Лекция 10
Авторизация в Web-приложениях

Мартин Комитски

образование

План на сегодня

- Основные понятия
- AAA
- Аутентификация по паролю
  - HTTP authentication
  - Forms authentication
  - Другие протоколы аутентификации по паролю
- Аутентификация по сертификатам
- Аутентификация по одноразовым паролям
- Аутентификация по ключам доступа
- Аутентификация по QR коду
- Аутентификация по токенам
  - SWT
  - JWT
  - SAML
- OAuth и OpenID Connect

Минутка бюрократии

- Внимание
- Отметки о посещении занятий
- Обратная связь о лекциях

Основные понятия

Основные понятия

Идентификация — ?

Аутентификация — ?

Авторизация — ?

Учёт — ?

Основные понятия

Идентификация (Identification) — это заявление о том, кем вы являетесь. В зависимости от ситуации, это может быть имя, адрес электронной почты, номер учетной записи, и т.д.

Аутентификация (Authentication) — предоставление доказательств, что вы на самом деле есть тот, кем идентифицировались (от слова “authentic” — истинный, подлинный).

Авторизация (Authorization) — проверка, что вам разрешен доступ к запрашиваемому ресурсу.

Учёт (Accounting) — слежение за потреблением ресурсов (преимущественно сетевых) пользователем. В учёт включается также и запись фактов получения доступа к системе (access logs).

AAA

AAA

AAA определяет архитектуру, которая аутентифицирует и предоставляет авторизацию пользователям и учетным записям для их действий.

Когда AAA не используется, сетевая архитектура является «открытой», где любой может получить доступ и делать что угодно без какого-либо отслеживания.

Открытая сетевая архитектура обычно используется на малых предприятиях, где доступ в офис можно контролировать физически.

AAA можно внедрять частично.

Корпоративные решения

- TACACS+
- RADIUS
- Diameter

Аутентификация по паролю

Аутентификация по паролю

- Пользователь должен предоставить username и password для успешной идентификации и аутентификации в системе
- Пара username/password задается пользователем при его регистрации в системе
- В качестве username может выступать адрес электронной почты пользователя
- Существует несколько стандартных протоколов для аутентификации по паролю

HTTP authentication

HTTP authentication

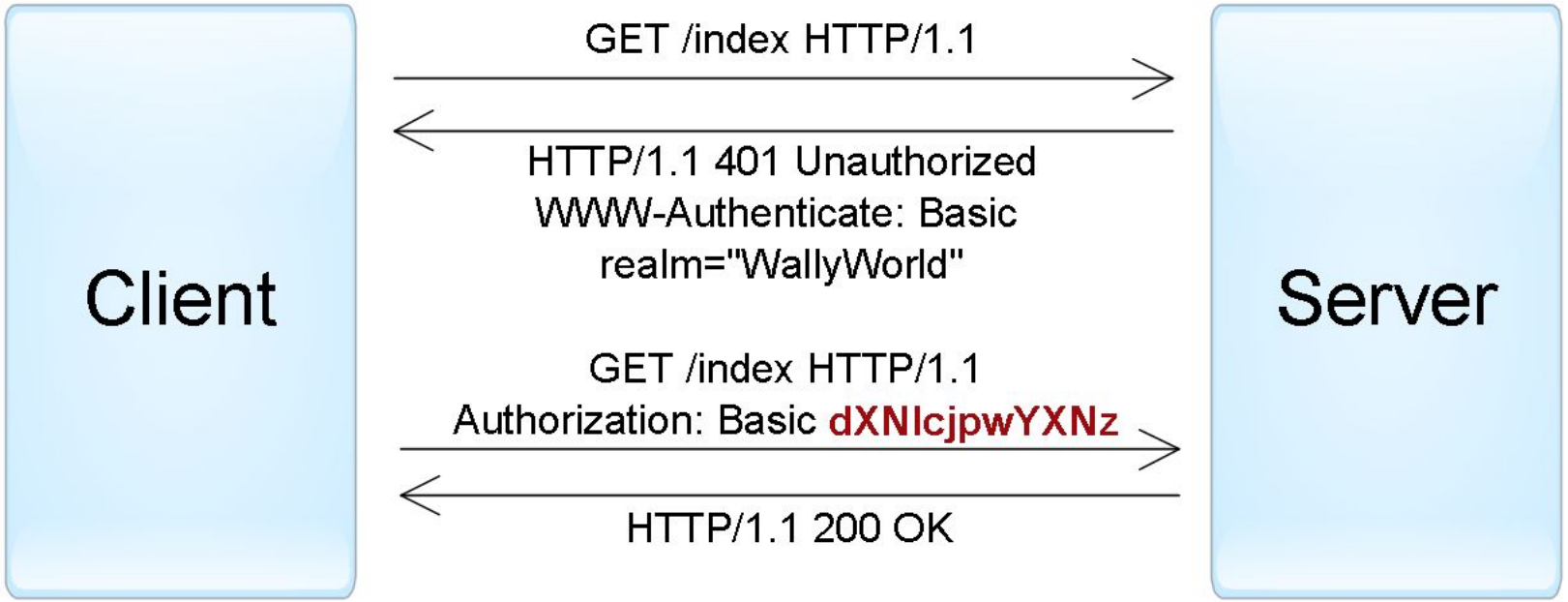
- Описан в стандартах HTTP 1.0/1.1 (существует очень давно)
- Активно применяется в корпоративной среде
- У пользователя нет стандартной возможности выйти из веб-приложения, кроме как закрыть все окна браузера

