AXXES
IT CONSULTANCY

Spring Security @Traineeship 2019

AXXES
IT CONSULTANCY

What can you expect

1. **Basic security concepts**
   - PKI
   - HTTPS
2. **Authentication methods**
   - Basic auth / OAuth2
3. **Spring Security**
4. **JWT tokens**

Getting acquainted

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

whoami

Gert-Jan Heireman

Prof. Bach. Toegepaste Informatica @ KdG

Master Toegepaste Informatica @ KUL

First project HealthConnect (3y)

gert-jan.heireman@axxes.com

AXXES
IT CONSULTANCY

Who are you?

Background?

- Studies
- Stage
- …

Experience with Spring?

AXXES IT CONSULTANCY

Useful links

Homepage

https://spring.io/

Docs

https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/

Single source of truth!

Let’s get started!

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

Difference?

Authentication

Authorization

AXXES
IT CONSULTANCY

Difference?

Authentication
-> Who is who

Authorization
-> What a user can do

Public key infrastructure (PKI)

The security backbone of the internet

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

Why?

Proof of identity

Persons, devices, services, ...

Encryption

Symmetric & asymmetric

Privacy

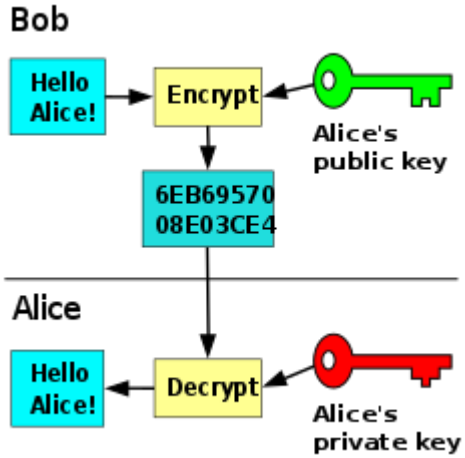
Hide from the gouverment (maybe)

=> **enables HTTPS**

How?

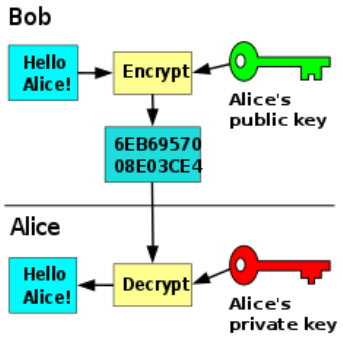
Private and public keys

- Content encrypted with Public Key can only be read by Private Key holder.
- Content encrypted with Private Key can be read by everyone with Public Key.



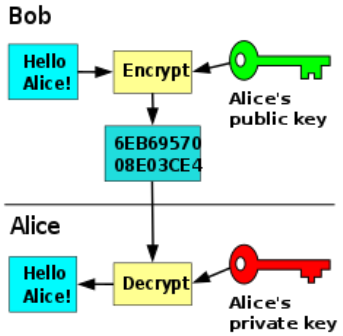
How?

How do we know that Alice is really Alice and not the NSA?



How?

How do we know that Alice is really Alice and not the NSA?



Certificates!

A certificate is a Public Key with more information about the entity (Alice).

