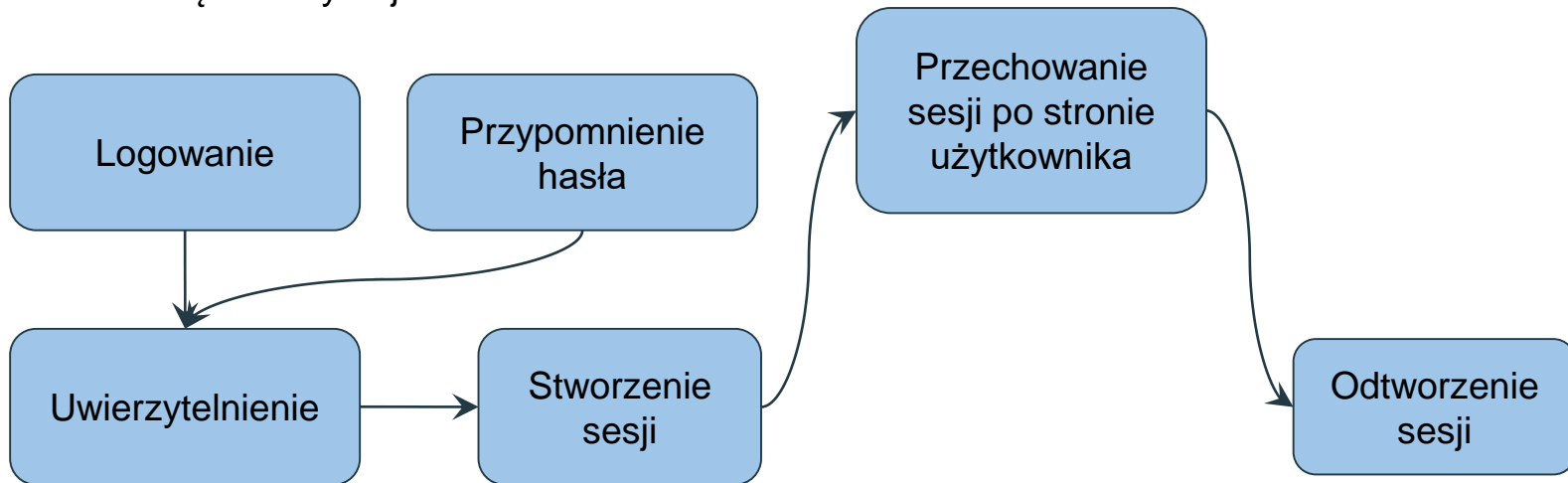


Uwierzytelnienie i sesja

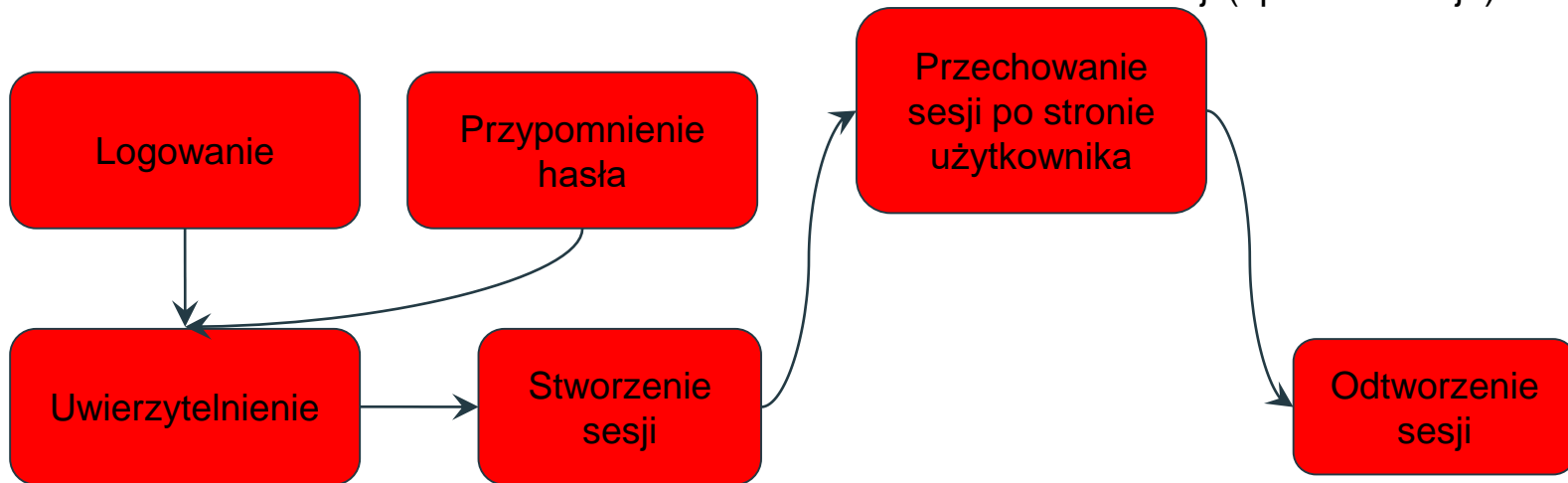
- HTTP jest bezstanowe.
- Sesja wiąże żądania między sobą -> utrzymuje stan.



