Projet Informatique

2019

Steve Hostettler

Software Modeling and Verification Group

University of Geneva

Sécurité

Vocabulaire

- Utilisateur
- Secret (mot de passed)
- Role

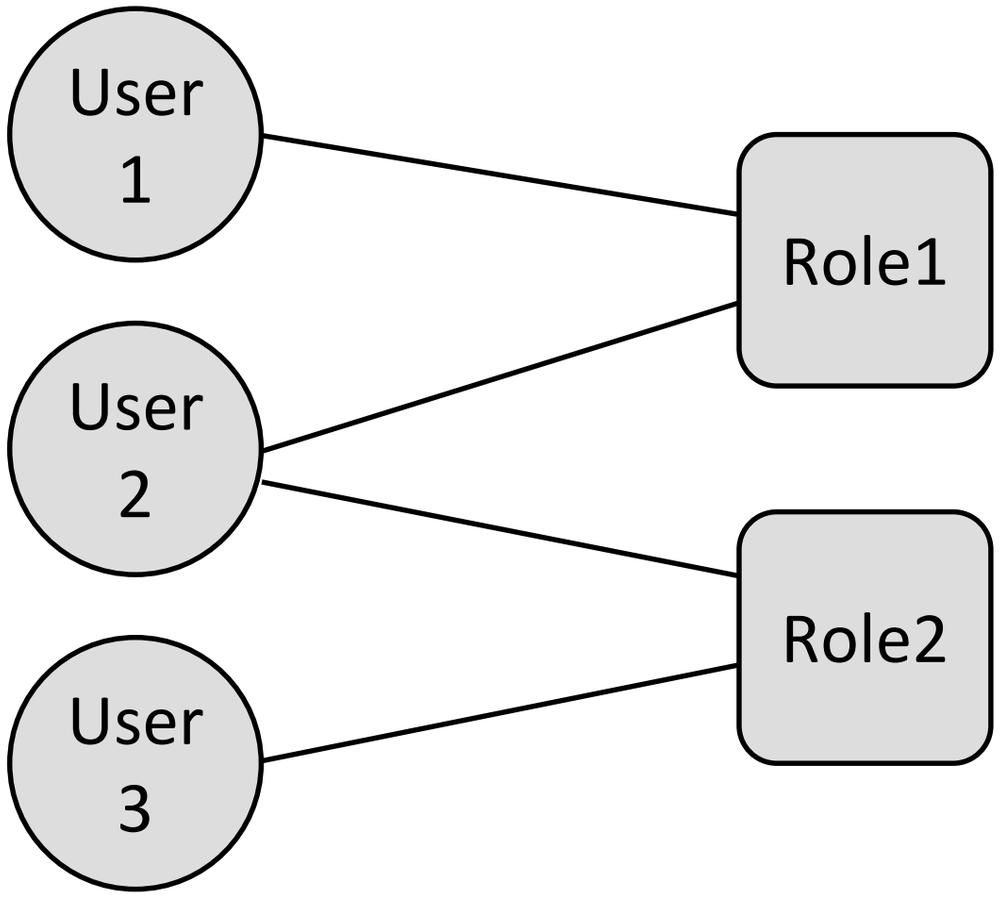
Vocabulaire

- Authentification (Authentication): est-ce que l'utilisateur est celui qu'il prétend être?
- Autorisation: à quel rôle appartient un utilisateur?
- Audit

Authentification multi-facteur

- Posseder un object physique : une clé USB, un telephone portable, une carte bancaire, ….
- Connaitre quelque chose de secret (mot de passe, pin)
- Une carastéristique physique (empreinte digitale, …)
- Être quelque part : GPS, dans un bâtiment précis, sur un reseau précis.

