M183 Applikationssicherheit Implementieren # 7

By Jürg Nietlispach

Recap # 6?

Recap # 6?

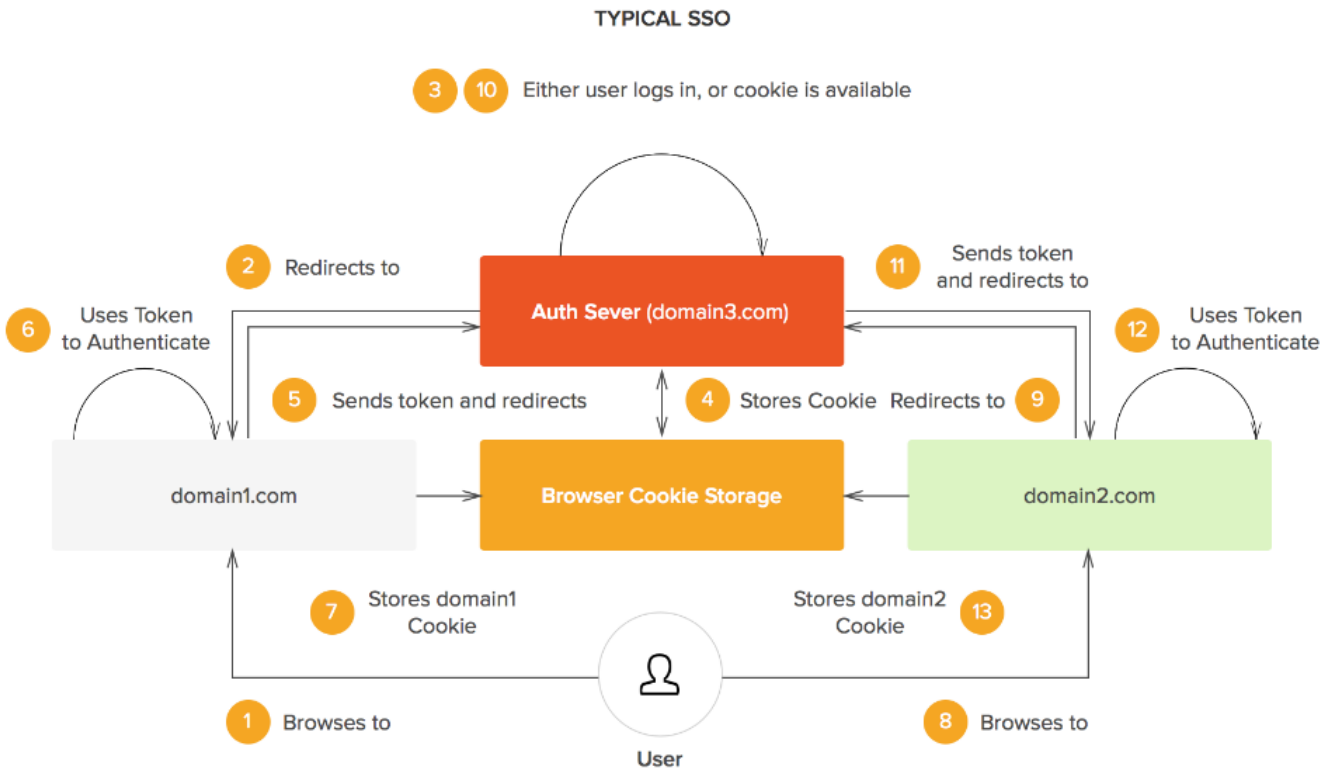
Single Sign On

- Types?
- Workflow?
- Benefits & Drawbacks?