Uwierzytelnienie i sesja

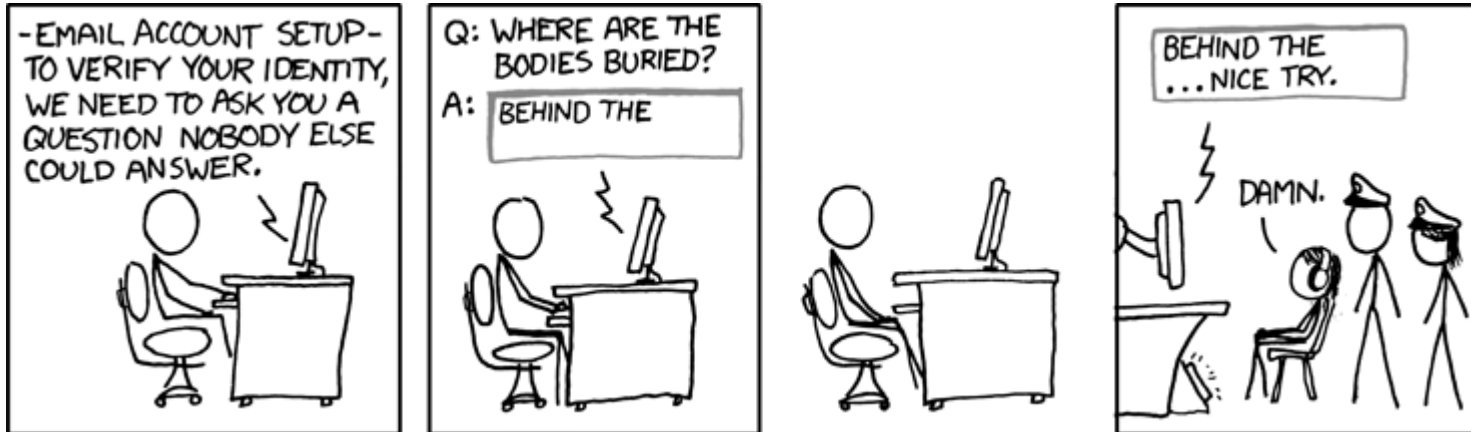
Co może pójść nie tak?

- Sfałszowanie sesji.
- Kradzież sesji.
- Przejęcie konta.
- Inne konsekwencje wynikające z nietypowego przetwarzania sesji (np. serializacja).



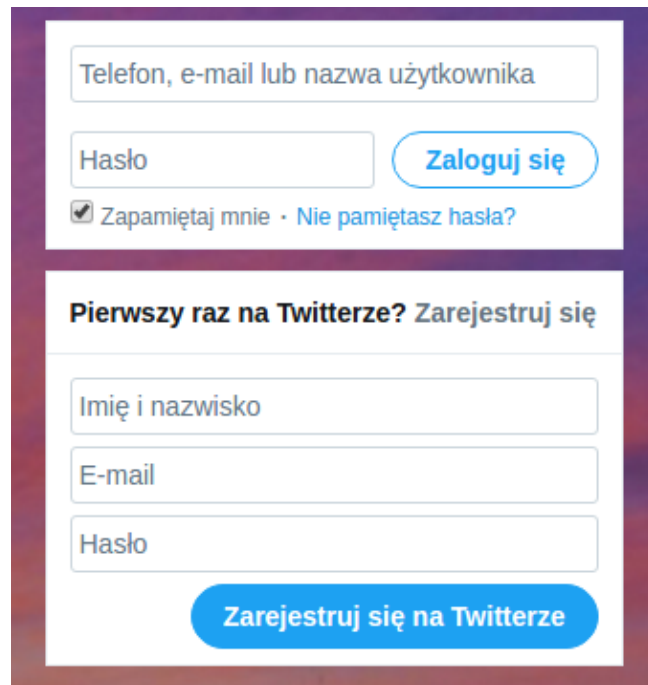
Uwierzytelnienie i sesja

- A2 – Broken Authentication and Session Management
- V2. Authentication
- V3. Session management
- Błędy implementacji:
 - Brak zabezpieczenia sesji.
 - Niebezpieczny format przechowywania sesji.
 - Ujawnianie identyfikatorów sesji. Niepoprawne generowanie identyfikatorów oraz unieważnianie sesji.



Uwierzytelnienie i sesja - Logowanie

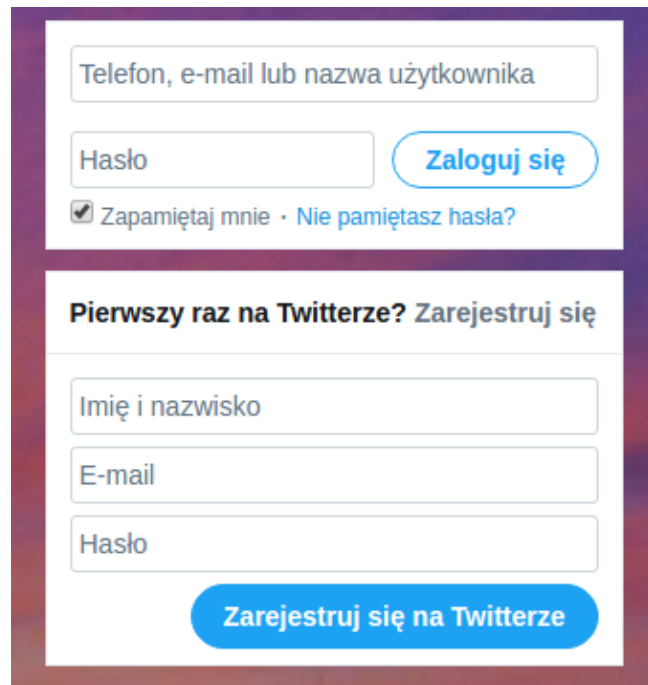
- Wejście:
 - Login (email),
 - Hasło.
- Weryfikacja, czy istnieje użytkownik i czy hasło jest poprawne.
- Blokowanie logowania, po nieudanych próbach.
- Słownikowe lub brute-force łamanie hasła.
- Komunikat o niepoprawnym logowaniu.
- Rozpoznanie, czy dany login istnieje.
 - Pierwsza faza łamania konta.
 - Ashley Madison.
- Analogicznie w funkcjonalności przypomnienia hasła.
 - Enumeracja loginów.



