FS20

DISTRIBUTED SYSTEMS

Authentication

T. Bocek
thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Information security – Key concepts / Access control

- **Confidentiality**
  - Confidentiality protects transmitted data against eavesdroppers
- **Integrity**
  - Provides protection against the modification of a message
- **Availability**
  - Data needs to be available when needed
- **Non-repudiation**
  - Non-repudiation provides that neither the sender nor the receiver can deny that a communication has taken place
- **Identification**
  - E.g. with a username “alice”, you are claiming to be Alice
- **Authentication**
  - Verifying a claim of identity. E.g., Alice shows passport, so an other person can authenticate her. Different authentication types:
    - Something you know: things such as a PIN, a password
    - Something you have: a key, a swipe card
    - Something you are: biometrics: palm, fingerprint
- **Authorization**
  - Determined what resources they an authenticated user is permitted to access

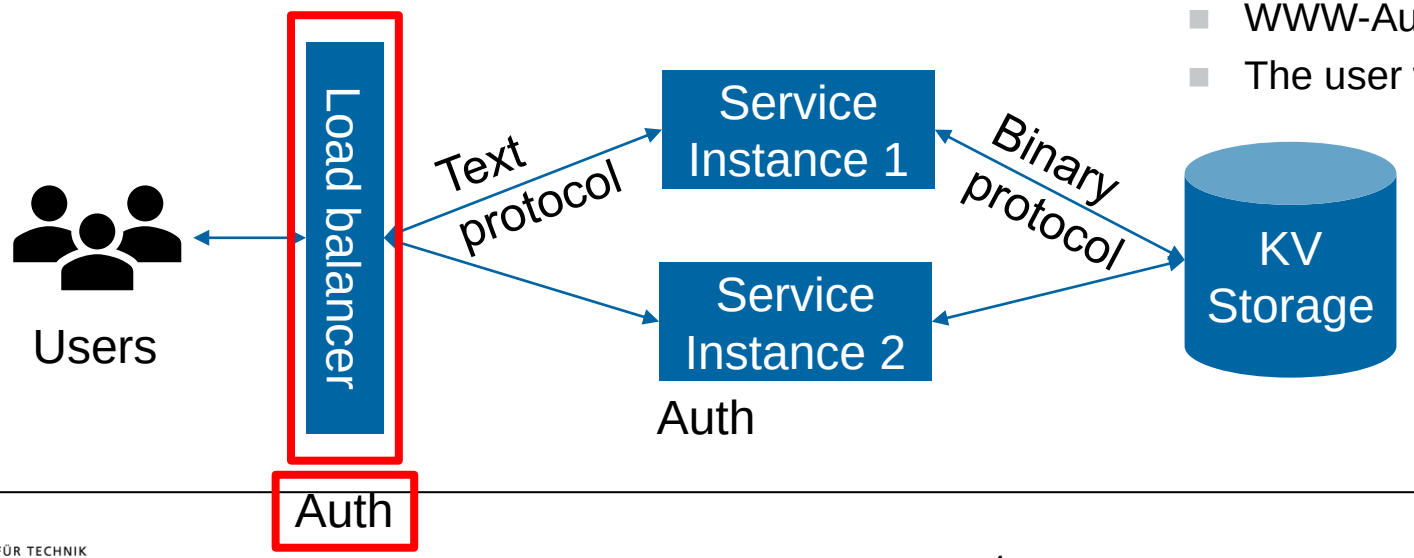
Authentication

- **Authentication**
  - Single-factor authentication
    - E.g. password
  - Multi-factor authentication / 2FA
    - E.g. password **and** software token
- **Password rules**
  - Don’t use:
    - The name of a pet, child, family member, or significant other
    - Anniversary dates and birthdays
    - Birthplace
    - Name of a favorite holiday
    - Name of a favorite sports team
    - The word "password“
  - Don’t’ reuse passwords, use password managers
  - Don’t enter passwords on unencrypted sites
  - Password length: password cracking with 5000$ in 2018 with hashcat
    - Hashtype: WPA/WPA2: 1190.5 kH/s

Pw len	com	time
6	11m	0.3d
7	656m	21d
8	38b	3.4y
9	$7 * 10^{15}$	197 y
10	$4 * 10^{17}$	11k y
11	$2 * 10^{19}$	665k y
12	$1 * 10^{21}$	38m y

