AUTHENTIFIZIERUNG

INHALT

- Motivation
- Zugriffskontrolle
- Authenticationfactors
- Authenticationmethods
- Standards

Speaker notes

- Motivation
- Zugriffskontrolle
  - DAC / MAC
  - Identitybased
  - Rolebased
  - Attributebased
- Authenticationfactors
- Authenticationmethods
- Standards
  - SAML
  - OAuth2/JWT

MOTIVATION

ZUGRIFFSKONTROLLE

ZUGRIFFSTYP

DISCRETIONARY ACCESS CONTROL

- Benutzerzentriert
- Objektbezogen
- Typisch: Read, Write, Execute
- Lack of Competence

MANDATORY ACCESS CONTROL

- Systemweit
- Das System dominiert
- lack of overview

Speaker notes

DISCRETIONARY ACCESS CONTROL

Jeder Benutzer kann die Rechte für ein Objekt einzeln einstellen. common challenged by lack of competence, overview

MANDATORY ACCESS CONTROL

Das System gibt die Rechte vor, Nutzer kann sie zwar anpassen aber das System dominiert. commonly challenged by lack of overview, BOfH

IDENTITYBASED ACCESS CONTROL

- Zugriff wird anhand der Identät bestimmt
- Access Control Matrix

ROLEBASED ACCESS CONTROL

- Zugriff wird über eine Rolle gesteuert
- Access Control List

ATTRIBUTEBASED ACCESS CONTROL

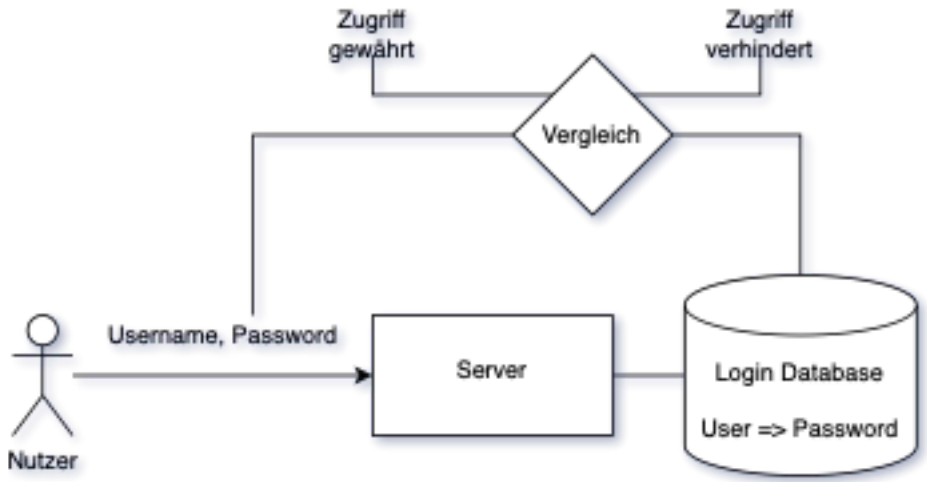
- ein Attribut entscheidet über den Zugriff

AUTHENTICATIONFACTORS

TYPEN

- Wissen
  - Password, Sicherheitsfragen ...
- Besitz
  - Security token, Smart Card ...
- Inhärenz
  - Biometrische Verfahren ...