Single Sign On - Types

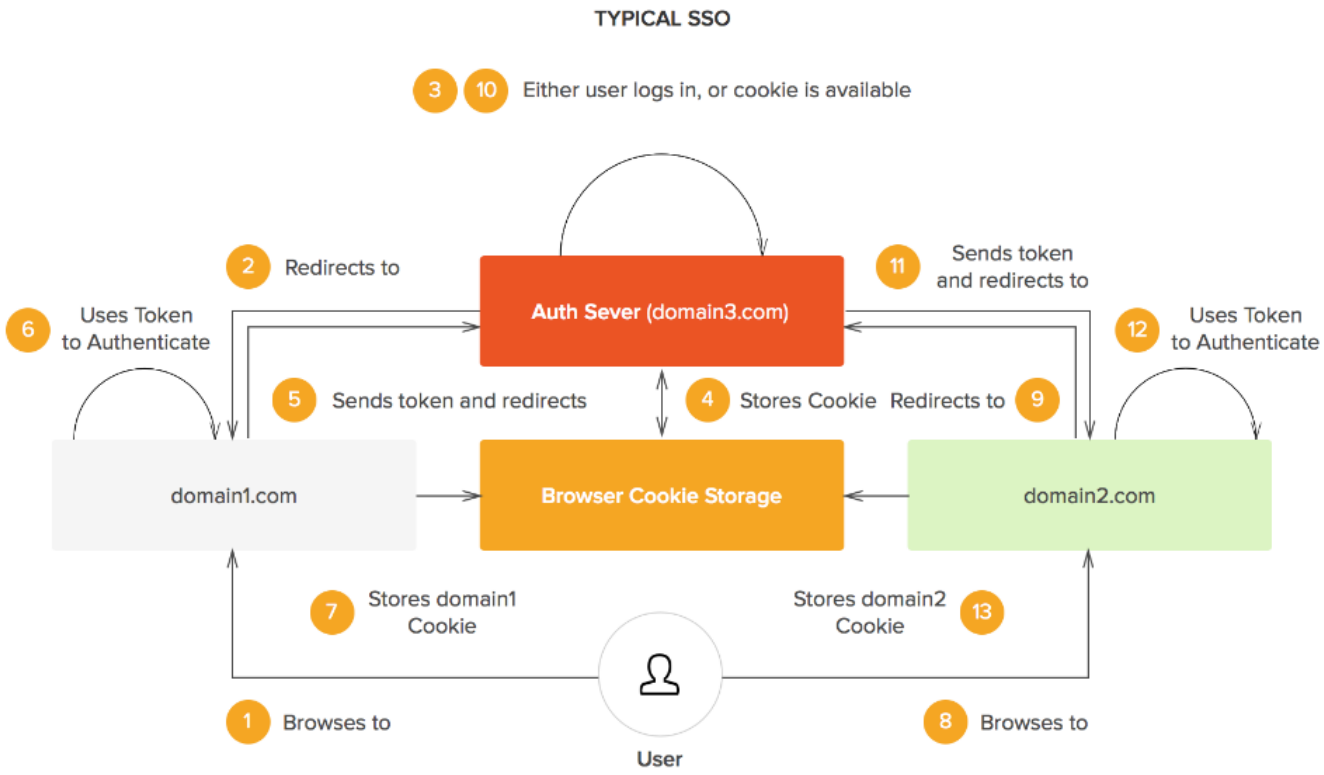
- **Portal Solutions (SAML, OAuth2, OpenID Connect, traditional HTTP-Cookie)**
- Local Solutions (z.B. Passwordmanager)
- Ticketing System (Kerberos)
- …
- (Public Key Infrastructure: Digitales Zertifikat)

Recap Single Sign On 1



1. User navigates to Login on domain1.com
2. domain1.com redirects to Auth Server (Login with Google-Button)
3. User logs in at Auth Server
    1. User gets Cookie for the Auth Server (future logins)
    2. User gets a signed Ticket for domain1.com
4. Client is redirected back to domain1.com with the signed Ticket
5. Information of the signed Ticket are used to check back at the Auth Server (from domain1.com) whether the user is loggedin (ID Token)
6. domain1.com can send a session cookie to the user
7. domain1.com can request additional user information (Claims) using the ID Token

Recap Single Sign On 2



8. User navigates to Login on domain2.com
9. domain2.com *redirects* to Auth Server (Login with Google Button)
10. User is **already loggedin** in at Auth Server
    8. User gets a signed Ticket for domain2.com
11. Client is *redirected* back to domain2.com with the signed Ticket
12. Information of the signed ticket are used to check back at the Auth Server (from domain2.com) whether the user is loggedin (ID Token)
13. domain2.com can send a session cookie to the user
14. domain2.com can request additional user information (Claims) using the ID Token

Benefits & Drawbacks

Benefits

- One Authentication-Procedure for N Systems
  - Time
  - Security
  - less (weak) passwords per user
  - May reduce phishing attacks

Drawbacks

- Attacker has instantly access to all services as soon as he has the credentials
- Sign off (Time-Out)?
- Availability of a SSO-Service?