Принцип работы:

- Сервер, при обращении неавторизованного клиента к защищенному ресурсу, отсылает HTTP статус “401 Unauthorized” и добавляет заголовок “WWW-Authenticate” с указанием схемы и параметров аутентификации
- Браузер, при получении такого ответа, автоматически показывает диалог ввода username и password
- Пользователь вводит детали своей учетной записи
- Во всех последующих запросах к этому веб-сайту браузер автоматически добавляет HTTP заголовок “Authorization”, в котором передаются данные пользователя для аутентификации сервером
- Сервер аутентифицирует пользователя по данным из этого заголовка. Решение о предоставлении доступа (авторизация) производится отдельно на основании роли пользователя, ACL или других данных учетной записи.

HTTP authentication. Basic

- Наиболее простая схема: username и password пользователя передаются в заголовке Authorization в незашифрованном виде (base64-encoded)
- При использовании HTTPS протокола, является относительно безопасной
- Можно передавать прямо в URL: https://username:password@URL — отправит тот же заголовок



HTTP authentication. Digest

- challenge-response схема, при которой сервер посылает уникальное значение nonce
- Браузер передает MD5 хэш пароля пользователя, вычисленный с использованием указанного nonce
- Более безопасная альтернатива Basic схемы при незащищенных соединениях
- Подвержена man-in-the-middle attacks (с заменой схемы на basic)
- Использование этой схемы не позволяет применить современные хэш-функции для хранения паролей пользователей на сервере

https://stackoverflow.com/questions/5288150/is-digest-authentication-possible-with-jquery/5288679#5288679

https://stackoverflow.com/questions/2384230/what-is-digest-authentication

HTTP authentication. NTLM (Windows authentication)

- Основана на challenge-response подходе, при котором пароль не передается в чистом виде
- Не является стандартом HTTP, но поддерживается большинством браузеров и веб-серверов
- Преимущественно используется для аутентификации пользователей Windows Active Directory в веб-приложениях
- Уязвима к pass-the-hash атакам

HTTP authentication. Negotiate

- Ещё одна схема из семейства Windows authentication, которая позволяет клиенту выбрать между NTLM и Kerberos аутентификацией
- Kerberos — более безопасный протокол, основанный на принципе Single Sign-On
- Может функционировать, только если и клиент, и сервер находятся в зоне intranet и являются частью домена Windows

Forms authentication

Forms authentication

- Нет определенного стандарта, поэтому все его реализации специфичны для конкретных систем

Работает по принципу:

- В веб-приложение включается HTML-форма, в которую пользователь должен ввести свои username/password
- Отправить их на сервер через HTTP POST для аутентификации
- В случае успеха веб-приложение создает session token, который обычно помещается в browser cookies
- При последующих веб-запросах session token автоматически передается на сервер и позволяет приложению получить информацию о текущем пользователе для авторизации запроса

