифрования, нечего страшного, если публичный ключ будет перехвачен, так как он может только расшифровать токен, но не подделать (рис.2).

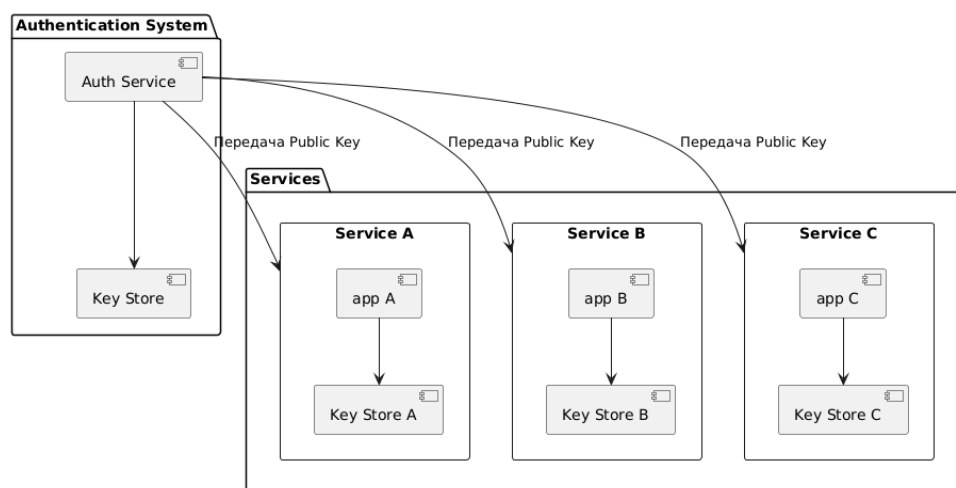


Рис. 2. Хранение ключей в каждом сервисе

Третий подход предполагает хранение пары ключей в базе данных, доступной всем сервисам. В этом случае сервисы получают доступ к ключам через запросы к базе данных и выполняют локальную проверку токенов. Преимущество такого подхода заключается в возможности централизованного управления ключами и их обновлением, однако он требует строгой защиты базы данных от несанкционированного доступа. 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 (рис.3).

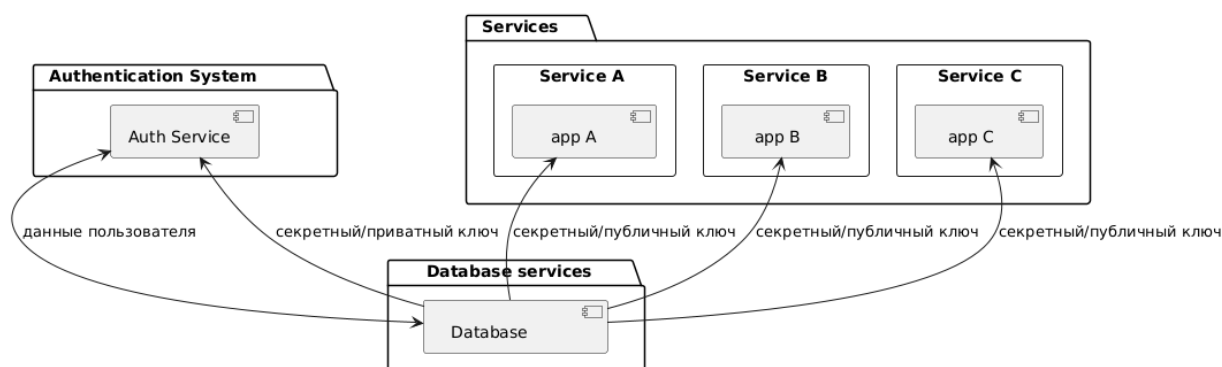


Рис. 3. Хранение ключей в общей базе данных

4 Выводы

В результате данной статьи, были описаны способы организации хранения и передачи ключей шифрования JWT-токена во много сервисной системе. Рассмотрены вариации использования симметричного и асимметричного ключа.

Библиографический список

1. Жестков А.Д. Анализ угроз и уязвимостей при хранении закрытого ключа // Научный аспект. 2024. Т. 17. № 5. С. 2287-2293.
2. Бирюков К.А. Средства безопасного хранения ключей // Безопасность информационных технологий. 2013. Т. 20. № 3. С. 50-53.
3. Агафьин С.С. Функции безопасности внешних аппаратных модулей, используемых для хранения криптографических ключей // Безопасность информационных технологий. 2014. Т. 21. № 3. С. 37-43.
4. Гергель В.П., Скотов Д.С. Эффективная структура хранения словаря строковых ключей с ассоциированными значениями // Вестник Нижегородского университета им. Н.И. Лобачевского. 2012. № 5-2. С. 301-308.
5. Банит В.В. Способ хранения закрытого ключа криптосистемы цифровой подписи // Молодой ученый. 2014. № 11. С. 19-22.
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.
9. 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.