Übung – Single Sign On mit Google API

1. Setup Auth Server (Google API Account Infos) for Client / Button Integration (Google-Account wird benötigt)
   https://developers.google.com/identity/sign-in/web/devconsole-project

2. Setup Client / Button
   https://developers.google.com/identity/sign-in/web/sign-in

3. Get Profile Infos
   https://developers.google.com/identity/sign-in/web/people

4. **Authenticate with Backend Server (in order to get a session)**
   https://developers.google.com/identity/sign-in/web/backend-auth

5. **Sign Out Button**
   https://developers.google.com/identity/sign-in/web/sign-in

Pre-Setup Backend Server

1. Extend onSignOn() Javascript – Function:

```
function onSignOn(googleUser){

        var id_token = googleUser.getAuthResponse().id_token;

        var xhr = new XMLHttpRequest();
        xhr.open('POST', 'http://localhost:PORT/Home/SSOTokenSigin');
        xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
        xhr.onload = function() {
          console.log('Signed in as: ' + xhr.responseText);
        };
        xhr.send('idtoken=' + id_token);

 }
```

Setup Backend Server

1. Create a Route for id_token validation (http://localhost:PORT/api/validateidtoken/{id_token})

2. Validate token (integrity) using .NET Oauth Client Libary:
    1. https://developers.google.com/api-client-library/dotnet/apis/oauth2/v2

3. Check ID-Token at
    1. https://www.googleapis.com/oauth2/v3/tokeninfo?id_token={ID_TOKEN}

4. If token is valid -> return SESSION

Logout Button

1. Place Logout-Button

```
<a href="#" onclick="signOut();">Sign out</a>
```

2. Integrate Javascript Snippet

```
function signOut() {
  var auth2 = gapi.auth2.getAuthInstance();
  auth2.signOut().then(function () {
    console.log('User signed out.');
  });
}
```

Authentication workflows complete?

Consider 2 Factor Authentication / Single Sign On. How to implement

- Account activation?
- Password forgotten?
- Force to refresh password?
- Access highly sensitive data?

Authentication workflow complete?

Consider 2 Factor Authentication / Single Sign On.

- Account activation
  -> Account/Password activation / recovery

- Password forgotten?
  -> Account/Password recovery

- Force to refresh password / passphrase?
  -> Password/Passphrase reset (length and complexity)

