M183 Applikationssicherheit Implementieren # 5

By Jürg Nietlispach

Recap # 4?

Recap # 4

1. Motivation 2-Factor-Authentication: defense in depth principle
2. 2-Factor Authentication paradigm
    1. One thing you know & one thing you have

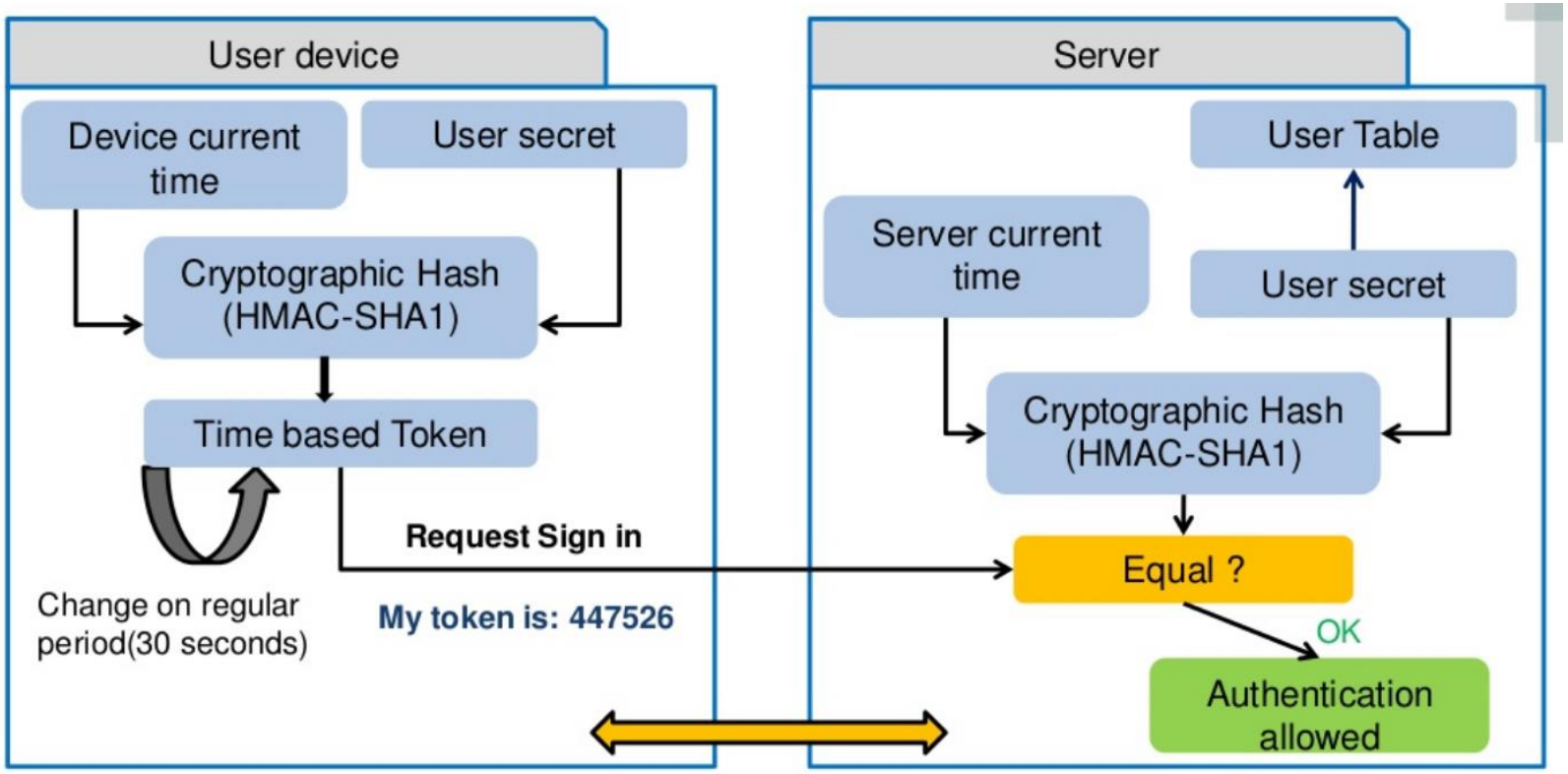
3. OTP & TOTP
    1. Constraints of System, Setup, Tokens?
    2. Consequences of System, Setup, Tokens?