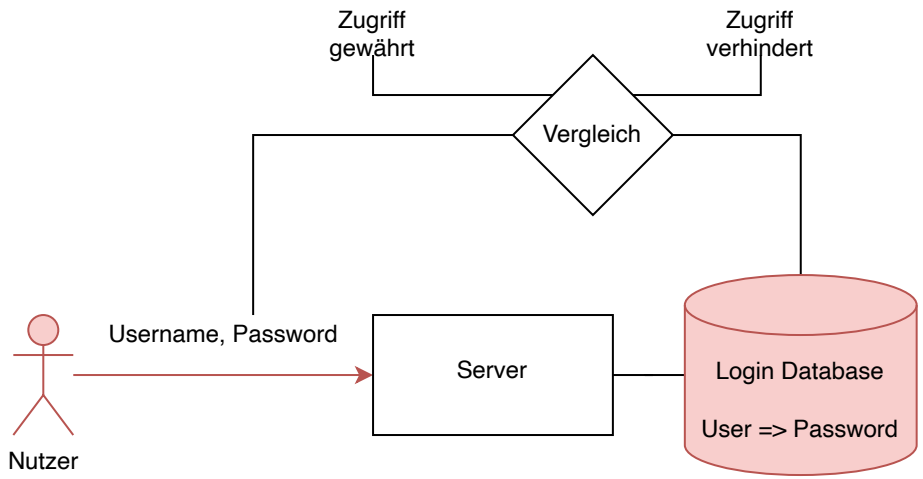
WISSEN: PASSWORT



Speaker notes

Passwort und Login wird vom User an den Server gesendet. Dieser prüft es gegen die Datenbank und gewährt entsprechen Zugriff oder verweigert ihn.

WISSEN: PASSWORT



Speaker notes

Es gibt verschiedene unsichere Punkte:

1. Der Nutzer kann eine mögliche Schwachstelle sein. Beispielsweise durch die Verwendung eines unsicheren Passwort. Aber auch weil der Rechner des Nutzers infiziert sein könnte.
2. Der Transportweg kann in der Regel als unsicher angenommen werden. Das heißt er erfüllt nicht Anforderungen an einen sicheren Kanal. Beispielsweise, dass Dritte Nachrichten abfangen oder manipulieren können. Bei ausreichender Verschlüsselung kann der Kanal als sicher angekommen werden.
3. Außerdem könnte die Datenbank aus verschiedenen Gründen geleakt werden. Liegen dann die Passwörter im Klartext vor hat ein Angreifer leichtes Spiel. Dies ließe sich einfach durch gehashte und gesaltet Passwörter verbessern.

Wie man die Sicherheit eines solchen Prozess erhöht betrachten wir später. Ähnliche Verfahren bei Sicherheitsfragen.

BESITZ

Speaker notes

Die Anwendung prüft ob der Anwender die richtige Smart Card oder ähnliches besitzt.

BIOMETRISCHE VERFAHREN

- Fingerabdruck
- Iriserkennung
- Gesichtserkennung
- Venenerkennung
- Brainwave basiert (noch in der Entwicklung)

BIOMETRISCHE VERFAHREN: SICHERHEIT?

Speaker notes

Bisher sind alle biometrischen Verfahren zu universell sowie nicht eindeutig genug um praxistauglich und sicher zugleich zu sein. Sind trotzdem weit verbreitet. Hinweis auf Schäubles Fingerabdruck

EXKURS BRAINWAVE BASED AUTHENTICATION

- Aktuell Forschungsgebiet
- Misst die Gehirnströme
- Mögliche Messarten:
  - einmalige Sequenz
  - dauerhaftes Messen und überprüfen

EXKURS BRAINWAVE BASED AUTHENTICATION: PROBLEME

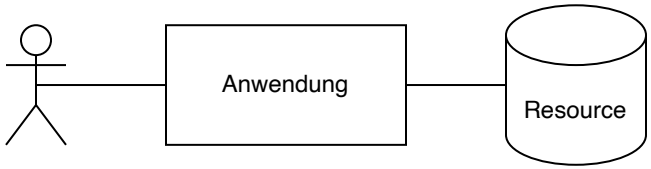
- Performance
- Akzeptanz
- Erfassung der Daten