- Access highly sensitive data?
  -> Re-Authentication

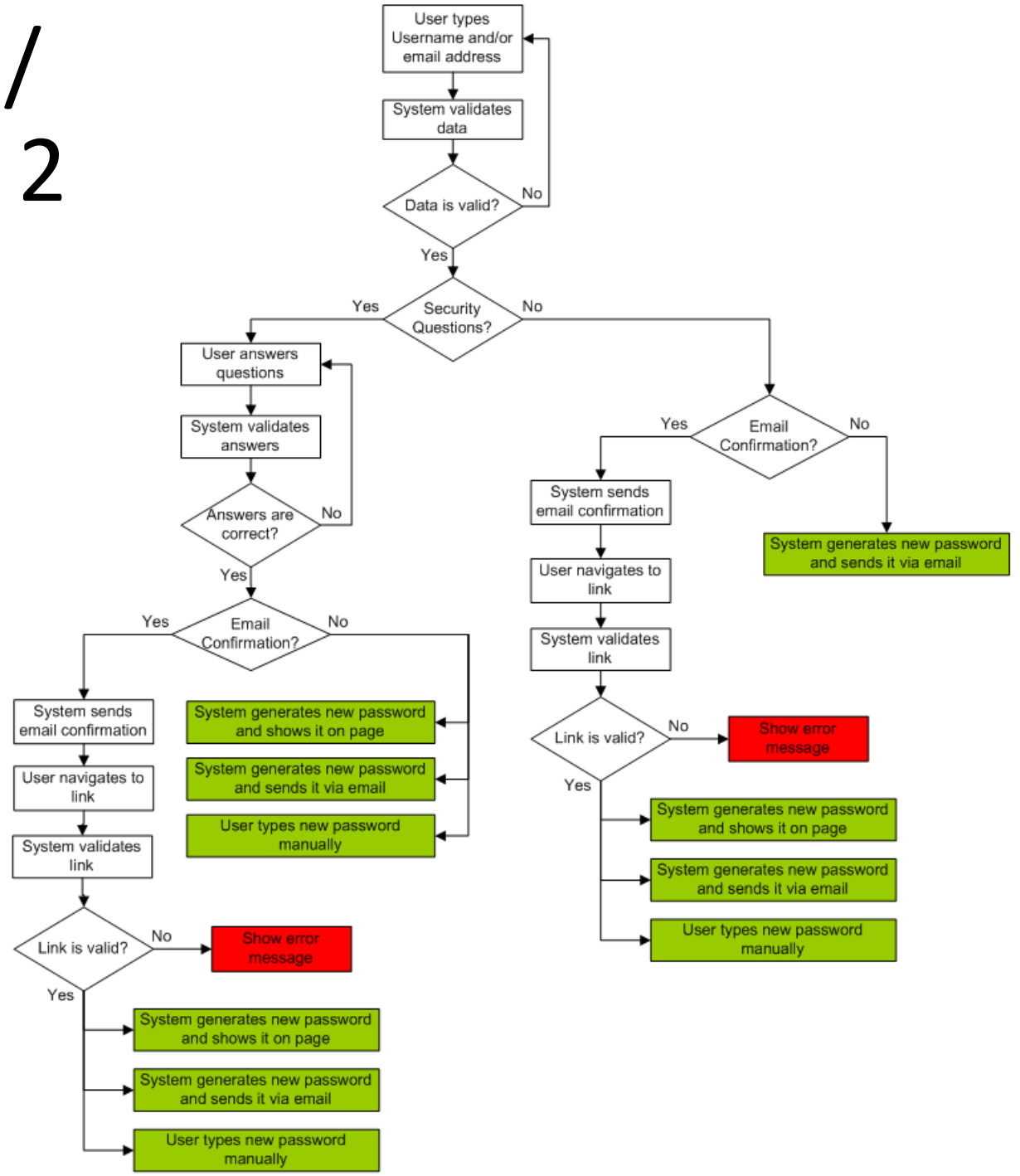
Account Activation / Password Recovery

There is no industry standard for implementing a Forgot Password feature!

1. Identity Data (for recovery, i.e. username)
2. Security Questions (and verification)
3. Send a Token / Password Reset Link (containing the token) over a Side-Channel (SMS, Email)
4. Logging

Account Activation / Password Recovery 2

Only one possible Workflow!



Password / Passphrase Reset

- Allow user to change password in the existing session
- Ensure the user changes their password and does not simply surf to another page in the application.
- The reset must be performed before any other operations can be performed by the user.

Passwords

Should have a minimum level of complexity that makes sense for the application and its users.

Complexity rules:

- at least 1 uppercase character (A-Z)
- at least 1 lowercase character (a-z)
- at least 1 digit (0-9)
- at least 1 special character (see https://www.owasp.org/index.php/Password_special_characters)

- **at least 10 characters (*), unsecure otherwise (may be a problem! -> see passphrases)**
- at most 128 characters
- not more than 2 identical characters in a row (e.g., 111 not allowed)

Passphrases

While minimum length enforcement may cause problems with memorizing passwords among some users, applications should encourage them to set *passphrases* (sentences or combination of words) that can be much longer than typical passwords and yet much easier to remember.

Passphrases shorter than 20 characters are usually considered weak if they only consist of lower case Latin characters (*)

Passwords vs Passphrases (*)

Passwords
26 (lowercase) + 26 (uppercase) + 10 (digits) + 33 (special characters) => max. 95 possibilities per position. In Total 95^10 => 58*10^18 combinations.

With 10^9 combinations per Second on one machine -> 58*10^9 Seconds for all combinations

Passphrases
26 (lowercase) => max. 26 possibilities per positon. In Total 26^20 => 19*10^27 combinations

With 10^9 combinations per Second on one machine -> 19*10^18 Seconds for all combinations

Re-Authentication

It's important to **require the current credentials** for an account **before updating** sensitive account information such as the user's password, user's email, or before sensitive transactions, such as shipping a purchase to a new address.
Without this countermeasure, an attacker may be able to execute sensitive transactions through a CSRF or XSS attack without needing to know the user's current credentials (i.e. the Session-Id).