Contrôle d’accès



Métaphore

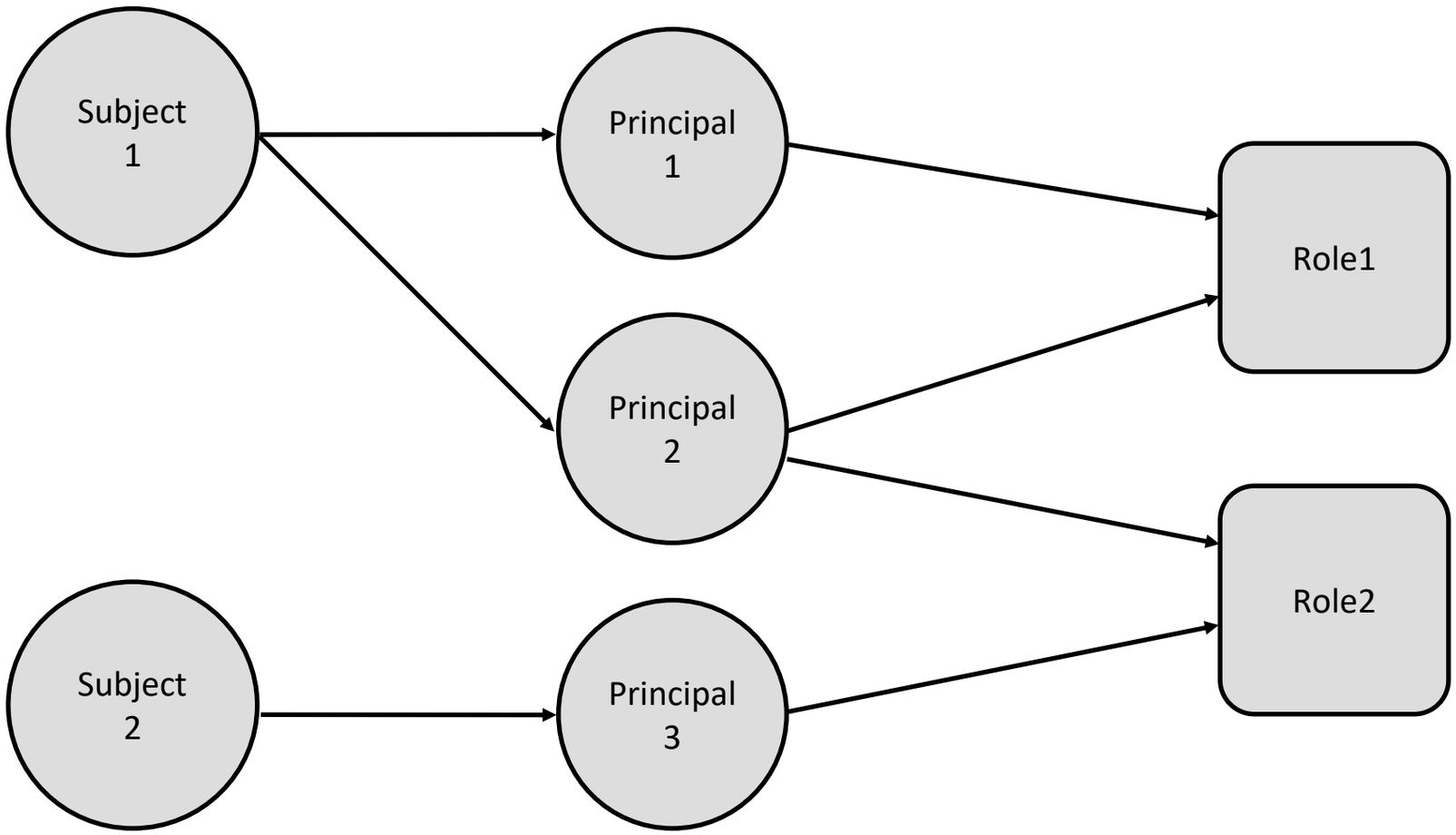
- Application: bâtiment à protéger
- Vulnérabilité: façons de rentrer dans le bâtiment
- Il faut vérifier les portes, les fenêtres et l’entrée du chat

Sécurisé une architecture microservices

Vocabulaire

- Subject: une personne ou une organisation
- Principal: nom d’utilisateur
- JAAS : Java Authentication and Autorisation Service
- Realm: liste d’utilisateurs + liste de rôles
- Données de connexion (crédentials)

Vocabulaire



Vocabulaire

- SSL (TLS) termination :
- Bearer token : is a string without meaning
- ID Token : used by the client to identify users
- Access tokens carry the necessary information to access a resource directly. Access tokens usually have an expiration date and are short-lived.

If the backend network is trusted, there is no need to re validate authentication for each microservice. If not, each microservices must revalidate the incoming request.