The image shows a user interface for login and registration, enclosed in a purple border. The top section is for login, featuring a text input field labeled "Telefon, e-mail lub nazwa użytkownika", a password input field labeled "Hasło", and a blue button labeled "Zaloguj się". Below the password field is a checkbox labeled "Zapamiętaj mnie" and a link "Nie pamiętasz hasła?". The bottom section is for registration, titled "Pierwszy raz na Twitterze? Zarejestruj się". It contains three input fields: "Imię i nazwisko", "E-mail", and "Hasło". At the bottom of this section is a blue button labeled "Zarejestruj się na Twitterze".

Uwierzytelnienie i sesja - Rejestracja

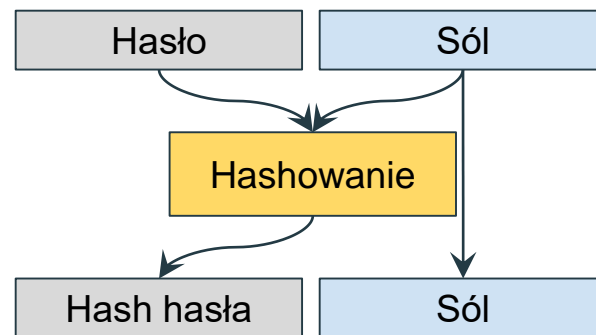
- Wejście:
 - Login (email),
 - Hasło,
 - Dane użytkownika.
- Weryfikacja, czy istnieje użytkownik.
- Komunikat po poprawnej rejestracji lub o błędzie.
- Rozpoznanie, czy dany login istnieje.
- Co z hasłem?
- Niedopuszczenie prostych haseł.
 - Wymagania dotyczące złożoności hasła.
- Sprawdzenie, czy hasło nie zostało złamane w innym serwisie.
 - <https://haveibeenpwned.com/passwords> (Troy Hunt)
 - "The original 306m hashes provided at the release of the service" (5GB)



The image shows two parts of the Twitter interface. The top part is the login form, which includes a text input for 'Telefon, e-mail lub nazwa użytkownika', a text input for 'Hasło', and a blue button labeled 'Zaloguj się'. Below the password input is a checkbox for 'Zapamiętaj mnie' and a link 'Nie pamiętasz hasła?'. The bottom part is the registration form, titled 'Pierwszy raz na Twitterze? Zarejestruj się'. It contains three text inputs: 'Imię i nazwisko', 'E-mail', and 'Hasło'. At the bottom of this section is a large blue button labeled 'Zarejestruj się na Twitterze'.

Uwierzytelnienie - Przechowywanie hasła

- Nie przechowuj haseł w formie jawnej (plaintext)! Nigdy!
- Nie przechowuj haseł w formie zaszyfrowanej. Gdzie jest klucz?
- Hashuj hasła! Jaka funkcja hashująca?
- MD5, SHA1, SHA256, ...
 - Rozmiar hasha - kolizyjność.
 - Szybkość.
- Sól hasła.
- Dedykowane algorytmy:
 - Bcrypt (moc obliczeniowa),
 - PBKDF2 (moc obliczeniowa),
 - Scrypt (moc obliczeniowa + pamięć),
 - libsodium (biblioteka, Argon2).



PHPBB2, PHPBB By Przemo, Joomla <1.0.13, PHPNuke, XOOPS
`md5(password)`

Joomla >= 1.0.13, Joomla 2.x
`md5(pass.salt(32))`

5f4dcc3b5aa765d61d8327deb882cf99

Uwierzytelnienie - Przechowywanie hasła

- Bcrypt

```
>>> import bcrypt
>>> bcrypt.hashpw("Twoje Hasło",bcrypt.gensalt(12))
'$2a$12$123456789012345678901ueEDm4W8S0bcR7tYCaovy5X64j.wKmA2'
```

- \$<version>\$<rounds>\$<saltaddon><pwhash>
 - rounds - trudność hasha,
 - saltaddon - sól,
 - pwhash - właściwy hash hasła.