Übungen 2-Factor Auth

1. OTP via SMS / Email (finish war Hausaufgabe)
2. TOTP via Google Authenticator

Recap: 2-Factor Auth using TOTP (Google-Authenticator)

1. Device and Backend-System agree on a Secret Key (User – Secret, via QR-Code)
2. Device generates Token on Device Current Time
3. Server generates Token on Server Current Time
4. Server compares Tokens and the user can enter the System.



Übung 2-Factor Auth: TOTP

Zeit: 45’

Google Authenticator

1. Aus App-Store downloaden

Applikation (.NET MVC) - Keyaustausch

1. TOTP-Function / Library suchen und installieren (Otp.NET)
2. Secret generieren
3. QR-Code generieren
4. Mit Google-Authenticator-App scannen

Applikation (.NET MVC) - Authentifizierung

1. Bei Form-Submission check von Username & Passwort. Stimmen diese:
    1. (Drittes) Input-Feld für OTP anzeigen
2. Form-Sumission check von Usernamen, Passwort & TOTP

Setup Übungen

1. Alle tragen sich in .git-Liste ein!
2. Alle bisherigen und zukünftigen Übungen **MÜSSEN je Person versioniert** sein!
3. Regelmässiger commit mit dem **aktuellen Arbeitsschritt** und **individuelle Kommentare** je Codeblock. Zweck: Nachvollziehbarkeit & Bewertung
4. Konzentriertes & stilles Arbeiten. Fragen werden individuell beantwortet.
5. Gänge zum Brünneli & Toilette -> Pausen!

2-Factor-Authentication – the perfect solution?

Consider: one user has many tools to use …

What are the issues? And Why?

Usability Issues with 2-Factor-Authentication

1. Separate Credentials for every Portal? -> In practice NO!
2. Time consuming to do 2-Factor-Authentication for every tool the user uses
3. N Tools -> N Logins! (-> Minimize Attack Surface Area!)
4. N Tools -> N Logins -> N-times Password- and Profilechanges!
5. …

Solution?

Single Sign On!
(in combination with 2-Factor-Auth)

*"**Single sign-on** (**SSO**) is a property of access control of multiple related, yet independent, software systems. With this property, a user logs in with a single ID and password to gain access to a connected system or systems without using different usernames or passwords, or in some configurations seamlessly sign on at each system."*