Forms authentication

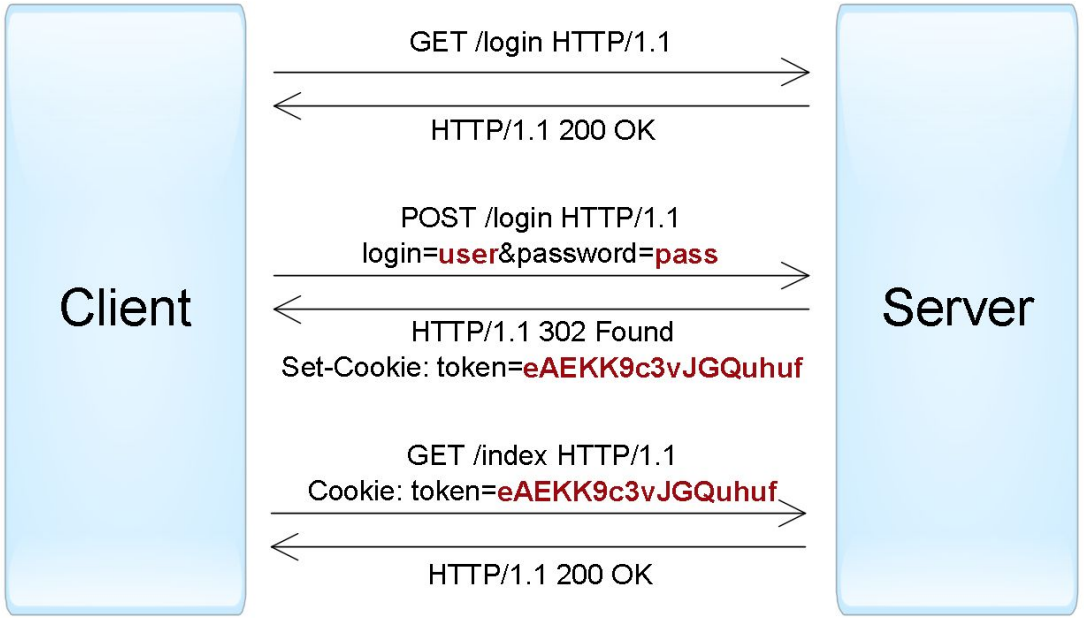
Приложение может создать session token двумя способами:

1. Как идентификатор аутентифицированной сессии пользователя, которая хранится в памяти сервера или в базе данных.
   Сессия должна содержать всю необходимую информацию о пользователе для возможности авторизации его запросов
2. Как зашифрованный и/или подписанный объект, содержащий данные о пользователе, а также период действия.
   Этот подход позволяет реализовать stateless-архитектуру сервера, однако требует механизма обновления сессионного токена по истечении срока действия.
   Несколько форматов таких токенов рассмотрим дальше

Forms authentication

Необходимо понимать, что перехват session token зачастую дает аналогичный уровень доступа, что и знание username/password.

Поэтому все коммуникации между клиентом и сервером в случае forms authentication должны производиться только по защищенному соединению HTTPS.



Другие протоколы аутентификации по паролю

Другие протоколы аутентификации по паролю

При разработке клиент-серверных приложений с использованием веб-сервисов (например, iOS или Android), наряду с HTTP аутентификацией, часто применяются нестандартные протоколы, в которых данные для аутентификации передаются в других частях запроса.

Существует всего несколько мест, где можно передать username и password в HTTP запросах:

- URL query — считается небезопасным вариантом, т. к. строки URL могут запоминаться браузерами, прокси и веб-серверами
- Request body — безопасный вариант, но он применим только для запросов, содержащих тело сообщения (такие как POST, PUT, PATCH)
- HTTP header — оптимальный вариант, при этом могут использоваться и стандартный заголовок Authorization (например, с Basic-схемой), и другие произвольные заголовки

Аутентификация по сертификатам

Аутентификация по сертификатам

Сертификат представляет собой набор атрибутов, идентифицирующих владельца, подписанный certificate authority (CA).

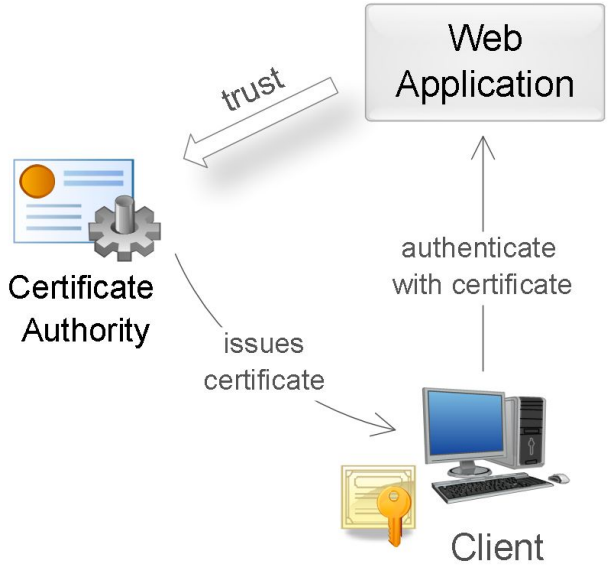
CA выступает в роли посредника, который гарантирует подлинность сертификатов.