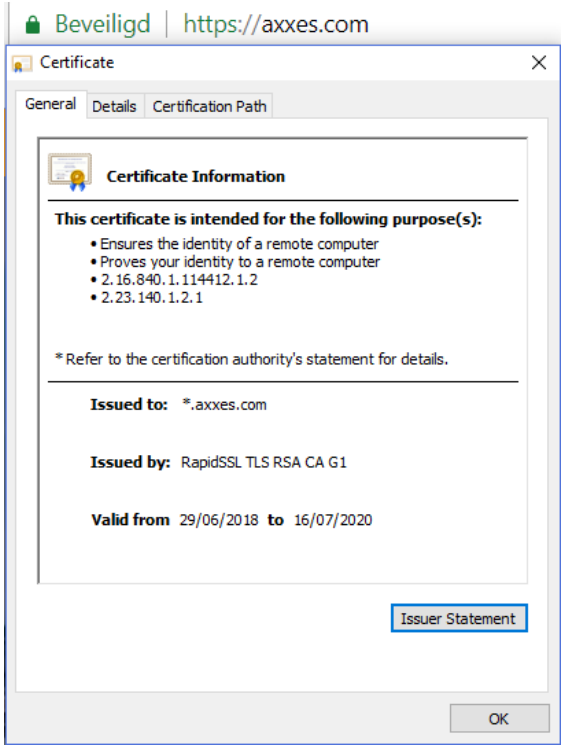
Certificate

Certificates are public keys with more information about the entity

Everyone can create Public/Private keypairs

Certificate authorities (CA) link keypair to entity

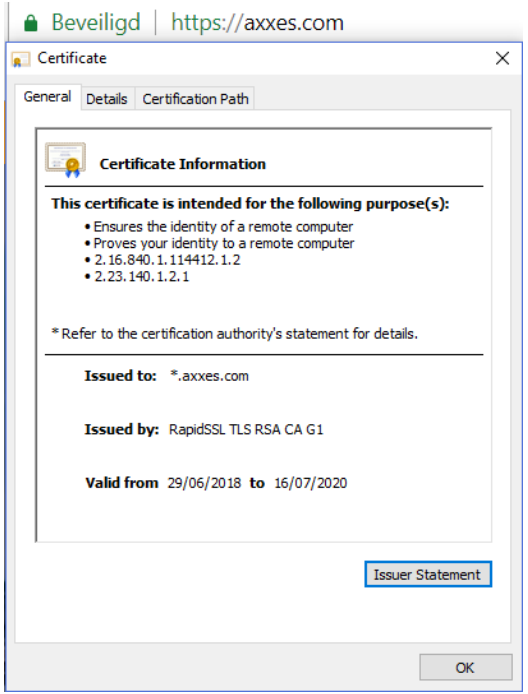
- CA's are a business
- Does background check
- Creates certificate
- Browsers trust the CA