Single Sign On - Types

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

Single Sign On – Portal Solution 1

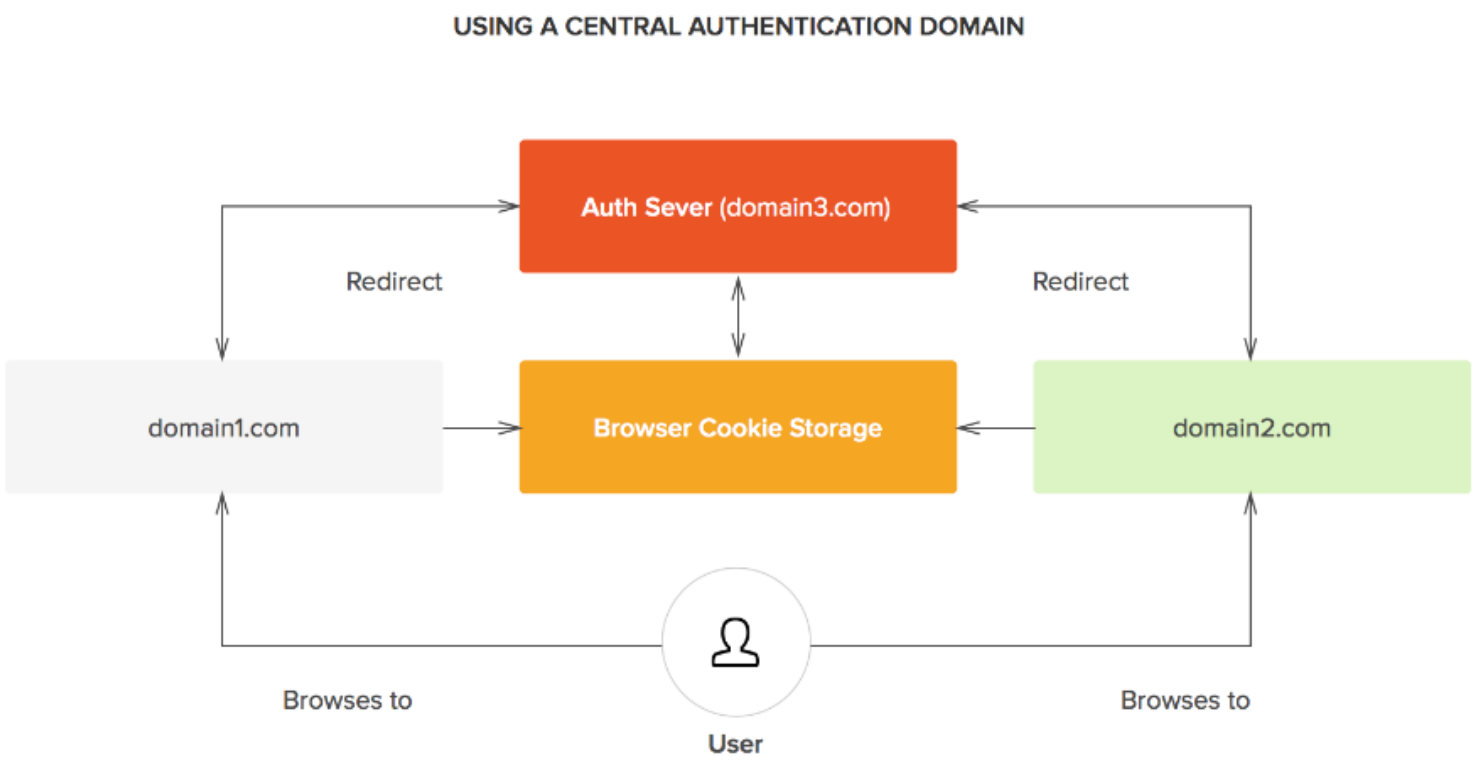
Motivation: Already Logged-In User on domain1.com automatically login on domain2.com