Также сертификат криптографически связан с закрытым ключом, который хранится у владельца сертификата и позволяет однозначно подтвердить факт владения сертификатом.

Аутентификация по сертификатам

На стороне клиента сертификат вместе с закрытым ключом могут храниться в операционной системе, в браузере, в файле, на отдельном физическом устройстве (smart card, USB token). Закрытый ключ дополнительно защищён паролем или PIN-кодом.

В веб-приложениях используют сертификаты стандарта X.509. Аутентификация с помощью X.509-сертификата происходит в момент соединения с сервером и является частью протокола SSL/TLS.
Этот механизм также хорошо поддерживается браузерами, которые позволяют пользователю выбрать и применить сертификат, если веб-сайт допускает такой способ аутентификации.



Аутентификация по сертификатам

Во время аутентификации сервер выполняет проверку сертификата на основании следующих правил:

- Сертификат должен быть подписан доверенным Certification Authority (проверка цепочки сертификатов)
- Сертификат должен быть действительным на текущую дату (проверка срока действия)
- Сертификат не должен быть отозван соответствующим CA (проверка списков исключения)

Аутентификация по сертификатам

После успешной аутентификации веб-приложение может выполнить авторизацию запроса на основании таких данных сертификата, как *subject* (имя владельца), *issuer* (эмитент), *serial number* (серийный номер сертификата) или *thumbprint* (отпечаток открытого ключа сертификата).

Использование сертификатов для аутентификации — куда более надёжный способ, чем аутентификация посредством паролей. Это достигается созданием в процессе аутентификации цифровой подписи, наличие которой доказывает факт применения закрытого ключа в конкретной ситуации (non-repudiation). Однако трудности с распространением и поддержкой сертификатов делает такой способ аутентификации малодоступным в широких кругах.

Аутентификация по одноразовым паролям

Аутентификация по одноразовым паролям

Аутентификация по одноразовым паролям обычно применяется дополнительно к аутентификации по паролям для реализации two-factor authentication (2FA).

Пользователю необходимо предоставить данные двух типов для входа в систему:

- Что-то, что он знает (например, пароль)
- Что-то, чем он владеет (например, устройство для генерации одноразовых паролей)

Наличие двух факторов позволяет в значительной степени увеличить уровень безопасности.

Другой сценарий использования одноразовых паролей — дополнительная аутентификация пользователя во время выполнения важных действий: перевод денег, изменение настроек и пр.

Аутентификация по одноразовым паролям

Источники для создания одноразовых паролей:

- Аппаратные или программные токены, которые могут генерировать одноразовые пароли на основании секретного ключа, введенного в них, и текущего времени. Секретные ключи пользователей, являющиеся фактором владения, также хранятся на сервере, что позволяет выполнить проверку введенных одноразовых паролей. Пример аппаратной реализаций токенов — FEITIAN OTP c100; программной — приложение Google Authenticator
- Случайно генерируемые коды, передаваемые пользователю через SMS, мессенджеры, push-нотификации или другой канал связи. В этой ситуации фактор владения — телефон пользователя (точнее — SIM-карта, привязанная к определенному номеру, мессенджер, конкретное приложение)
- Распечатка или scratch card со списком заранее сформированных одноразовых паролей. Для каждого нового входа в систему требуется ввести новый одноразовый пароль с указанным номером

Аутентификация по одноразовым паролям

В веб-приложениях такой механизм аутентификации часто реализуется посредством расширения forms authentication: после первичной аутентификации по паролю, создается сессия пользователя, однако в контексте этой сессии пользователь не имеет доступа к приложению до тех пор, пока он не выполнит дополнительную аутентификацию по одноразовому паролю.

Аутентификация по ключам доступа

Аутентификация по ключам доступа

Чаще всего используется для аутентификации устройств, сервисов или других приложений при обращении к веб-сервисам. Здесь в качестве секрета применяются ключи доступа (access key, API key) — длинные уникальные строки, содержащие произвольный набор символов, по сути заменяющие собой комбинацию username/password.