Trusted network and therefore traffic is not encrypted (this would not hold for a multi-tenant shared network)

UI - API Gateway (konga)

Client

API Gateway (kong)

Instrument Service

Valuation Service

Regulatory Service

Counterparty Service

Message broker (Kafka)

Cpty-db

Logging (ELK)

IAM (keycloak)

Web UI

api-db

sso-db

API gw (& potentially microservices) must know the IAM public key to know whether it can trust the security token (by validating the signature)

Backend network

financial-app.com

Gateway network

Frontend network

Encrypted traffic

Unencrypted traffic

No request can get to the a microservices without get through the gw and thus being authenticated.
API gw does SSL termination (rest of the traffic is non-encrypted). API gw only listens to 443 and requires encrypted traffic.
API gw does DDOS control

Static Content

Technical Component

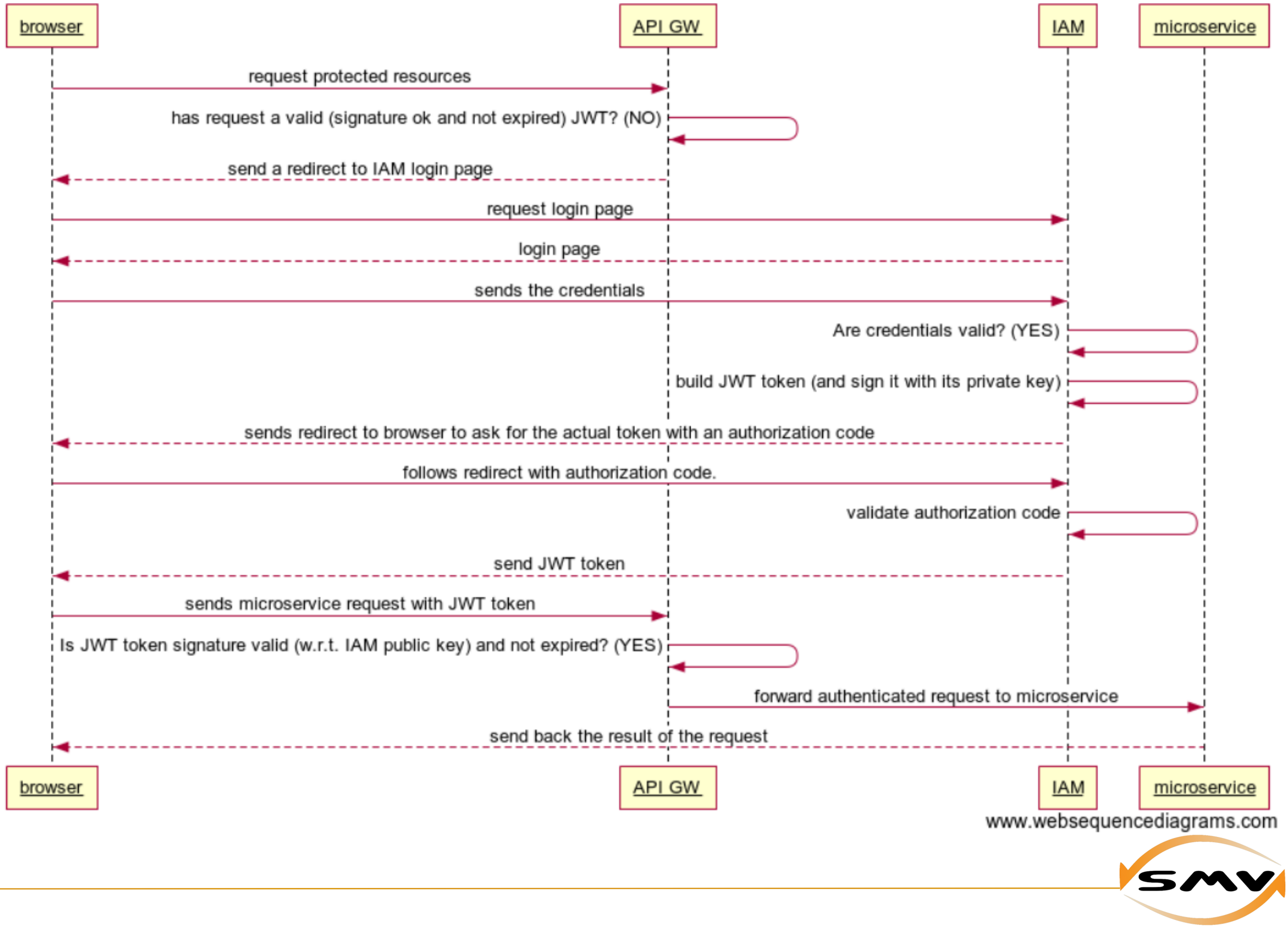
Micro Service

Firewall (port 443) Opened to the WWW

Firewall (port 80 http and port 5432 for db) Only accessible from gateway network