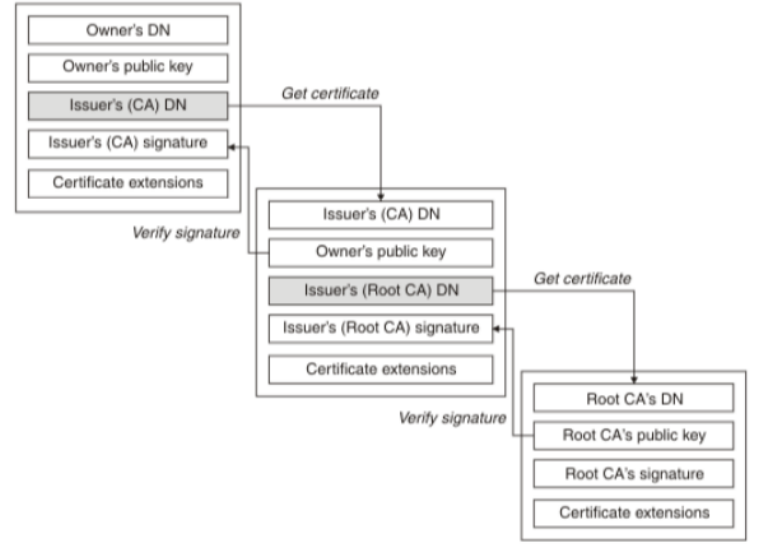
AXXES
IT CONSULTANCY

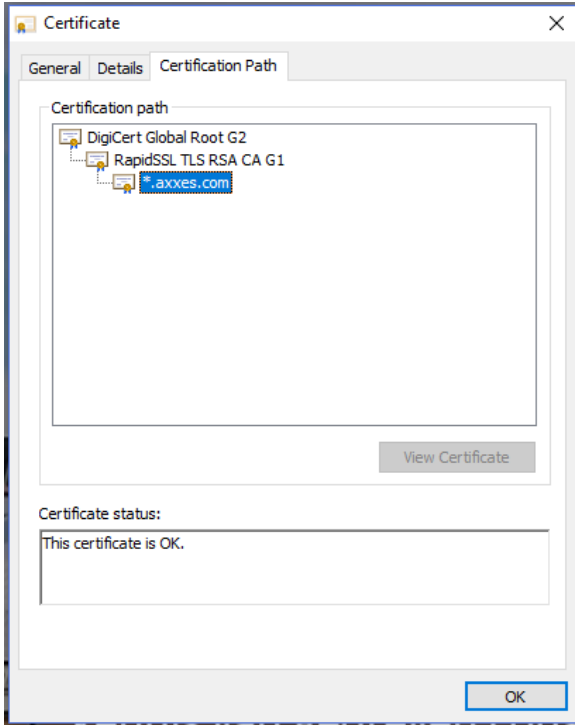
Certificate

How do we know that a certificate is the real certificate?



Certificate chain





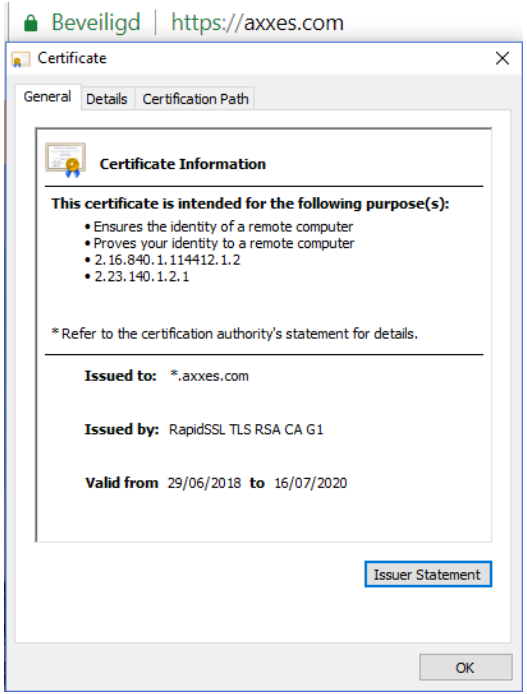
Certificate

Browsers trust CA's

But what if a CA is malicious?

What if a Private key gets stolen?

Bad luck.. Thats why it is also called the web of trust.



AXXES IT CONSULTANCY

Web of trust

Certificate issues

Recently in Kazachstan

https://tweakers.net/nieuws/155336/isps-kazachstan-moeten-https-verkeer-onderscheppen-op-last-van-overheid.html

Iranian hacker

https://tweakers.net/nieuws/76444/iran-gebruikt-nederlands-certificaat-om-gmail-te-onderscheppen.html