Authentication

- **Software token: TOTP (Time-based One-time Password)**
  - Often used as 2nd factor, e.g., in andOTP
  - Based on keyed-hash message authentication code
  - ~ hash(key + message)
  - TOTP: message → T = (Current Unix time - T0) / X (X default is 30sec)



- **Challenge Task**
  - In service, e.g., your HTTP server
  - In load balance, e.g. traefik
  - Choices…
- **Load balancer**
  - Basic Auth, only with HTTPS, no logout!
  - Server will reply with header
    - WWW-Authenticate: Basic realm="restricted area"
    - The user will see the information "restricted area"
  - Client will provide
    - Authorization: Basic <base64>
    - <base64> contains the username:password in base64
  - Can be encode in URL
    - https://username:password@dsl.hsr.ch
    - Attention:
    - http://www.google.com:search@evil.com

Authentication

- **Digest Auth**
  - Also available in traefik
  - Hash + nonce, against replay attacks
  - Server sends
    - WWW-Authenticate: Digest realm="testrealm@host.com",
      …
      nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
      …
  - Client sends in HTTP header
    - Authorization: Digest
      username=“Alice",
      realm="testrealm@host.com",
      nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
      uri="/dir/index.html",
      …
      response="6629fae49393a05397450978507c4ef1"
  - Advantages
    - PW not in clear text
    - Nonce for replay protection for client and server
  - Disadvantages
    - Browser L&F
    - Strong security is optional
    - Cannot use scrypt or bcrypt to store PWs
- **Public/private key**
  - Client-side certificate in nginx:

```
# This will return a 403 to all clients without a proper certificate
if ($ssl_client_verify != "SUCCESS") { return 403; }

# This tells Nginx what CA to verify against
ssl_client_certificate /tmp/ca.pem;

# This tells Nginx to verify clients
ssl_verify_client on;
```

Authentication

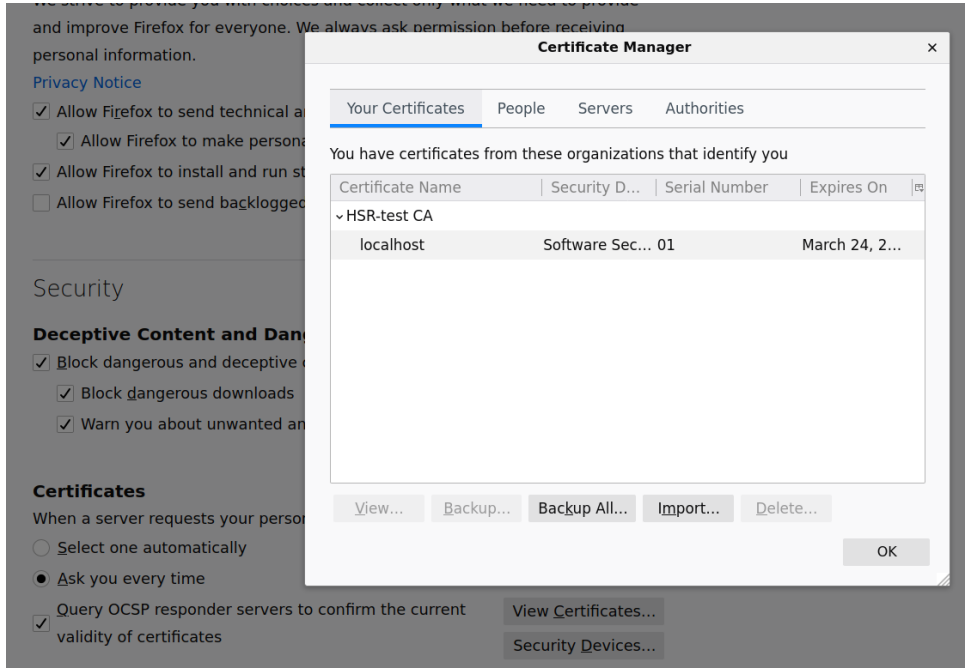
- **Create SSL CA certificates for server with openssl command**