EXKURS BRAINWAVE BASED AUTHENTICATION

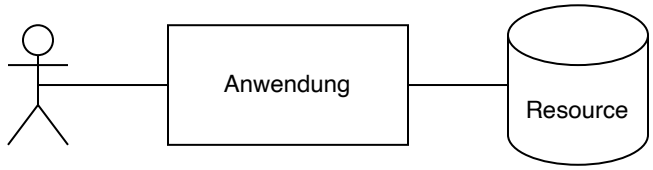
AUTHENTIFIZIERUNGSARTEN

- Direkt
- über einen dritten Abiter

DIREKT

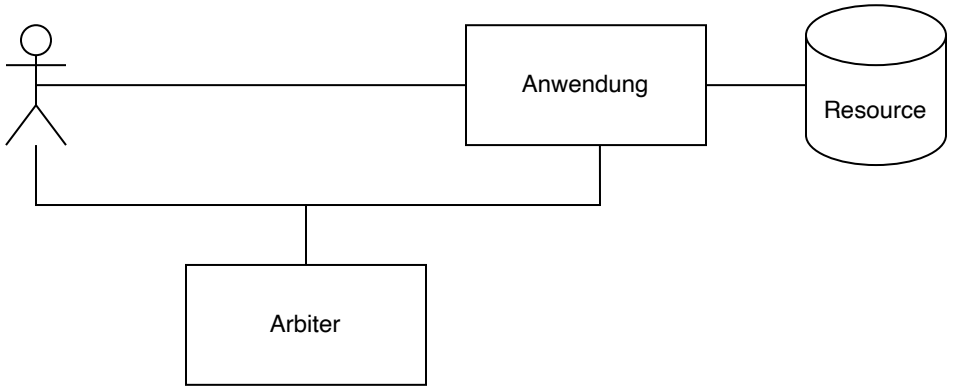


DIREKT



Vorteil: Anwendung hat die Hoheit über die Daten.
Nachteil: Nutzer muss der Anwendung möglicherweise mehr Daten bereitstellen (mindestens: Password).

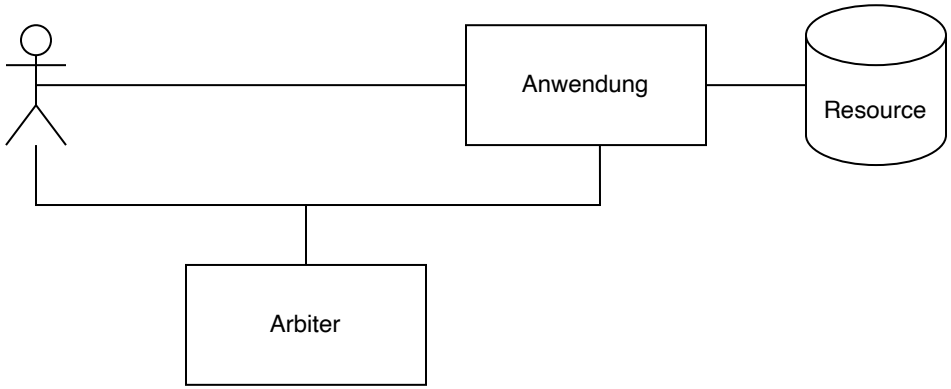
ARBITER



Speaker notes

Der Nutzer kann sich eine Session/Token bei einem externen Arbiter (meist ein Identity Provider wie bspw. Shibboleth) erstellen lassen. Diese kann er dann nutzen um sich bei einer Anwendung zu authentifizieren.

ARBITER



Vorteil: Der Anwender muss seine persönlichen Daten gegenüber der Anwendung nicht sichtbar machen.

Nachteil: Beide müssen dem Arbiter vertrauen.

MULTI FAKTOR

- meist zwei Faktor
- erhöht die Sicherheit signifikant
- mindestens zwei unterschiedliche Authentifizierungsfaktoren (Wissen+Besitz)
- gängige Arten: Zeitbasiert (OTP), Tokens (SMS), Smart Cards

GÄNGIGE VERFAHREN

- OTP
- TOTP
- U2F

OTP

- One Time Password
- meist von der Anwendung generiert und an den Nutzer über einen getrennten Kanal übermittelt
  - E-Mail
  - SMS
  - WhatsApp (neuerdings bspw: Paypal)
- nicht mit One Time Pad verwechseln

TOTP

- OTP wird aus einem Secret und Timestamp generiert
- Secret wird zunächst zwischen Anwendung und Client ausgetauscht

U2F

- spezielles Challenge Response Verfahren der FIDO Allianz
- alle gängigen Browser unterstützen mittlerweile U2F
- mittlerweile auch viele unterstütze Anwendungen
  - Nextcloud
  - Gitlab
- Spezielle USB Keys
  - Yubikey
  - Nitro

MULTI FAKTOR: PROBLEME

- bei generierten Tokens (bspw. OTP, TOTP)
  - Generierung sollte nicht auf gleichem Gerät stattfinden wie auf dem Benutzergerät

Speaker notes

Generierte Tokens: Beispiel für problematische Anwendung Banken: PushTan plus Onlinebanking App auf gleichem Gerät. an der Tafel Angriff ausführen