Solution: **share session information** across different domains

Single Sign On – Portal Solution 2

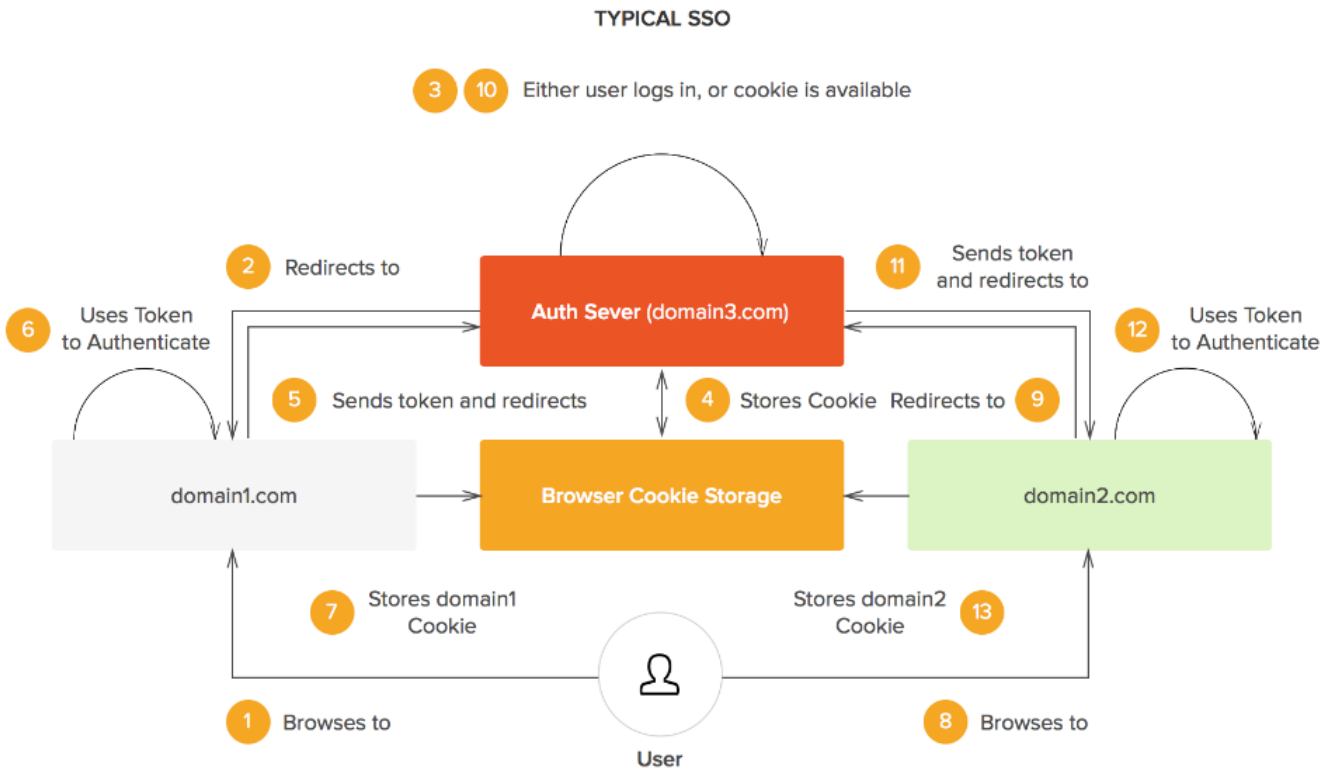
Problem: *same origin policy*. This policy dictates that cookies (and other locally stored data, JWT) can **only be accessed by its creator**! (i.e. domain1.com)

Single Sign On – Portal Solution 3



Solution 1: Central Authentication Provider

Single Sign On – OpenId Connect



1. Authenticate user on domain1.com
2. domain1.com redirects to Auth Server
3. User logs in at Auth Server
    1. User gets Cookie for the Auth Server (future logins)
    2. User gets an identiy token
4. Client is redirected back to domain1.com with the identity token
5. Informations of the identity token are used to check back at the Auth Server 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

Single Sign On – OpenId Connect

1. Create Link (i.e. login with google button) on domain1.com (containing redirect_uri, client_id, etc)

```
HTTP/1.1 302 Found
Location: https://openid.c2id.com/login?
          response_type=code
          &scope=openid
          &client_id=s6BhdRkqt3
          &state=af0ifjsldkj
          &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

1. This link redirects the user to the Auth Server User logs in at Auth Server
    1. The user gets Cookie for the Auth Server (future logins)
    2. Client is redirected back to domain1.com with an identity token

```
HTTP/1.1 302 Found
Location: https://client.example.org/cb?
          code=SplxlOBeZQQYbYS6WxSbIA
          &state=af0ifjsldkj
```

Single Sign On – OpenId Connect

1. Informations of the identity token are used to check back at the Auth Server whether the user is loggedin

```
POST /token HTTP/1.1
Host: openid.c2id.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

grant_type=authorization_code
 &code=SplxlOBeZQQYbYS6WxSbIA
 &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
  "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
    yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
    NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZ
    fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
    AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
    Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
    NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
    QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
    K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
    XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
  "access_token": "SlAV32hkKG",
  "token_type": "Bearer",
  "expires_in": 3600,
}
```

Single Sign On – OpenId Connect

1. Informations of the identity token are used to check back at the Auth Server whether the user is loggedin
2. domain1.com can send a session cookie to the user
3. Additionally, domain1.com can now get further «Claims» about the user at the Auth Server (additional user infos) using the id_token

Single Sign On

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?