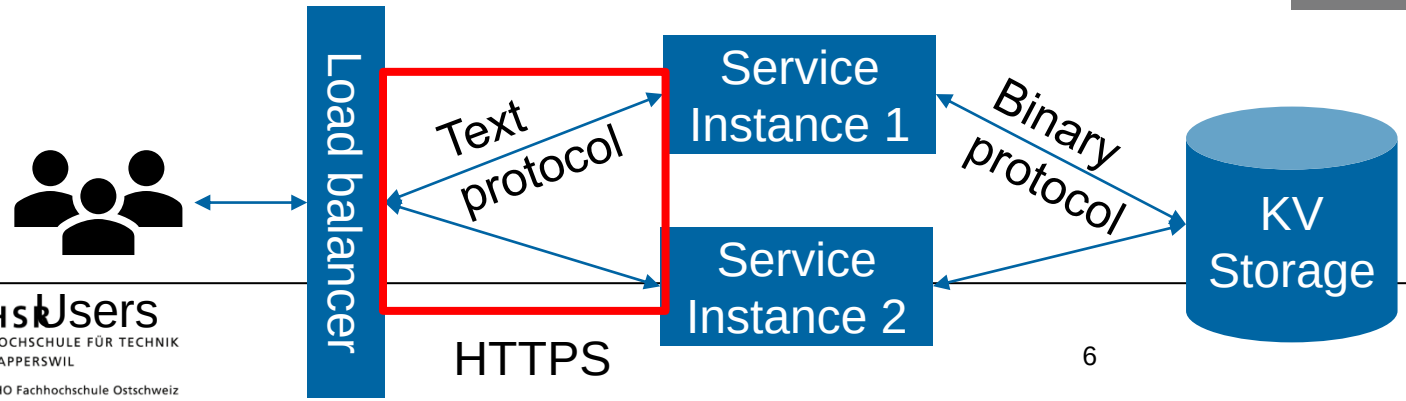
```
#generate CA certificate and private key
openssl req -new -nodes -x509 -keyout ca.key -
newkey rsa:4096 -out ca.pem -days 3650
```

  - Create client certificate

```
# create signing request
openssl req -new -nodes -keyout user.key -newkey
rsa:4096 -out user.pem
# sign the previous request
openssl x509 -req -days 365 -in user.pem -CA
ca.pem -CAkey ca.key -set_serial 01 -out user.crt
# convert to pkcs12 format
openssl pkcs12 -export -out user.pfx -inkey
user.key -in user.crt -certfile ca.pem
```

- **Add client cert to FireFox**

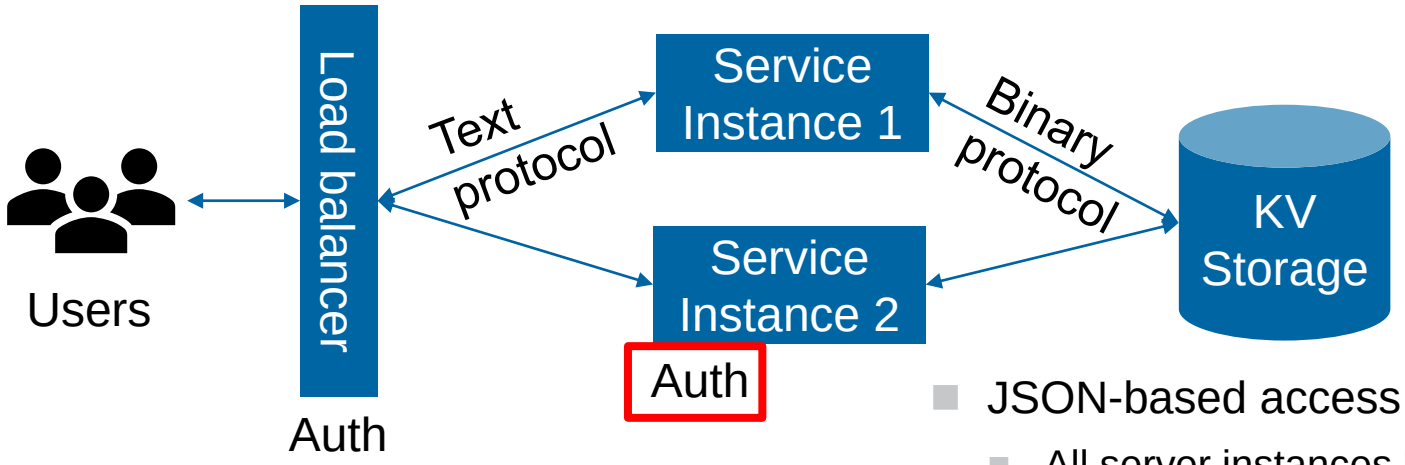




- **Add nginx security in your local network**
  - proxy_ssl_certificate, proxy_ssl_certificate_key