PROBLEME IN DER PRAXIS

NOTWENDIGKEIT SESSION

- HTTP ist zustandslos
- Zustand für Authentifizierung nötig
- Abstraktes Konstrukt: Session

SPEICHERUNG DER SESSIONS

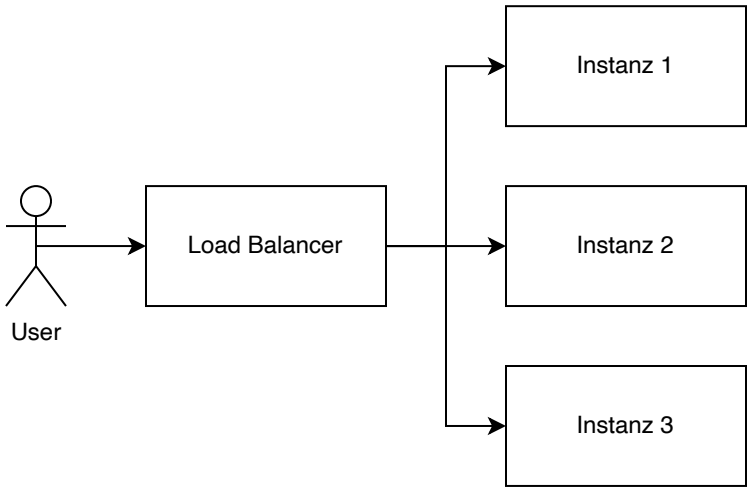
COOKIES

- Zustimmung erforderlich
- Session Riding nicht möglich

URL-REWRITING

- Client unanbhängig
- Session-ID offensichtlich
- Gefahr durch "Session Riding"

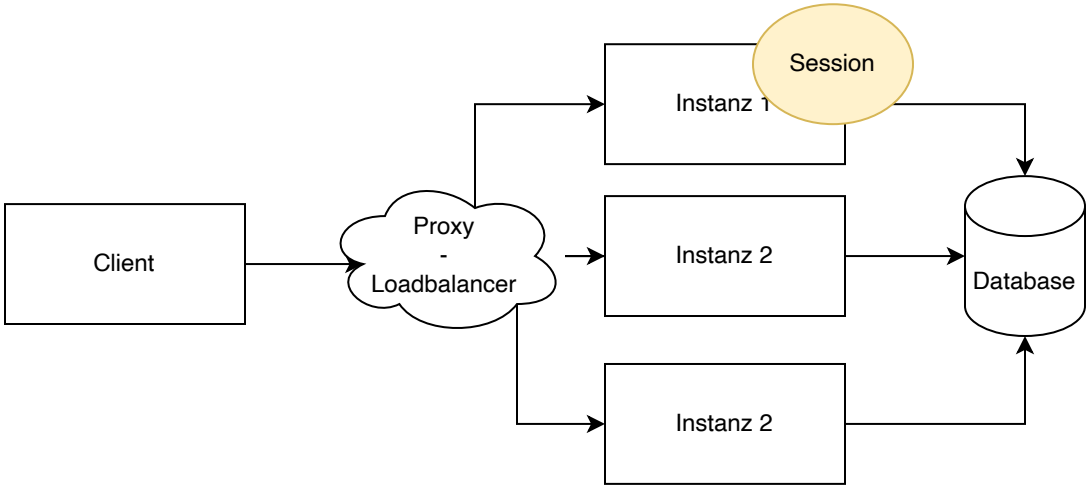
PROBLEME IN VERTEILTEN ANWENDUNGEN



Speaker notes

Sobald die Anwendung skaliert werden soll, entsteht ein Problem mit der Session Verwaltung. Dabei gibt es verschiedene Probleme die auftreten können und verschiedene Lösungen.

PROBLEME IN VERTEILTEN ANWENDUNGEN



Speaker notes

Ist eine Anwendung nicht für eine skalierte Nutzung konzipiert kann folgendes Szenario eintreten. Nutzer A verbindet sich über den Load Balancer mit der Instanz A. Dort erzeugt er eine für sich gültige Session. Diese kann er nun benutzen um sich bei Instanz A als Nutzer auszugeben. Während seiner Nutzungszeit kann es jedoch vorkommen, dass der Loadbalancer den Nutzer auf eine andere Instanz verschiebt. In diesem Fall besitzt der Nutzer keine gültige Session mehr.