Authentication and Authorization

- OAuth provides only authorization and more specifically authorization delegation
- OpenID Connect provides authentication (on top of OAuth 2.0)


browser
API GW
IAM
microservice
request protected resources
has request a valid (signature ok and not expired) JWT? (NO)
send a redirect to IAM login page
request login page
login page
sends the credentials
Are credentials valid? (YES)
build JWT token (and sign it with its private key)
sends redirect to browser to ask for the actual token with an authorization code
follows redirect with authorization code.
validate authorization code
send JWT token
sends microservice request with JWT token
Is JWT token signature valid (w.r.t. IAM public key) and not expired? (YES)
forward authenticated request to microservice
send back the result of the request
browser
API GW
IAM
microservice
www.websequencediagrams.com

JWT Token


```
{
  "iss": "http://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "gender": "female",
  "birthdate": "0000-10-31",
  "email": "janedoe@example.com",
  "picture": "http://example.com/janedoe/me.jpg"
}
```

Multi-Factor Authentication

- Something you know (password)
- Something you have (a trusted device that is not easily duplicated, like an RSA SecurID)
- Something you are (fingerprints, retina scan)

Exemples d’attaques (liste pas du tout exhaustive)

Cachez cette session que je ne saurais voir

- Fixation de session : Copier le token de session qui est dans l'URL ou envoyer un lien avec un sessionid existant
- Cachez la session dans un cookie

```
<session-config>
    <cookie-config>
        <http-only>true</http-only>
    </cookie-config>
    <tracking-mode>COOKIE</tracking-mode>
</session-config>
```

- Recréer un nouveau session pour tout login

“*ç”)R*ç”*ç=”(&=7)ç

- Toujours crypter la conversation

```
<security-constraint>...
    </auth-constraint>
    <!-- The following enforces SSL -->
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

...

- Toujours crypter les cookies

```
<cookie-config>
    <http-only>true</http-only>
    <!-- The following can only be activated when on SSL -->
    <secure>true</secure>
</cookie-config>
```

...

Injection de code

Injection de code XSS

- Entrée des données dans les formulaires pour qu'elles soient interprétées lors de l'affichage.
- JPA est protégé par défaut:

http://ha.ckers.org/xss.html
Owasp Organisation
Plein de video sur Youtube: Hacksplaning

Tips and check list

Pour vivre heureux, vivons cachés

Ne pas affichés le listing des fichiers

