SINGLE-SIGN-ON
FOR WEB APIS

Niko Köbler
Software-Architect, Developer & Trainer

niko@n-k.de | www.n-k.de | @dasniko
Niko Köbler 🌆
@dasniko

Doing stuff with computers, serverless in the Cloud... mostly (functional) on the JVM, Co-Lead of @JUG_DA, Speaker, Author, Conference-Addict

Darmstadt, Germany · n-k.de

@dasniko
SECURITY

OWASP
AUTHENTICATION

I don't know who you are!

AUTHORIZATION

I know who you are, but you're not allowed!
SECURITY

Same centralized approach for Web-APIs, Microservices and Monoliths!?
SIMPLY
SECURE
Log in to Twitter

Phone, email or username

Password

Log In  ✓ Remember me  ·  Forgot password?

New to Twitter? Sign up now »

Already using Twitter via text message? Activate your account »
APIS ARE OFTEN DISTRIBUTED

ONE DOES NOT SIMPLY LOGIN
TOKEN

TO THE RESCUE
<saml:Assertion
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    ID="b07b804c-7c29-ea16-7300-4f3d6f7928ac"
    Version="2.0"
    IssueInstant="2004-12-05T09:22:05">
    <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"/>
    <saml:Subject>
        <saml:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
            b07b804c-7c29-ea16-7300-4f3d6f7928ac
        </saml:NameID>
        <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"
            InResponseTo="aaf23196-1773-2113-474a-fe114412ab72"
            Recipient="https://sp.example.com/SAML2/SSO/POST"
            NotOnOrAfter="2004-12-05T09:27:05"/>
    </saml:Subject>
    ...
OAUTH2

Authorization, NOT Authentication!

The OAuth 2.0 authorization framework enables a 3rd-party application to obtain limited access to an HTTP service.

IETF, RFC 6749, 2012
ABSTRACT PROTOCOL FLOW

1. Authorization Request
2. Authorization Grant
3. Authorization Grant
4. Access Token
5. Access Token
6. Protected Resource

Application (Client)

User (Resource Owner)

Authorization Server

Resource Server

Service API
## GRANT TYPES

<table>
<thead>
<tr>
<th>Grant Type</th>
<th>Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization Code</td>
<td>Web, Apps</td>
</tr>
<tr>
<td>Implicit</td>
<td>JavaScript, etc.</td>
</tr>
<tr>
<td>Resource Owner Password Credentials</td>
<td>Apps</td>
</tr>
<tr>
<td>Client Credentials</td>
<td>Web</td>
</tr>
<tr>
<td>Refresh Token</td>
<td>Web, Apps</td>
</tr>
</tbody>
</table>
ACCESS TOKEN

{
    "access_token": "6041a9d7-8c39-4945-b7c6-eaf7bd5d0907",
    "token_type": "Bearer",
    "expires_in": 3600,
    "refresh_token": "e339b569-6d95-482d-9534-5c0147136ab0"
}
OIDC

OpenID Connect - NOT OpenID

Authentication layer on top of OAuth 2.0

- verify the identity of an end-user
- obtain basic profile information about the end-user
- RESTful HTTP API, using JSON as data format
- allows clients of all types (web-based, mobile, JavaScript)

OpenID Foundation, 2014
OIDC

OpenID Connect adds the \textit{IDENTITY TOKEN}

\begin{verbatim}
{
    "access_token": "6041a9d7-8c39-4945-b7c6-eaf7bd5d0907",
    "token_type": "Bearer",
    "expires_in": 3600,
    "identity_token": "???",
    "refresh_token": "e339b569-6d95-482d-9534-5c0147136ab0"
}
\end{verbatim}
JWT

JSON WEB TOKEN

RFC 7519 Standard, 2015
JSON WEB TOKEN

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Base64 encoded
JSON WEB TOKEN

Encoded

Decoded

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

VERIFY SIGNATURE

```javascript
HMACSHA256(
    base64UrlEncode(header) + "." +
    base64UrlEncode(payload),
    secret
) □secret base64 encoded
```

☑️ Signature Verified
JSON WEB TOKEN

PAYLOAD DATA

{
  "sub": "1234567890",
  "iss": "https://sso.myapi.com",
  "aud": "myApi",
  "exp": 1479814753,
  "name": "John Doe",
  "admin": true
}

Reserved claims:

sub, iss, aud, exp
ACCESS TOKEN

```json
{
    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "token_type": "Bearer",
    "expires_in": 3600,
    "identity_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```
TOKENS

Base for access on secured resources.

A token is signed and contains all necessary information about the user and its roles.

Kinds: Identity-, Refresh-, Offline- and Accessstokens

Send in Bearer format:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkXVCJ9...
```

Have a TTL!