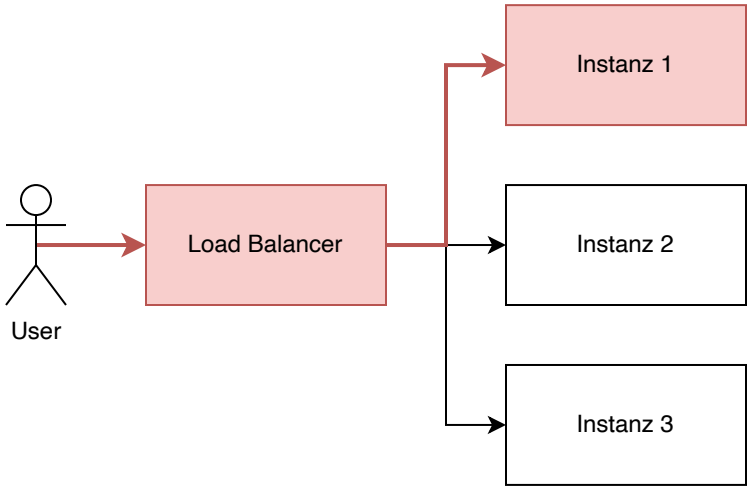
MÖGLICHE LÖSUNGEN

- Nutzer wird nach der initialen Zuweisung an eine Instanz dauerhaft gebunden
- Sessions werden Instanz übergreifend gespeichert
- Session Gateway
- Session ist tokenbasiert beim Nutzer

Speaker notes

Zusammenfassung

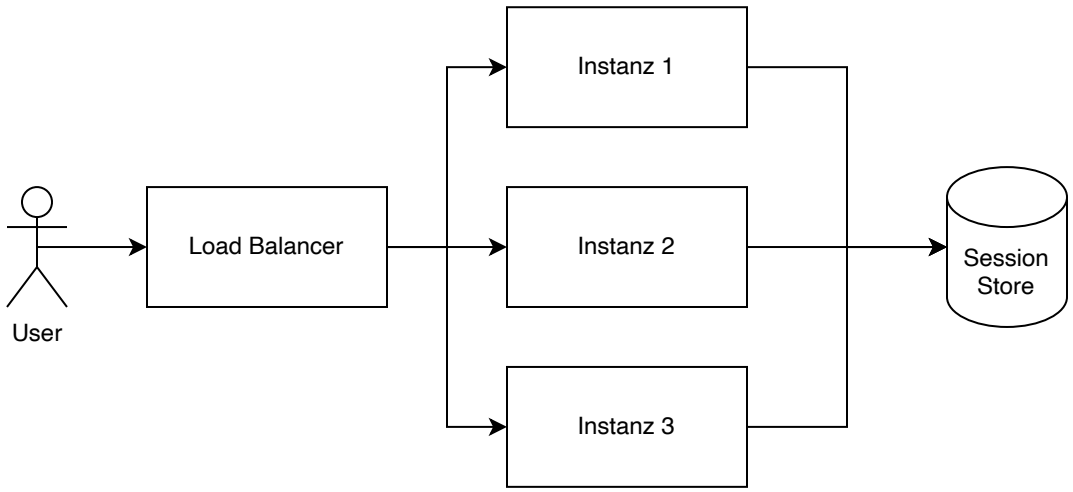
INSTANZBINDUNG



Speaker notes

Der Anwender wird in diesem Fall vom Loadbalancer immer an die gleiche Instanz weitergeleitet wird. Da der Nutzer dann immer bei der gleichen Instanz landet, muss man sich keine Gedanken über das Session Sharing machen. Vorteil: * keine extra Technik benötigt Nachteil: * Die Instanz darf nicht sterben * Last auf einer Instanz verhält sich dynamisch