```
<servlet>
   <servlet-name>DefaultServlet</servlet-name>
   <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
   <init-param>
      <param-name>debug</param-name>
      <param-value>0</param-value>
   </init-param>
   <init-param>
      <param-name>listings</param-name>
      <param-value>false</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
   <servlet-name>DefaultServlet</servlet-name>
   <url-pattern>/</url-pattern>
</servlet-mapping>
```

Toujours valider du coté serveur

- Les services REST et SOAP NE DOIVENT PAS faire confiance aux clients
- Toujours valider les données en entrée
- Utiliser des filtres pour se protéger des injections de codes
- Utiliser JPA et pas JDBC natif et éviter les query construite mais utilizer le “**Query Parameterization** language”
- Apply JSON sanitizer

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Injection_Prevention_Cheat_Sheet_in_Java.md

OAuth2

OAuth2 is about access delegation (and revokation)

Terminology:

- Protected resources : data or api
- Resource server : host protected resources
- Client: 3rd party that want to access protected resources
- Resource owner : the one that grants access to the client for specific protected resources
- Auth Server: authenticate both client and resource owner and determine
- Access Token: delivered by the auth server, used by the client to get access to protected resources on behalf of the resource owner.

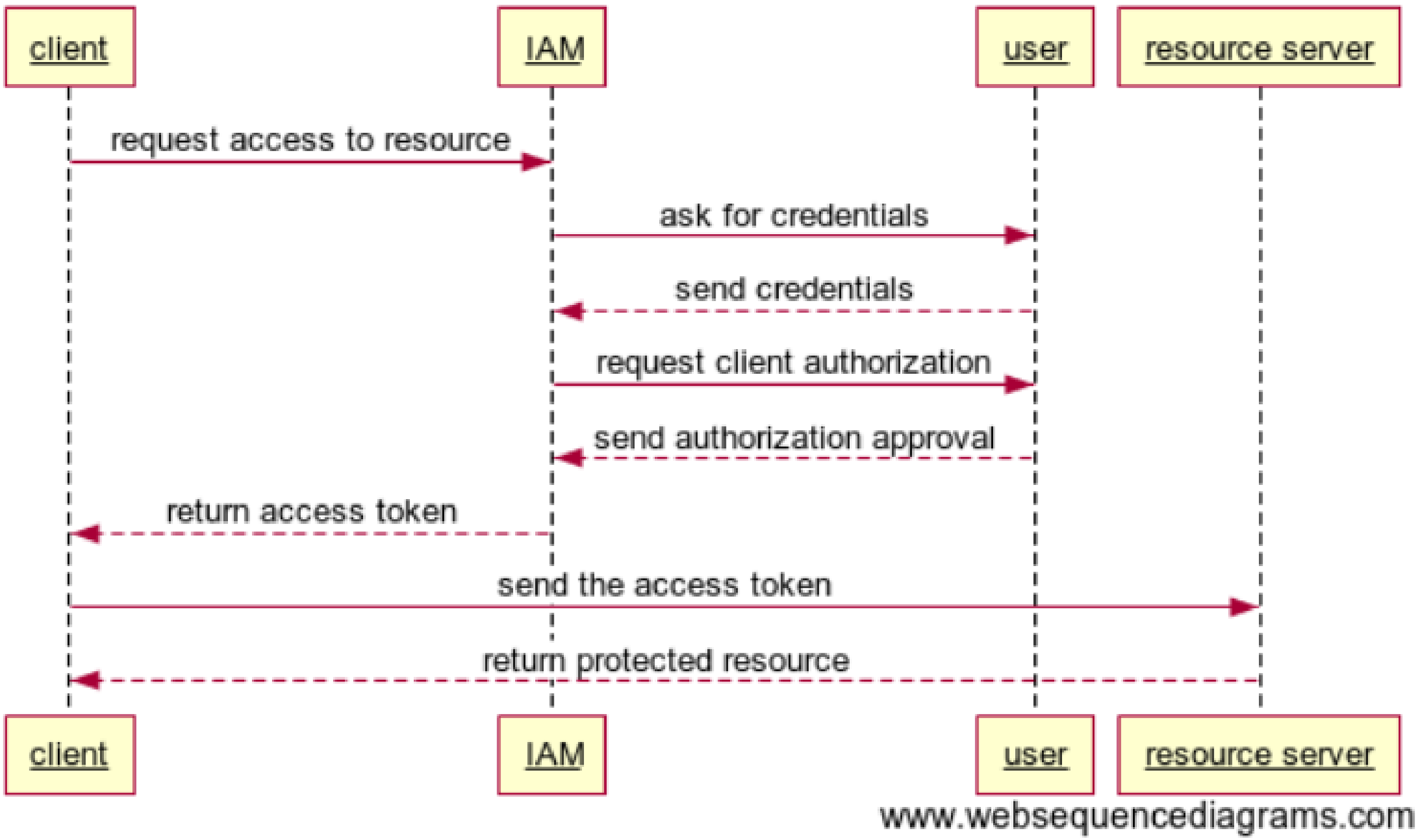
OAuth2 :clients

- Confidential clients: considered to be a protected and secure environment that can safeguard secrets. It receives a client identifier **and** a client secret to authenticate against the authorization server. Usually used to grant and powerful access.