Uwierzytelnienie - Przechowywanie hasła

- libsodium
- <https://download.libsodium.org/doc/>

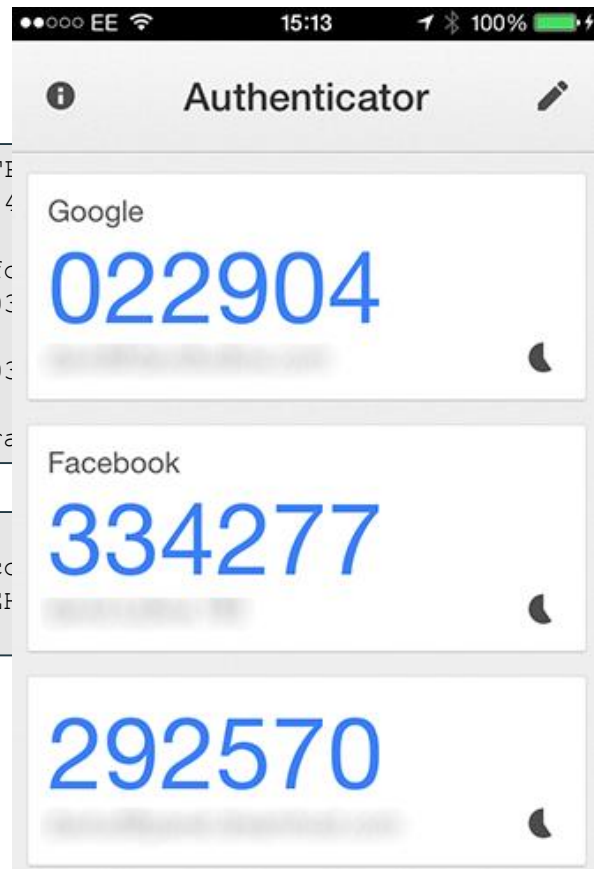
```
$hash_str = \Sodium\crypto_pwhash_str(  
    $password,  
    \Sodium\CRYPTO_PWHASH_OPSLIMIT_INTERACTIVE,  
    \Sodium\CRYPTO_PWHASH_MEMLIMIT_INTERACTIVE  
);  
  
if (\Sodium\crypto_pwhash_str_verify($hash_str, $password)) {  
    \Sodium\memzero($password);  
  
    // Hasło OK  
} else {  
    \Sodium\memzero($password);  
  
    // Hasło niepoprawne  
}
```


Uwierzytelnienie i sesja - 2FA

- Dwuskładnikowe uwierzytelnienie
 - Coś co mam,
 - Coś co wiem,
 - Coś czym jestem.
- Coś co wiem - hasło, coś co mam - komórka.
- Google Authenticator
 - <https://github.com/pyotp/pyotp>

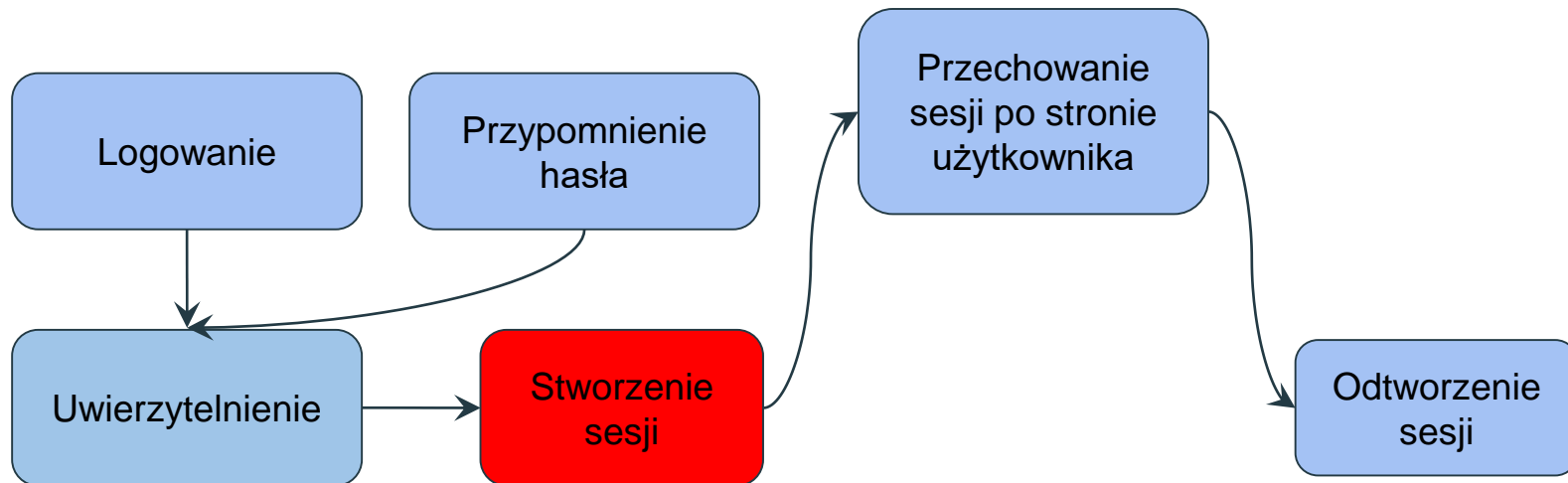
```
pyotp.totp.TOTP('JBSWY3DPEHPK3PXP').p  
>>> 'otpauth://totp/Secure%20App:alice'
```

```
totp = pyotp.TOTP('JBSWY3DPEHPK3PXP')  
totp.now() # => 4  
  
# OTP verified for 49203  
totp.verify(49203)  
time.sleep(30)
```



Sesja - Tworzenie nowej sesji

- Identyfikator sesji to sesja i konto użytkownika.
- Generowanie identyfikatora.
- Przechowywanie identyfikatora.