HTTPS

IT IS ABOUT PEOPLE

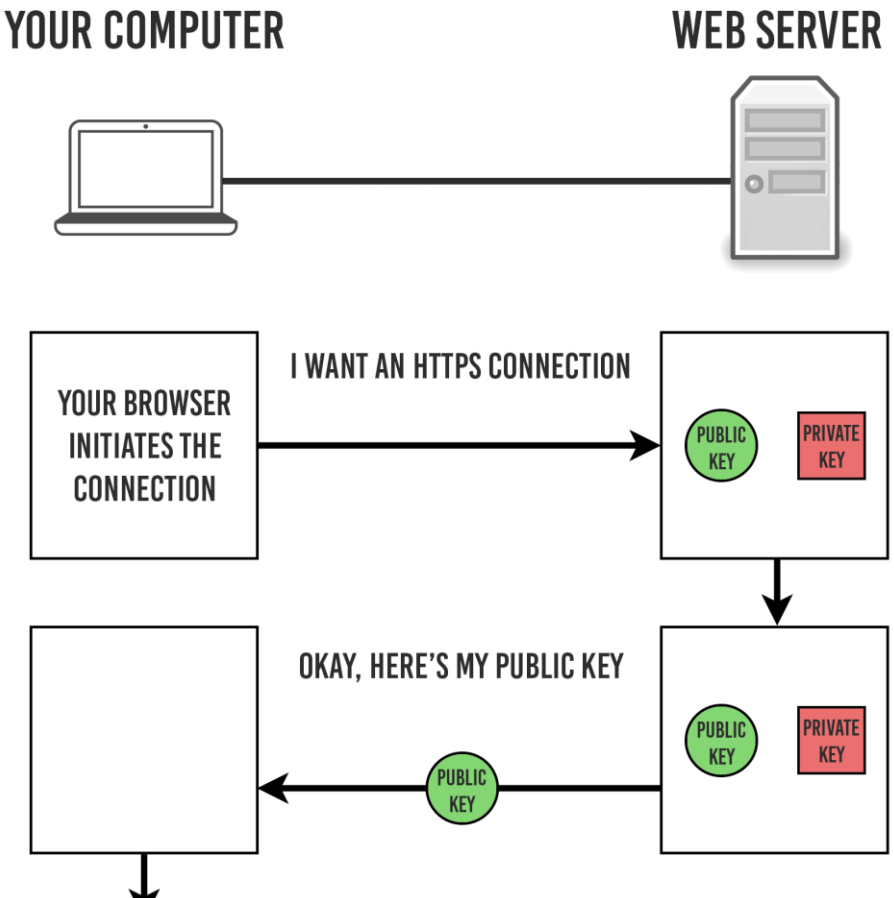
AXXES
IT CONSULTANCY

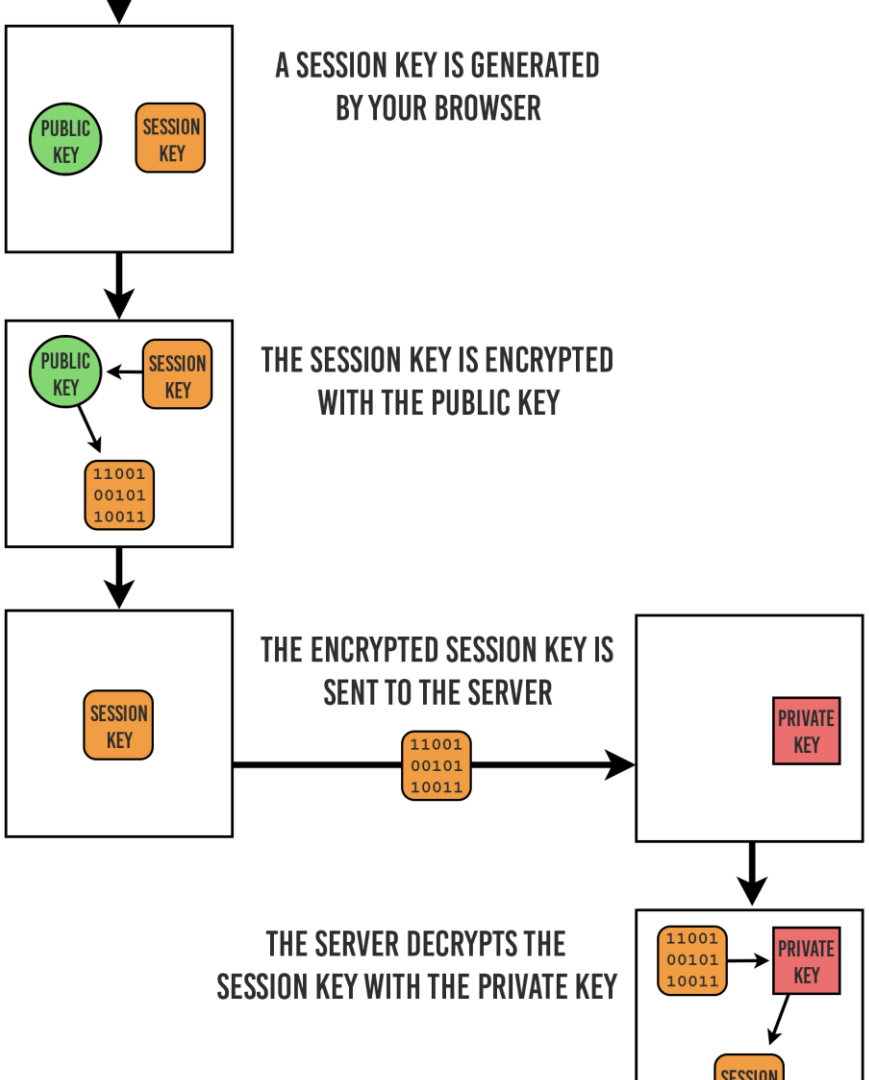
HTTPS made simple

1. **Browser** asks server for certificate
2. **Browser** checks certificate (do I trust the CA?)
3. **Browser** creates new encryption key and sends it to the server (encrypted with public key from certificate)
4. **Server** decrypts encryption key with private key
5. Connection is now secured with encryption key

AXXES
IT CONSULTANCY

HOW HTTPS ENCRYPTION WORKS




PUBLIC KEY
SESSION KEY