INSTANZÜBERGREIFEND

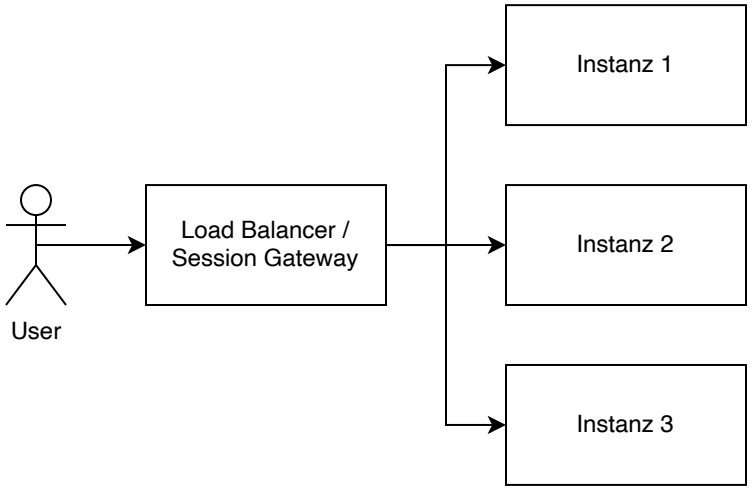


Speaker notes

Die Session eines Users wird in einem Session Store abgelegt. Jede Instanz hat Zugriff auf diese. Vorteil:

- Nutzer kann je nach Last auf den Instanzen verschoben werden
- Instanzen können auch sterben Nachteil:
- der Session Store kann ein Performance Bottleneck werden

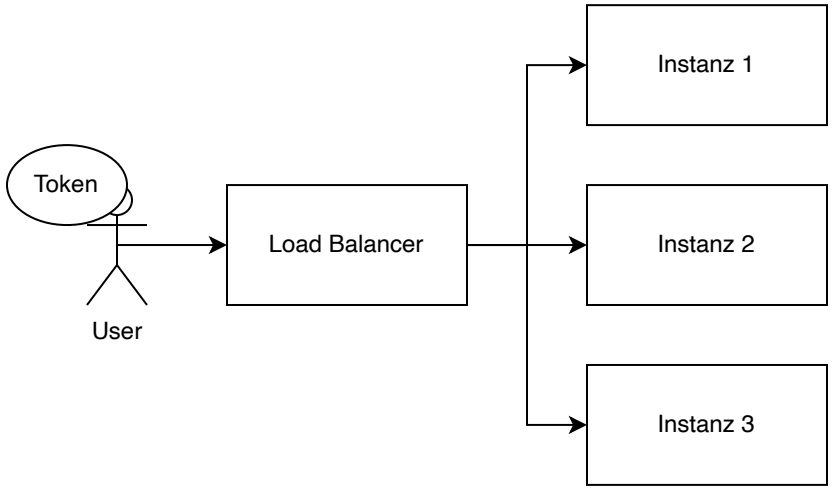
SESSIONGATEWAY



Speaker notes

Jeder Aufruf wird durch das Session Gateway geleitet. Meistens übernimmt der Loadbalancer diese Rolle. Vorteil: * Die Instanz muss sich nicht darum kümmern * eine zentrale Stelle Nachteil: * Performance Bottleneck * unter Umständen zu grob granular

TOKENBASIERTE SESSIONS



Speaker notes

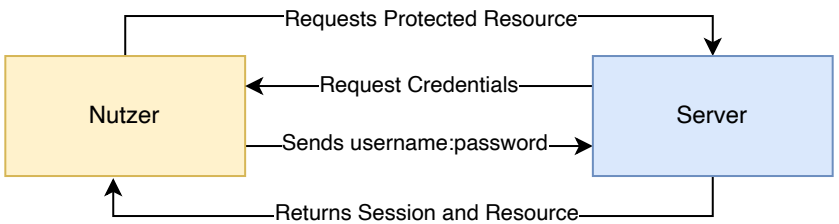
Der User schickt bei jeder Anfrage das Token mit. Eine Möglichkeit dabei ist dieses Token im HTTP Header mitzusenden. Vorteil und Nachteil zugleich: * Jede Instanz kann bzw. muss selber entscheiden können ob das Token gültig ist

MÖGLICHE AUTHENTIFIZIERUNGVERFAHREN

HTTP BASIC AUTHENTICATION

- Browser stellt Forumlar bereit
- Credentials-Tupel username:password
- meist authentifiziert der Server
- Formular nicht editierbar

HTTP BASIC AUTHENTICATION: ABLAUF



FORM BASED AUTHENTICATION

- Formular wird von der Anwendung erzeugt
- Anwendung entscheidet über Zugang
- bessere Fehlerbehandlung

PROTOKOLLE

- Security Assertion Markup Language (SAML)
- OAuth2

AUTHORISATION

gewährt Usern Zugriff auf Resourcen

AUTHENTICATION

stellt sicher, dass der Nutzer auch wirklich der ist für den er sich ausgibt

Speaker notes

Unterscheidung Authorisation vs. Authentication wichtig zur Erkennung der Unterschiede zwischen OAuth2 und SAML.