Sesja - Tworzenie nowej sesji

- Identyfikator sesji musi być losowy.
- Dlaczego? Bo jak zgadnę, to przejmę czyjąś sesję.
- Identyfikator musi być generowany przez serwer.
- Dlaczego? Bo mogę komuś ustawić identyfikator sesji (Session Fixation) i przejąć sesję.
- Identyfikator musi być generowany na nowo po każdym logowaniu.
- Dlaczego? Bo mogę komuś ustawić identyfikator sesji przed logowaniem, poczekać i przejąć uwierzytelnioną sesję.
- Identyfikator musi być unieważniany po pewnym czasie bezczynności.
- Dlaczego? Bo przechwycenie starej sesji prowadzi do przechwycenia uwierzytelnionej sesji. Bo jak ktoś się zapomni wylogować, to ktoś może przejąć jego sesję.

Sesja - Przechowywanie identyfikatora

- URL?
 - Przekazywanie sesji wraz z odnośnikiem.

- Ciasteczka

```
HTTP/1.1 200 OK
Date: Mon, 21 Aug 2017 07:49:33 GMT
Content-Type: text/html
Set-Cookie:
PHPSESSID=9aca96c4d386a8b46afc3023df8dddlb; path=/
Content-Length: 1989
```

```
GET / HTTP/1.1
Host: bootcamp.threats.pl
User-Agent: Mozilla/5.0 (Windows NT
10.0; WOW64; rv:54.0) Gecko/20100101
Firefox/54.0
Cookie:
PHPSESSID=9aca96c4d386a8b46afc3023df8dddlb
```

- Zabezpieczenie
 - HttpOnly
 - Secure

```
Set-Cookie: PHPSESSID=9aca96c4d386a8b46afc3023df8dddlb; path=/; HttpOnly; Secure
```

Sesja - HttpOnly (Serwer)

- Apache (>= 2.2.4)
 - Dodaj rozszerzenie **mod_headers.so**
 - Dodaj do pliku **httpd.conf**:
Header edit Set-Cookie ^(.)\$ \$1;HttpOnly;Secure # Apache >= 2.2.4*
Header set Set-Cookie HttpOnly;Secure # Apache < 2.2.4
- Tomcat
 - Edytuj plik definiujący kontekst: **/META-INF/context.xml**
 - Ustaw **useHttpOnly** na **true** (domyślnie jest **true**).

```
<Context useHttpOnly="true">  
...  
</Context>
```

Sesja - HttpOnly (Framework)

- Yii

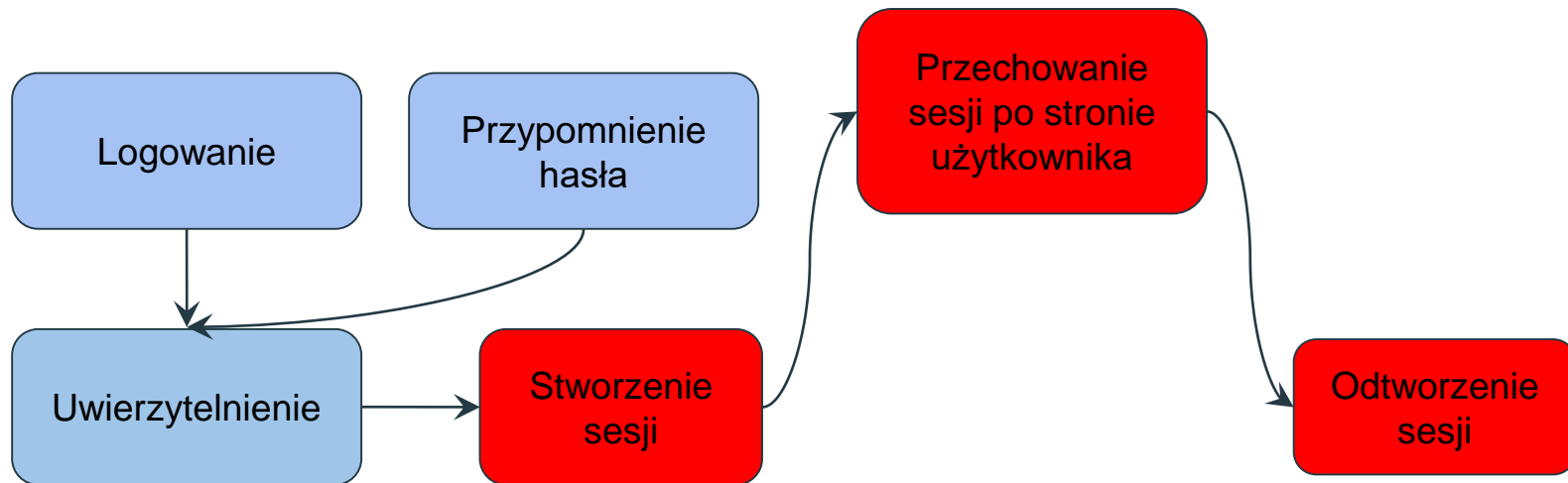
```
$options = array('httpOnly' => true, 'secure' => true);  
$cookieCollection['name'] = new CHttpCookie('name', 'value', $options);
```

- Spring

```
Cookie cookie = new Cookie("timestamp", new Long(new Date().getTime()).toString());  
cookie.setSecure(true);  
cookie.setHttpOnly(true);  
httpServletResponse.addCookie(cookie);
```

Sesja - Po stronie klienta

- Sesja przechowywana po stronie klienta.
- Identyfikator nie wystarczy.