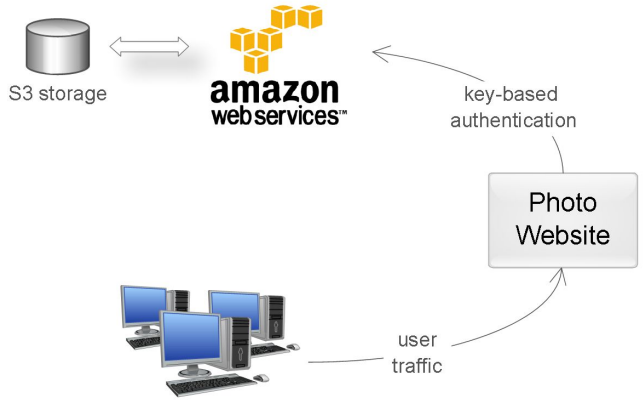
В большинстве случаев, сервер генерирует ключи доступа по запросу пользователей, которые далее сохраняют эти ключи в клиентских приложениях. При создании ключа также возможно ограничить срок действия и уровень доступа, который получит клиентское приложение при аутентификации с помощью этого ключа.

Хороший пример применения аутентификации по ключу — облако Amazon Web Services. Предположим, необходимо использовать сервис Amazon S3 для хранения файлов. В таком случае, через консоль AWS можно создать ключ, имеющий ограниченный доступ к облаку: только чтение/запись его файлов в Amazon S3. Этот ключ в результате можно применить для аутентификации веб-приложения в облаке AWS.

Аутентификация по ключам доступа

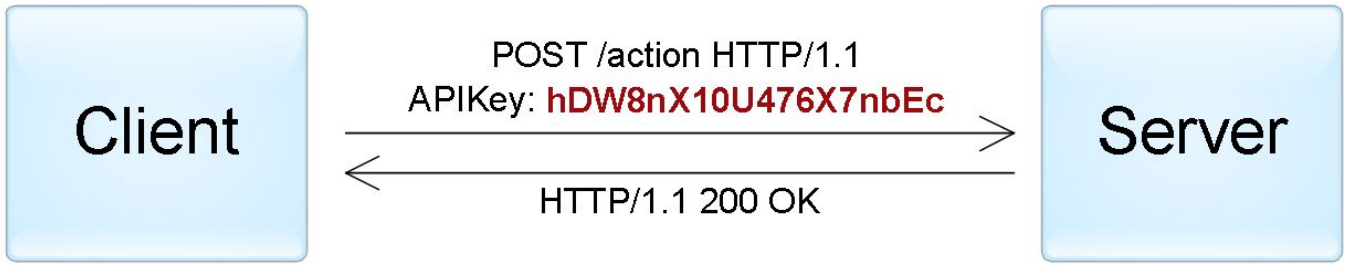
Использование ключей позволяет избежать передачи пароля пользователя сторонним приложениям (в примере выше пользователь сохранил в веб-приложении не свой пароль, а ключ доступа). Ключи обладают значительно большей энтропией по сравнению с паролями, поэтому их практически невозможно подобрать. Кроме того, если ключ был раскрыт, это не приводит к компрометации основной учетной записи пользователя — достаточно лишь аннулировать этот ключ и создать новый.

С технической точки зрения, здесь не существует единого протокола: ключи могут передаваться в разных частях HTTP-запроса: URL query, request body или HTTP header. Как и в случае аутентификации по паролю, наиболее оптимальный вариант — использование HTTP header. В некоторых случаях используют HTTP-схему Bearer для передачи токена в заголовке (Authorization: Bearer [token]). Чтобы избежать перехвата ключей, соединение с сервером должно быть обязательно защищено протоколом SSL/TLS.



Аутентификация по ключам доступа

Кроме того, существуют более сложные схемы аутентификации по ключам для незащищенных соединений. В этом случае, ключ обычно состоит их двух частей: публичной и секретной. Публичная часть используется для идентификации клиента, а секретная часть позволяет сгенерировать подпись. Например, по аналогии с digest authentication схемой, сервер может послать клиенту уникальное значение nonce или timestamp, а клиент — возвратить хэш или HMAC этого значения, вычисленный с использованием секретной части ключа. Это позволяет избежать передачи всего ключа в оригинальном виде и защищает от replay attacks.



Аутентификация по QR коду

Аутентификация по QR коду

Ещё одна из разновидностей аутентификации — с использованием QR кодов.

Принцип работы:

- Создаётся пара аутентификационный токен - токен ID
- Токен зашифровывается в изображение QR кода. Опционально, можно создать ссылку для упрощения доступа с некоторых мобильных устройств
- На странице логина опрашивается токен с использованием токен ID
- Как только QR код будет успешно прочитан, токен будет помечен как “получен”, а пользователя пропускает

Аутентификация по токенам

Аутентификация по токенам

Такой способ аутентификации чаще всего применяется при построении распределенных систем Single Sign-On (SSO), где одно приложение (*service provider* или *relying party*) делегирует функцию аутентификации пользователей другому приложению (*identity provider* или *authentication service*).