- Public clients: cannot hold a secret. Usually less powerful and short term

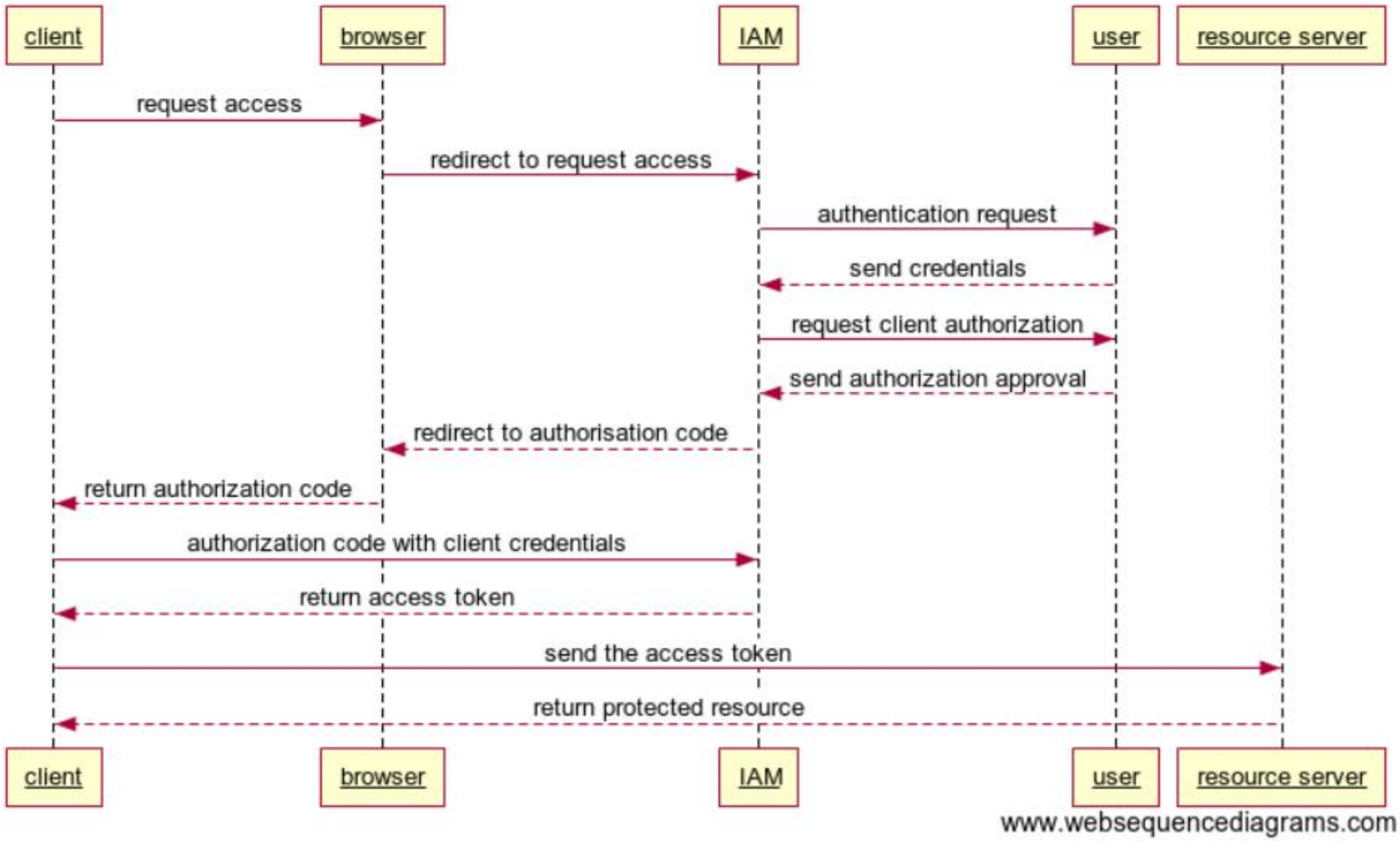
OAuth2 : Grants

- Implicit grant : for unsecured clients such as web app. The implicit grant is an access token.
- Authorization Code Grant: most elaborate and secure grant. To gain long term access. It requires client authentication. Only for confidential clients (not public)
- Resource Owner Password Credentials Grant : For trusted application as it request the user credentials.
- Client credentials grant : authenticate using client id and client credentials (not resources owner credentials)

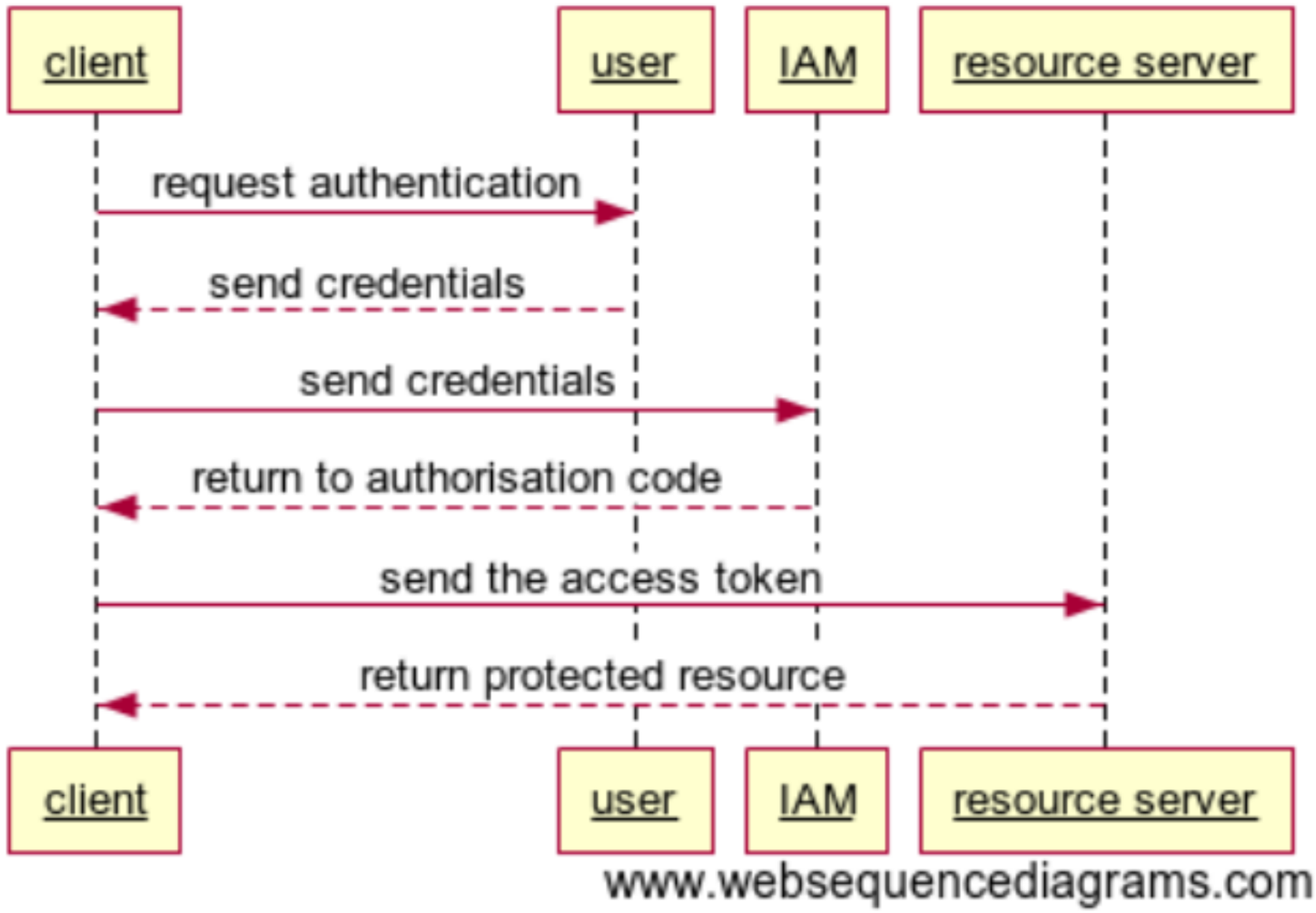
Implicit Grant

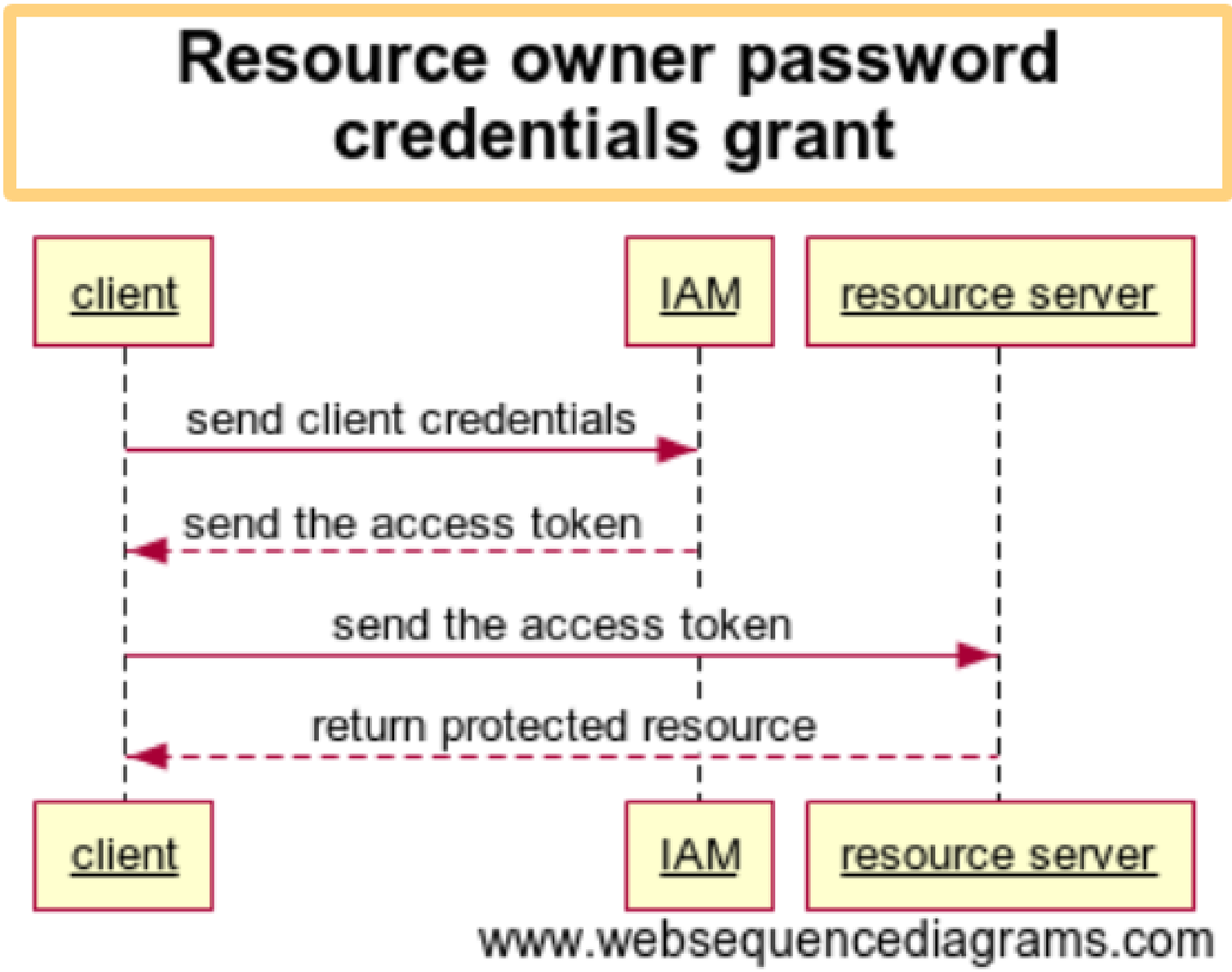


Authorization code Grant



Resource owner password credentials grant




Resource owner password credentials grant
client
IAM
resource server
send client credentials
send the access token
send the access token
return protected resource
client
IAM
resource server
www.websequencediagrams.com

How to find security issues?

- Static analysis of code (Sonar)
- Dynamic testing (NetSparker)
- Penetration testing
- Threat Modelling

Threat Modelling

Threat modeling is a process by which potential threats, such as structural vulnerabilities can be identified, enumerated, and prioritized – all from a hypothetical attacker's point of view.

STRIDE is a Microsoft methodology for threat modelling.

STRIDE

Threat	Desired property
Spoofing	Authenticity
Tampering	Integrity
Repudiation	Non-Repudiability
Information disclosure	Confidentiality
Denial of Service	Availability
Elevation of privilege	Authorization

What can go wrong in this system we're working on?

With a flow diagrams, for each component and each threat, then focus on trust and component boundaries first.