A SESSION KEY IS GENERATED BY YOUR BROWSER
PUBLIC KEY
SESSION KEY
11001 00101 10011
THE SESSION KEY IS ENCRYPTED WITH THE PUBLIC KEY
SESSION KEY
THE ENCRYPTED SESSION KEY IS SENT TO THE SERVER
11001 00101 10011
PRIVATE KEY
THE SERVER DECRYPTS THE SESSION KEY WITH THE PRIVATE KEY
11001 00101 10011
PRIVATE KEY

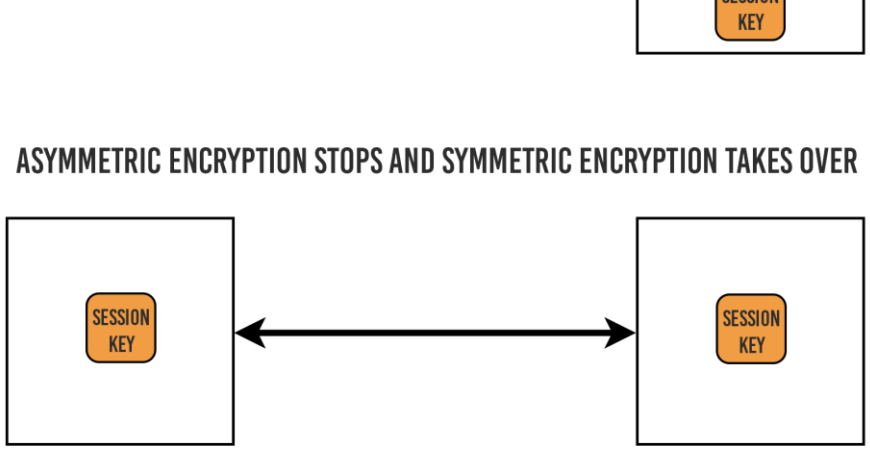


Image source: https://tiptopsecurity.com/how-does-https-work-rsa-encryption-explained/#!prettyPhoto

AXXES
IT CONSULTANCY

HTTPS made simple

Why not use private and public key for security?