Типичный пример этого способа — вход в приложение через учетную запись в социальных сетях (*oauth*). Здесь социальные сети являются сервисами аутентификации, а приложение доверяет функцию аутентификации пользователей социальным сетям.

Единый вход (SSO) — это метод аутентификации, который позволяет пользователям безопасно проходить аутентификацию в нескольких приложениях и на веб-сайтах, используя только один набор учетных данных.

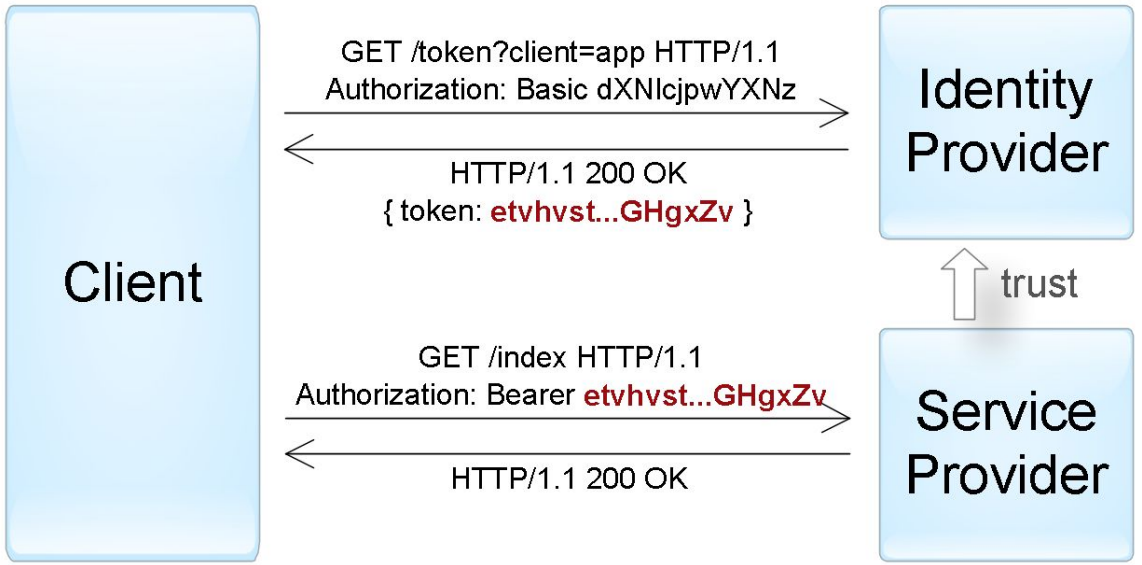
Реализация способа аутентификации по токенам заключается в том, что identity provider (IP) предоставляет достоверные сведения о пользователе в виде токена, а service provider (SP) приложение использует этот токен для идентификации, аутентификации и авторизации пользователя.

Важно понимать, что SSO отличается от oauth, несмотря на то, что в целом они немного похожи.

Аутентификация по токенам

На общем уровне, весь процесс выглядит следующим образом:

- Клиент аутентифицируется в identity provider одним из способов, специфичным для него (пароль, ключ доступа, сертификат, Kerberos, и т.д.)
- Клиент просит identity provider предоставить ему токен для конкретного SP-приложения. Identity provider генерирует токен и отправляет его клиенту
- Клиент аутентифицируется в SP-приложении при помощи этого токена

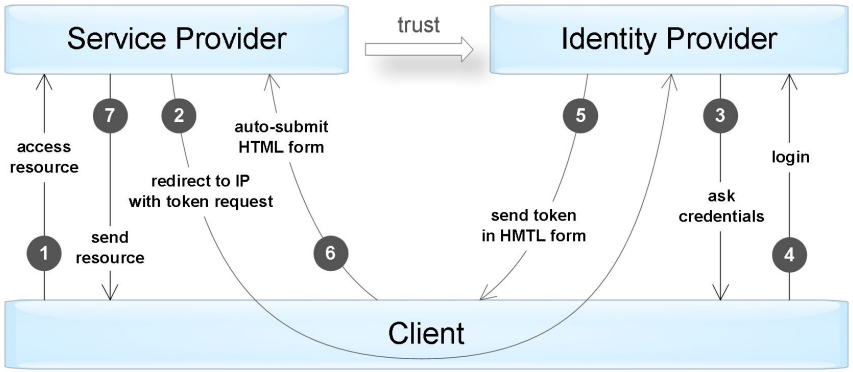


Аутентификация по токенам

Существует несколько стандартов, определяющих протокол взаимодействия между клиентами и IP/SP-приложениями и формат поддерживаемых токенов. Среди наиболее популярных стандартов — OAuth, OpenID Connect и SAML.