Authentication

- **Session-based authentication (stateful)**
  - org.apache.catalina.session.StandardManager based on ConcurrentHashMap
- **JSON Web Token (JWT) (stateless)**

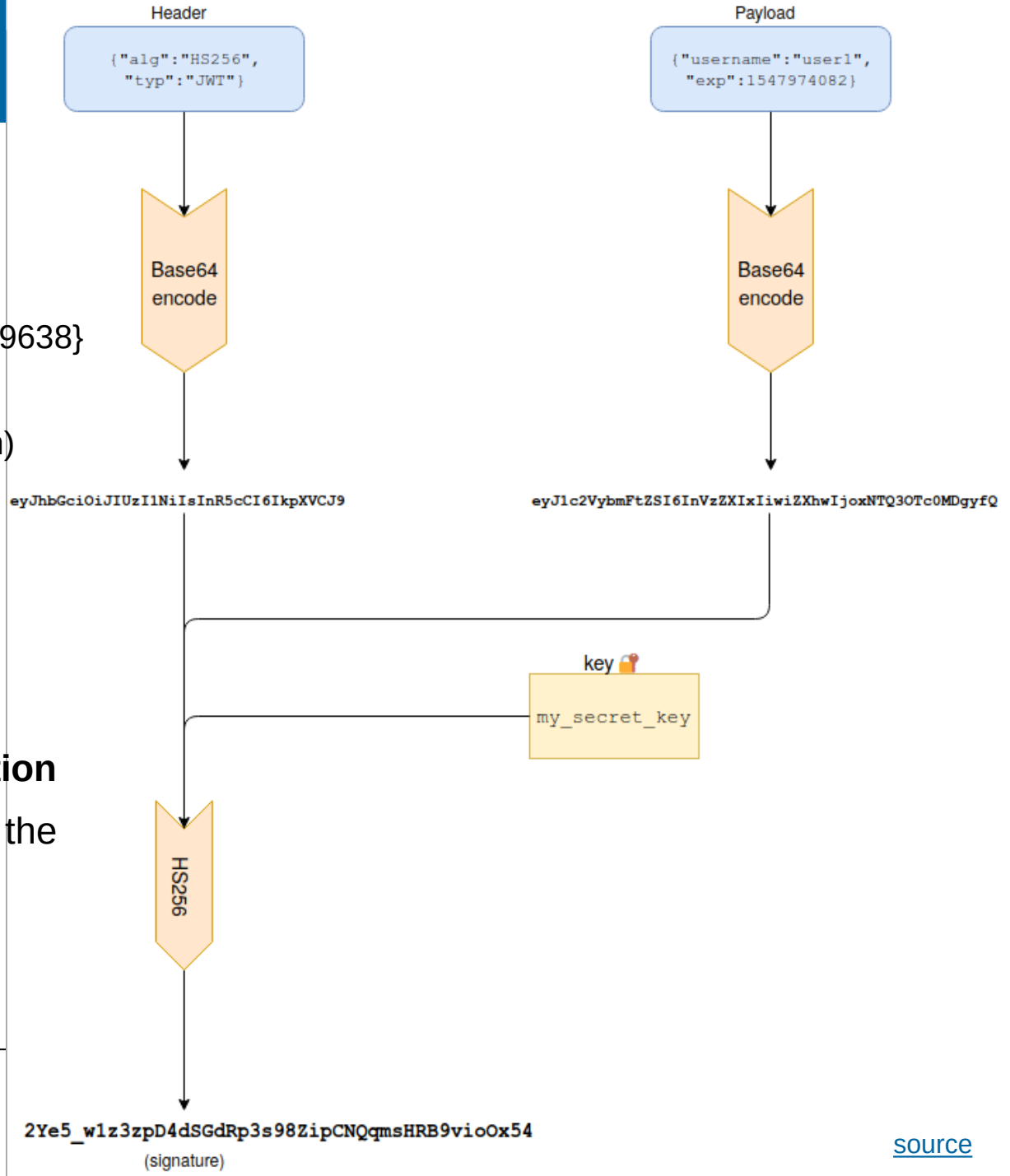


- **E.g., spring-boot**
  - Simple login app
  - JSESSIONID, Session information, that user was successfully authenticated: memory
- JSON-based access tokens
  - All server instances know a secret token
  - When user logs in, server responds with token:
  - Authorization: Bearer <token>
  - const user_token = base64urlEncoding(header) + '.' + base64urlEncoding(payload) + '.' + base64urlEncoding(signature)