Must be revocable!
TOKEN USAGE

Stateless sessions

Passing identity to 3rd parties

Token exchange
WHAT DOES JAVA OFFER?
WHAT DOES JAVA OFFER?

JAVA EE

nothing useful so far

JAAS?
(proprietary)

Java EE 8 with Security API, JSR-375
WHAT DOES JAVA OFFER?

SPRING SECURITY

good, powerful

Spring Cloud Security / OAuth2
WHAT DOES JAVA OFFER?

APACHE SHIRO

https://shiro.apache.org

OAuth2? / OIDC?
WHAT DOES JAVA OFFER?

APACHE OLTU

https://oltu.apache.org/

OAuth2 / OIDC / JWT

Who knows this project?
WHAT DOES JAVA OFFER?

PAC4J

http://www.pac4j.org

The Java security engine to protect all your web applications.

Available for most frameworks/tools:
J2E • Spring Web MVC (Spring Boot) • Spring Security (Spring Boot) • Shiro
Play 2.x • Vertx • Spark Java • Ratpack • Undertow
CAS server • JAX-RS • Dropwizard • Knox • Jooby
WHAT DOES JAVA OFFER?

JWT LIBRARIES

github.com/auth0/java-jwt

bitbucket.org/b_c/jose4j

bitbucket.org/connect2id/nimbus-jose-jwt

github.com/jwtk/jjwt
AND THE **ECOSYSTEM**?

**SAAS? AAAS?**

Auth0
auth0.com

AWS Cognito
aws.amazon.com/cognito

Stormpath
stormpath.com

*BUT*: you have to outsource your users personal data!
So, what to do?

Develop on your own?

How? Much effort!
INTEGRATED SSO AND IAM FOR BROWSER APPS AND RESTFUL WEB SERVICES
KEYCLOAK

- JBoss
- since ~2013
- Open Source Software
- hosted at GitHub
- very active Community
  (commits, pullrequests, mailinglists)
- constant and regular feature- and bugfix-releases
current version: 3.1.0.Final
- good & comprehensive documentation
#FEATURES

Single-Sign-On, Single-Sign-Out, Self-Registration, Forgot Password, Verify User/Email, TOTP, various Verification (Work-)Flows, Customer Attributes, Custom Federation Provider, SPIs, Social Logins, Custom Themes, JWT, OAuth2, Bearer Token, Open ID Connect (OIDC), SAML, LDAP/AD-Integration, Account Management, Management Console, CORS handling, Impersonation, etc...
JBoss EAP / Wildfly, Tomcat, Jetty, JBoss Fuse, Spring Boot, Spring Security, JavaScript, Apache Karaf, Servlet Adapter, Apache mod openid connect, Keycloak Proxy Server, own implementations
DEMOS

Spring Boot ("full" web app)

dasniko/keycloak-springboot-demo

Java EE (JBoss Wildfly, REST-only services)

dasniko/keycloak-javaee-demo

React.JS SPA (JavaScript client app)

dasniko/keycloak-reactjs-demo
CLOUD

AMAZON API GATEWAY

AWS IAM

Amazon Cognito / STS

Custom Authorizer (AWS Lambda)
CUSTOM AUTHORIZER

Verify token send in HTTP Header

Lambda function will be invoked with event:

```json
{
    "type": "TOKEN",
    "authorizationToken": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    "methodArn": "arn:aws:execute-api:eu-central-1:012345678910:v7twmt"
}
```
CUSTOM AUTHORIZER

Lambda function will create a Principal and an AWS Policy:

```json
{
    "principal": "1234567890",
    "policy": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "execute-api:Invoke",
                "Effect": "Allow",
                "Resource": "arn:aws:execute-api:eu-central-1:012345678910:vn1v2v3v4v5"
            }
        ]
    }
}
```

will be cached
API GATEWAY AUTHORIZATION

Client sends a request with a Bearer Token to the API Gateway. The API Gateway checks the Token and forwards the request to the Lambda Auth Function. The function evaluates the context and the Token to determine if the request is allowed. If the policy is evaluated as 'Allowed', the request is forwarded to the Amazon Lambda Functions. If the policy is evaluated as 'Denied', the request is returned with a 403 error. The policy is also cached in a Cache storage. The Lambda Functions provide the endpoint for the request, which is available publicly.
DEMO

Custom Authorizr AWS Lambda Function
dasniko/jwtAuthorizr
Thank you!

Niko Köbler
Software-Architect, Developer & Trainer

ANY QUESTIONS?


niko@n-k.de | www.n-k.de | @dasniko