Сам токен обычно представляет собой структуру данных, которая содержит информацию:

- Кто сгенерировал токен
- Кто может быть получателем токена
- Срок действия
- Набор сведений о самом пользователе (claims)



Кроме того, токен дополнительно подписывается для предотвращения

несанкционированных изменений и гарантий подлинности.

При аутентификации с помощью токена SP-приложение должно выполнить следующие проверки:

- Токен был выдан доверенным identity provider приложением (проверка поля issuer)
- Токен предназначается текущему SP-приложению (проверка поля audience)
- Срок действия токена еще не истек (проверка поля expiration date)
- Токен подлинный и не был изменен (проверка подписи)

В случае успешной проверки SP-приложение выполняет авторизацию запроса на основании данных о пользователе, содержащихся в токене.

SWT

SWT

Simple Web Token (SWT) — наиболее простой формат, представляющий собой набор произвольных пар имя/значение в формате кодирования HTML form. Стандарт определяет несколько зарезервированных имен: Issuer, Audience, ExpiresOn и HMACSHA256. Токен подписывается с помощью симметричного ключа, таким образом оба IP- и SP-приложения должны иметь этот ключ для возможности создания/проверки токена.

Пример SWT токена (после декодирования):

1. Issuer=http://auth.myservice.com&
2. Audience=http://myservice.com&
3. ExpiresOn=1435937883&
4. UserName=John Smith&
5. UserRole=Admin&
6. HMACSHA256=KOUQRPSpy64rvT2KnYyQKtFFXUIggnesSpE7ADA4o9w

JWT

JWT

JSON Web Token (JWT) — содержит три блока, разделенных точками: заголовок, набор полей (claims) и подпись. Первые два блока представлены в JSON-формате и дополнительно закодированы в формат base64.
Набор полей содержит произвольные пары имя/значения, притом стандарт JWT определяет несколько зарезервированных имен (iss, aud, exp и другие). Подпись может генерироваться при помощи и симметричных алгоритмов шифрования, и асимметричных. Кроме того, существует отдельный стандарт, описывающий формат зашифрованного JWT-токена.

Пример подписанного JWT токена (после декодирования 1 и 2 блоков):

1. { «alg»: «HS256», «typ»: «JWT» }.
2. { «iss»: «auth.myservice.com», «aud»: «myservice.com», «exp»: «1435937883», «userName»: «John Smith», «userRole»: «Admin» }.
3. S9Zs/8/uEGGTVVtLggFTizCsMtwOJnRhjaQ2BMUQhcY

SAML

SAML

Security Assertion Markup Language (SAML) — определяет токены (SAML assertions) в XML-формате, включающем информацию об эмитенте, о субъекте, необходимые условия для проверки токена, набор дополнительных утверждений (statements) о пользователе.
Подпись SAML-токенов осуществляется при помощи ассиметричной криптографии. Кроме того, в отличие от предыдущих форматов, SAML-токены содержат механизм для подтверждения владения токеном, что позволяет предотвратить перехват токенов через man-in-the-middle атаки при использовании незащищенных соединений.

https://auth0.com/blog/how-saml-authentication-works/

OAuth и OpenID Connect

OAuth и OpenID Connect

OAuth определяет механизм получения доступа одного приложения к другому от имени пользователя. Существуют схемы, позволяющие осуществить аутентификацию пользователя на базе этого стандарта.

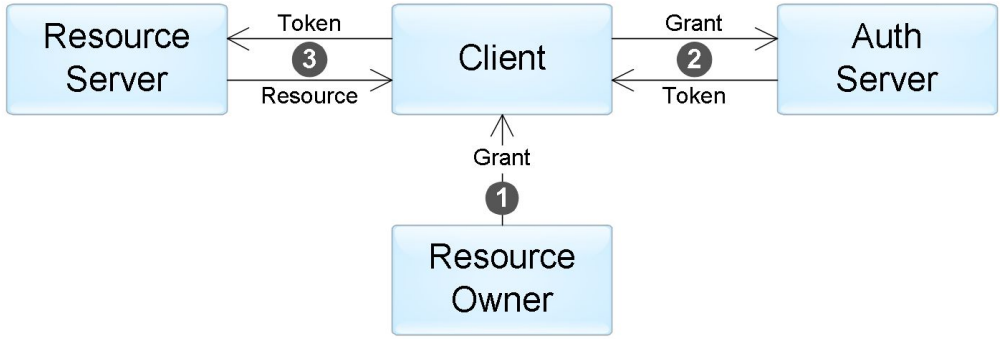
Первая версия стандарта разрабатывалась в 2007 – 2010 гг., а текущая версия 2.0 опубликована в 2012 г. Версия 2.0 значительно расширяет и в то же время упрощает стандарт, но обратно несовместима с версией 1.0. Сейчас OAuth 2.0 очень популярен и используется повсеместно для предоставления делегированного доступа и третье-сторонней аутентификации пользователей.