Authentication

- JSON-based access tokens
  - Header: {"alg" : "HS256"}
  - Payload: {"user" : "tom", "role" : "admin", “exp" : 1422779638}
- Signature (simple): keyed-hash message
  - ~hash(base64(header)+base64(payload) + secret token)
- Client can store user_token in
  - localStorage.setItem(“token", userToken);
- Example in golang with JWT
  - Tutorial: here and here

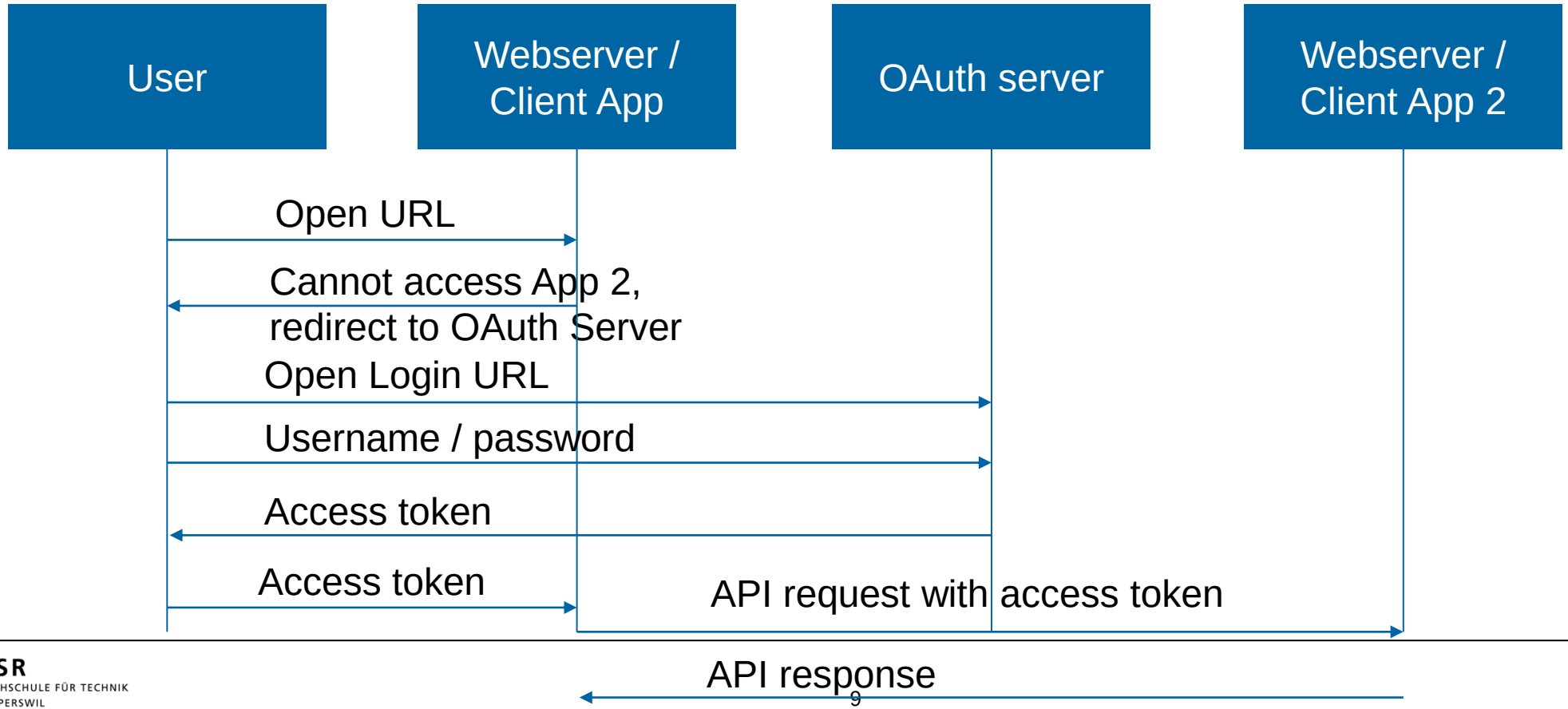
- **OAuth - protocol for authorization 3rd party integration**
  - Grant access on other websites without giving them the passwords



source

OAuth

- **OAuth flow**
  - Tokens have expiry date
  - Tokens can be refreshed



Questions?

Dr. Thomas Bocek
thomas.bocek@hsr.ch