Sesja - Zwyczajny JSON

- Sesja przechowywana w formacie JSON.

```
HTTP/1.1 200 OK
Date: Mon, 21 Aug 2017 07:49:33 GMT
Content-Type: text/html
Set-Cookie: Auth=eyJ1c2VybmFtZSI6ICJkb2hueSIsICJpZCI6IDF9; path=/; HttpOnly; Secure;
```

BASE64

```
{"username": "Johny", "id": 1001}
```

```
{"username": "Johny", "id": 1}
```

```
eyJ1c2VybmFtZSI6ICJkb2hueSIsICJpZCI6IDF9
```

```
GET / HTTP/1.1
Host: bootcamp.threats.pl
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
Cookie: Auth=eyJ1c2VybmFtZSI6ICJkb2hueSIsICJpZCI6IDF9
```


Sesja - JSON Web Token

- JWT (RFC 7515)
 - Wymiana uwierzytnionych danych.

Format: <Header>.<Payload>.<Signature>

eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFtcGxlIjMnNvbS9pc19yb290Ijp0cnV1fQ.dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk

- Nagłówek:
 - typ - rodzaj danych,
 - alg - algorytm uwierzytnienia wiadomości (MAC).

```
{"typ": "JWT", "alg": "HS256"}
```

- Dane - dowolna treść.

```
{"iss": "joe", "exp": 1300819380,  
  "http://example.com/is_root": true}
```

- MAC - poświadczenie, że wiadomość została zatwierdzona przez serwer.

Sesja - JSON Web Token

- Typy uwierzytelnienia:
 - HS256 - HMAC using SHA-256,
 - HS384 - HMAC using SHA-384,
 - HS512 - HMAC using SHA-512,
 - oparte na RSA,
 - oparte na EC,
 - none.
- Bezpieczeństwo oparte na tajności klucza uwierzytelniającego.

```
header = '{"alg":"HS256","typ":"JWT"}'  
payload = '{"username":"Johny","id":1001}'  
key = 'secretkey'
```

```
unsignedToken = encodeBase64(header) + '.' + encodeBase64(payload)  
signature = HMAC-SHA256(key, unsignedToken)
```

```
token = encodeBase64(header) + '.' + encodeBase64(payload) + '.' + encodeBase64(signature)
```

Sesja - JWT - Django REST

- Django REST Framework JWT
- <https://github.com/GetBlimp/django-rest-framework-jwt>
- <http://getblimp.github.io/django-rest-framework-jwt/>

settings.py

```
curl -X POST -d "username=admin&password=password123" http://localhost:8000/api-token-auth/
```

```
    'DEFAULT_AUTHENTICATION_CLASSES': (
```

```
        curl -H "Authorization: JWT <your_token>" http://localhost:8000/protected-url/
```

```
    )
```

urls.py:

```
from rest_framework_jwt.views import obtain_jwt_token
```

```
urlpatterns = [
```

```
    ...
    url(r'^api-token-auth/', obtain_jwt_token),
```

```
]
```

Sesja - JWT - Biblioteka JWT.io

- NodeJS
- <https://www.jsonwebtoken.io/>

```
var uuid = require('uuid');
var nJwt = require('njwt');

var claims = {"name": "Johny", "id": 1001}

var jwt =
nJwt.create(claims, "secret", "HS256");
var token = jwt.compact();
```

```
var nJwt = require('njwt');

nJwt.verify("eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1Y291bnkiLCJpZCI6MTAwMSwiYWV0dDlYWU3ODktY2RlNS00ZDY0LThtYTctMWEuODdhMzc2Y2Q2IiwiaWF0IjoxNTAzMzI0Mzc5LCJleHAiOjE1MDMzMjc5NzI9.z195Gnr2Jewlg1vHcZHDj1bFqNwnEGooUFLS6TpEXI","secret", 'HS512');
```