OAuth и OpenID Connect

Как веб-приложение может безопасно получить доступ к почте пользователей?

Как раз эту проблему и позволяет решить стандарт OAuth: он описывает, как приложение (client) может получить доступ к данным пользователя (resource server) с разрешения пользователя (resource owner). В общем виде весь процесс состоит из нескольких шагов:

1. Пользователь (resource owner) дает разрешение приложению (client) на доступ к определенному ресурсу в виде гранта
2. Приложение обращается к серверу авторизации и получает токен доступа к ресурсу в обмен на свой грант. Например, сервер авторизации — Google. При вызове приложение дополнительно аутентифицируется при помощи ключа доступа, выданным ему при предварительной регистрации
3. Приложение использует этот токен для получения требуемых данных от сервера ресурсов (в нашем случае — сервис Gmail)



OAuth и OpenID Connect

Стандарт описывает четыре вида грантов, которые определяют возможные сценарии применения:

- **Authorization Code** — этот грант пользователь может получить от сервера авторизации после успешной аутентификации и подтверждения согласия на предоставление доступа. Такой способ наиболее часто используется в веб-приложениях. Процесс получения гранта очень похож на механизм аутентификации пассивных клиентов в SAML
- **Implicit** — применяется, когда у приложения нет возможности безопасно получить токен от сервера авторизации (например, JavaScript-приложение в браузере). В этом случае грант представляет собой токен, полученный от сервера авторизации, а шаг № 2 исключается из сценария выше
- **Resource Owner Password Credentials** — грант представляет собой пару username/password пользователя. Может применяться, если приложение является «интерфейсом» для сервера ресурсов (например, приложение — мобильный клиент для Gmail)
- **Client Credentials** — в этом случае нет никакого пользователя, а приложение получает доступ к своим ресурсам при помощи своих ключей доступа (исключается шаг № 1)

OAuth и OpenID Connect

Стандарт не определяет формат токена, который получает приложение: в сценариях, адресуемых стандартом, приложению нет необходимости анализировать токен, т. к. он лишь используется для получения доступа к ресурсам. Поэтому ни токен, ни грант сами по себе не могут быть использованы для аутентификации пользователя. Однако если приложению необходимо получить достоверную информацию о пользователе, существуют несколько способов это сделать:

- Зачастую API сервера ресурсов включает операцию, предоставляющую информацию о самом пользователе. Приложение может выполнять эту операцию каждый раз после получения токена для идентификации клиента. Такой метод иногда называют псевдо-аутентификацией
- Использовать стандарт OpenID Connect, разработанный как слой учетных данных поверх OAuth. В соответствии с этим стандартом, сервер авторизации предоставляет дополнительный identity token на шаге № 2. Этот токен в формате JWT будет содержать набор определенных полей (claims) с информацией о пользователе

Стоит заметить, что OpenID Connect, заменивший предыдущие версии стандарта OpenID 1.0 и 2.0, также содержит набор необязательных дополнений для поиска серверов авторизации, динамической регистрации клиентов и управления сессией пользователя.

Авторизация в Web-приложениях?

Вопросы?

Полезные материалы

- Аутентификация в веб-приложениях
- Про токены, JSON Web Tokens (JWT), аутентификацию и авторизацию. Token-Based Authentication
- Обзор способов и протоколов аутентификации в веб-приложениях
- https://stormpath.com/blog/oauth-is-not-sso
- https://www.onelogin.com/learn/how-single-sign-on-works

Домашнее задание №10

1. Сверстать форму логина и интегрировать ее с backend
2. Подготовиться к квизу за модуль 2. Квиз будет на консультации

Расширенное описание задания, подсказки, а также презентации с лекций всегда есть в репозитории.

Срок сдачи

8 декабря

Спасибо за внимание!

образование

Пока!

Присоединяйтесь к сообществам про образование в VK

- VK Джуниор
- VK Образование

образование