TERMINOLOGIE

SAML

- Client
- Identity Provider (IDP)
- Service Provider (SP)

OAuth2

- Client
- Authorisation Server
- Resource Server

Speaker notes

- Client ist der jeweilige User/Browser/Server der sich authentifizieren bzw. autorisieren möchte
- Server, welcher die Identitäten und Zugangsdaten enthält
- Die geschützte Resource/Anwendung

SECURITY ASSERTION MARKUP LANGUAGE

- XML basiertes Authentication Protokoll
- Single Sign On (SSO)
- Optional Single Sign Off (SLO)
- Identity Management

Speaker notes

Single Sign On erfordert nur ausgetauschte Zertifikate zwischen IDP und SP. Der SP kann dann das vom IDP signierte Zertifikat des Client verifizieren. Für Single Logout wird eine aktive Verbindung zwischen SP und IDP benötigt. Der IDP bestätigt dann die Gültigkeit des Zertifikat. Logout funktioniert ansonsten durch das "Vergessen" des Zertifikat. Daher sollten die Zertifikate ein Ablaufdatum haben.

OAUTH2

- meist JSON Web Tokens (JWT)
- Client muss nicht zwingend ein Browser sein
- Autorisierungsprotokoll
- Access and Refreshtokens

Speaker notes

- Accesstokens haben in der Regel eine Lifetime
- läuft diese ab, kann man mit dem Refreshtoken einen neuen Accesstoken anfordern
- Refreshtoken haben daher eine längere Lifetime

ABLAGE DER TOKENS

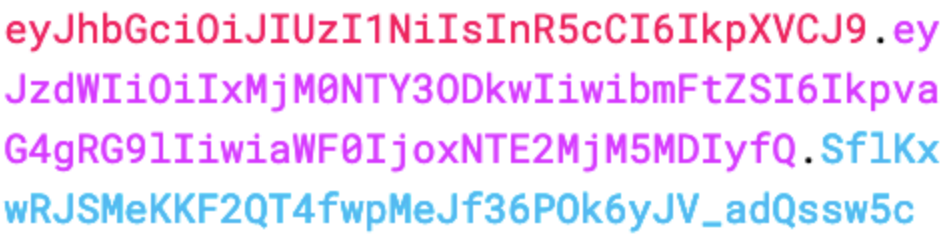
- Besondere Vorsicht wo die Tokens gespeichert werden
  - NICHT im Local- / Sessionstorage
- Auth0 Doku bietet Best Practices für verschiedene Szenarien

JSON WEB TOKEN

GENERELLES

- von AUTH0 bereitgestellt
- mittlerweile Libraries für alle gängigen Sprachen
- Framework für Autorisierungstokens

AUFBAU JWT


eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

Speaker notes

- ein Token besteht aus drei Teilen
- die Teile werden durch einen Punkt getrennt
- der 1. Teil (rot) beinhaltet Daten über den Hash Algorithmus und den Typ
- der 2. Teil (lila) beinhaltet die Payload Daten
- der 3. Teil (cyan) ist ein Hash über die beiden ersten Teile plus einem Secret

TOKEN LIFECYCLE

1. JWT wird mit Header und Payload wird vom Autorisierungsserver bestückt
2. Autorisierungsserver signiert den Token mit dem Secret und sendet ihn an den Client
3. Client sendet den Token an die Anwendung
4. Anwendung prüft mithilfe des Secret den Token

Speaker notes

- Secret darf nur dem Autorisierungsserver und dem Anwendungserver bekannt sein
- sollte das Secret öffentlich werden müssen alle Tokens als invalide behandelt werden

JWT.IO PRAXIS