Sesja - JWT - Biblioteka JWT.io

- PHP
- <https://www.jsonwebtoken.io/>

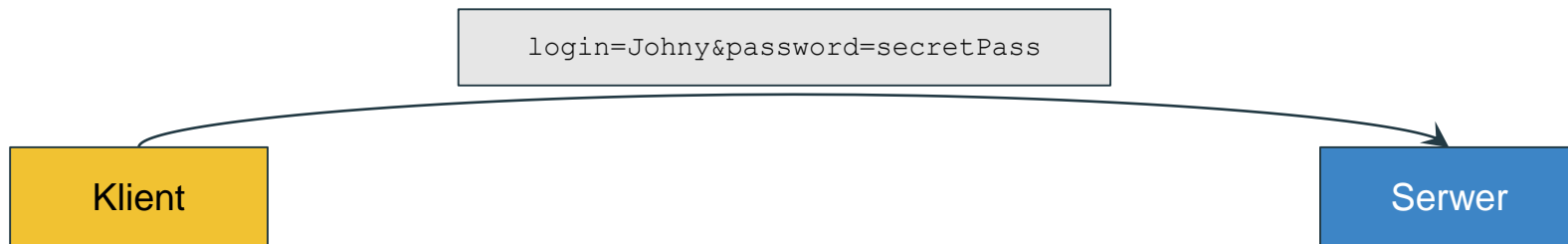
```
use \Firebase\JWT\JWT;

$token = array (
    'name' => 'Johny',
    'id' => 1001,
);

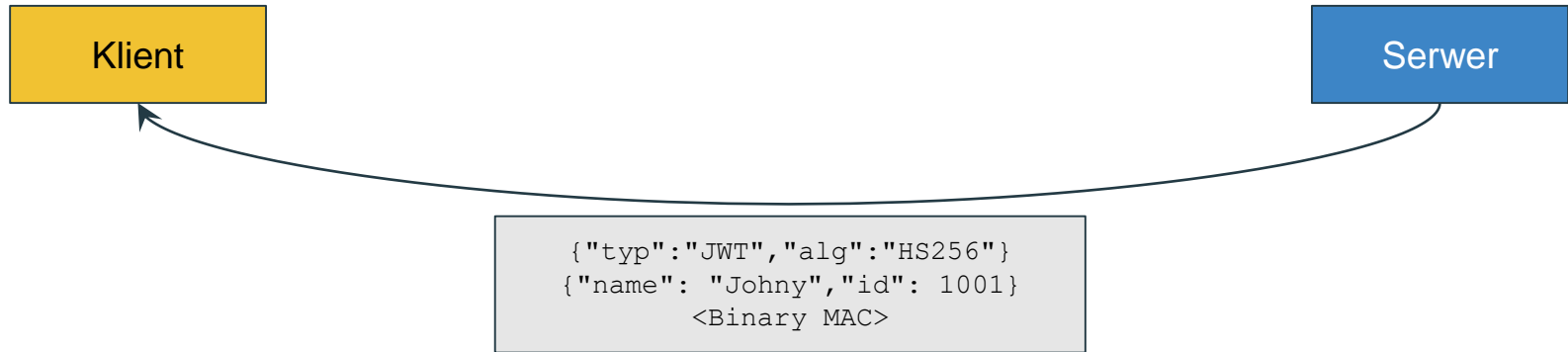
$jwt = JWT::encode($token, "secret");
```

```
use \Firebase\JWT\JWT;  
  
$decoded =  
JWT::decode("eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1eWl1IjoiaSm9obnkiLCJpZCI6MTAwMSwiYW50dDlYWU3ODktY2RlNS00ZDY0LTl1YTctMWEuODdhMzc2Y2Q2IiwiaWF0IjoxNTAzMzI0Mzc5LCJleHAiOjE1MDMzMjc5Nz19.z195GnR2Jewlg1vHcZHDj1bFqNwnEGooUFLS6TpEXI","secret", ['HS256']);
```

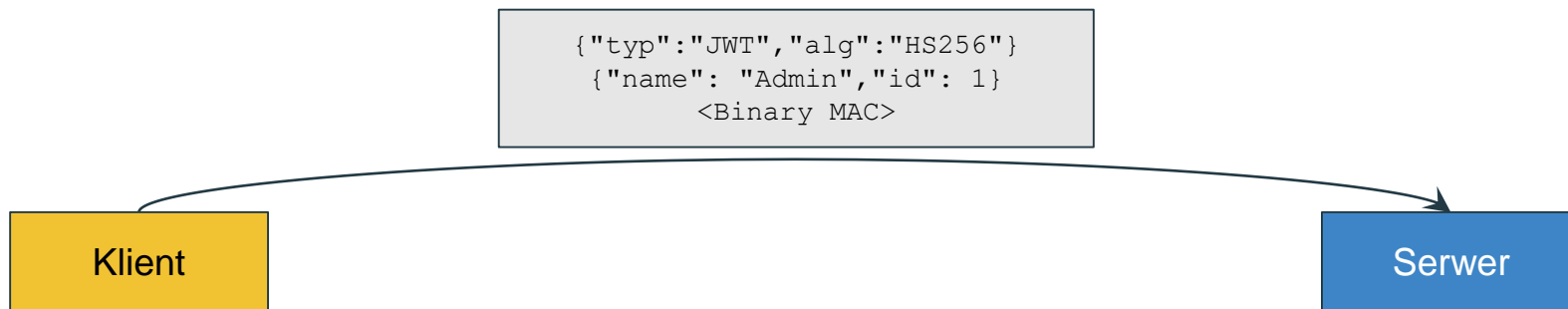
Sesja - JWT - Co może pójść nie tak?



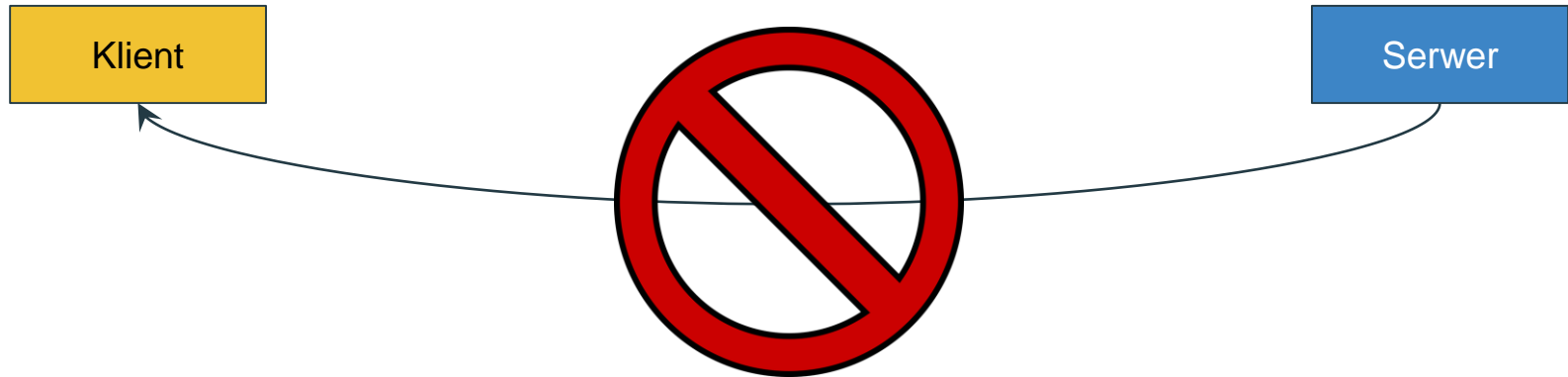
Sesja - JWT - Co może pójść nie tak?



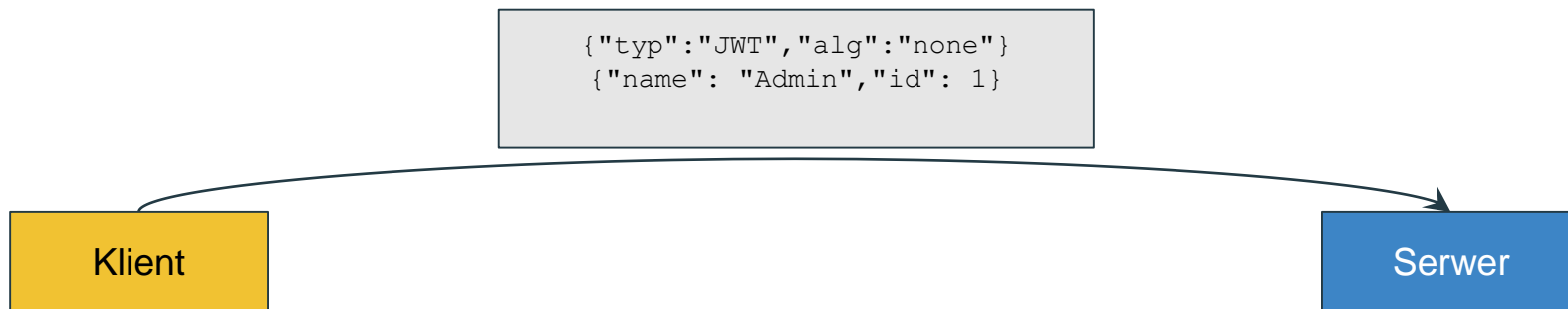
Sesja - JWT - Co może pójść nie tak?



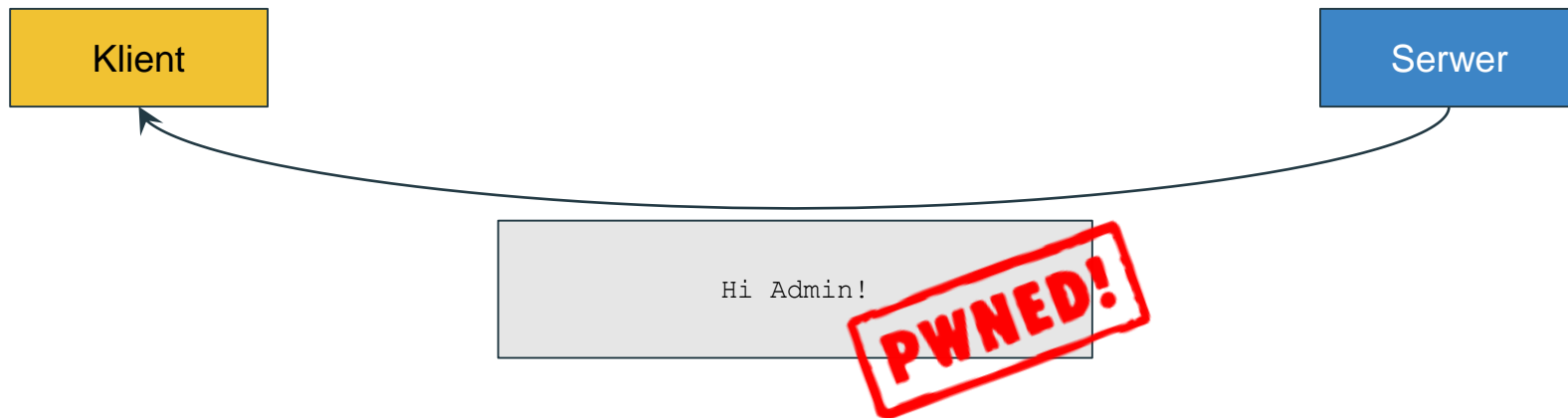
Sesja - JWT - Co może pójść nie tak?



Sesja - JWT - Co może pójść nie tak?



Sesja - JWT - Co może pójść nie tak?



Zarządzanie sesją - Podsumowanie

- Logowanie i rejestracja
 - Blokowanie kont po nieudanych próbach.
 - Jednolite odpowiedzi, nie zdradzające informacji o istniejących użytkownikach.
- Weryfikacja i przechowywanie hasła
 - Wymuszenie trudnego hasła (sprawdzenie, czy nie istnieje w bazie złamanych haseł).
 - Hashowanie haseł dedykowanymi algorytmami.
- Generowanie identyfikatora sesji
 - Losowość.
 - Generowanie nowej sesji po uwierzytelnieniu.
 - Unieważnienie sesji po wylogowaniu.
- Przechowywanie identyfikatora sesji
 - W ciasteczku, nie w URLu.
 - Zabezpieczenie ciasteczka.
 - Jako prosty i bezpieczny typ, np. liczba, słownik JSON, itp..
 - Wykorzystanie gotowego mechanizmu np. JWT.