AXXES
IT CONSULTANCY

HTTPS explained in detail

The First Few Milliseconds of an HTTPS Connection:

http://www.moserware.com/2009/06/first-few-milliseconds-of-https.html

Security mechanisms

IT IS ABOUT PEOPLE

AXXES
IT CONSULTANCY

Security mechanisms

- Basic authentication
- Digest authentication
- Oauth 1
- Oauth 2

AXXES
IT CONSULTANCY

Security mechanisms

- Basic authentication
- Digest authentication
- Oauth 1
- Oauth 2

AXXES
IT CONSULTANCY

Basic authentication

- Username + password in plain text to the server
- Encoded base64 (not secure)
- Sent with every request
- Unsafe over HTTP, HTTPS somewhat safe

AXXES IT CONSULTANCY

Basic authentication

```
GET /secured HTTP/1.1
Authorization: Basic dXNlcjpzdXBlci1zZWN1cmVk
User-Agent: PostmanRuntime/7.16.3
Accept: */*
Cache-Control: no-cache
Host: localhost:8080
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Base64 decode: dXNlcjpzdXBlci1zZWN1cmVk -> user:super-secured

AXXES IT CONSULTANCY

Small exercise

What is the username and password for this website?

```
GET /secured HTTP/1.1
User-Agent: PostmanRuntime/7.16.3
Accept: */*
Cache-Control: no-cache
Host: localhost:8080
Authorization: Basic dXNlcjpzdXBlci1zZWN1cmVk
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

AXXES
IT CONSULTANCY

Digest Authentication

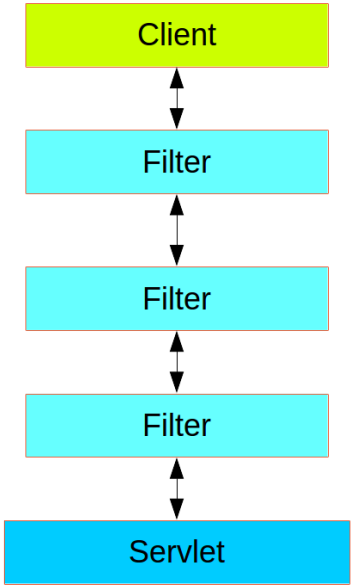
- Username + password encrypted to the server
- Sent with every request
- Use of nonce
- Encrypted with MD5, not considered safe anymore
- Created to use over HTTP for lack of HTTPS

How does Spring do it?

IT IS ABOUT PEOPLE

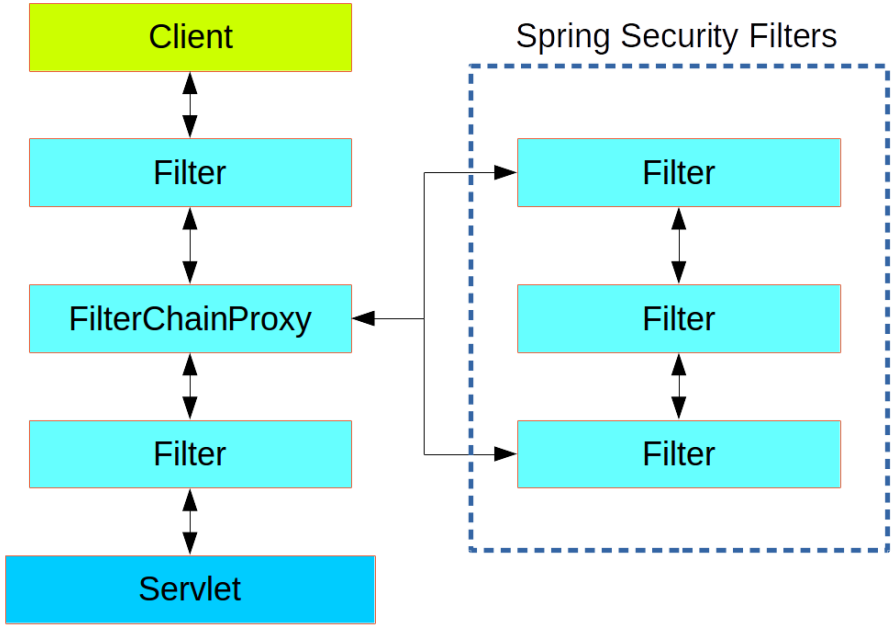
AXXES
IT CONSULTANCY

Spring internals

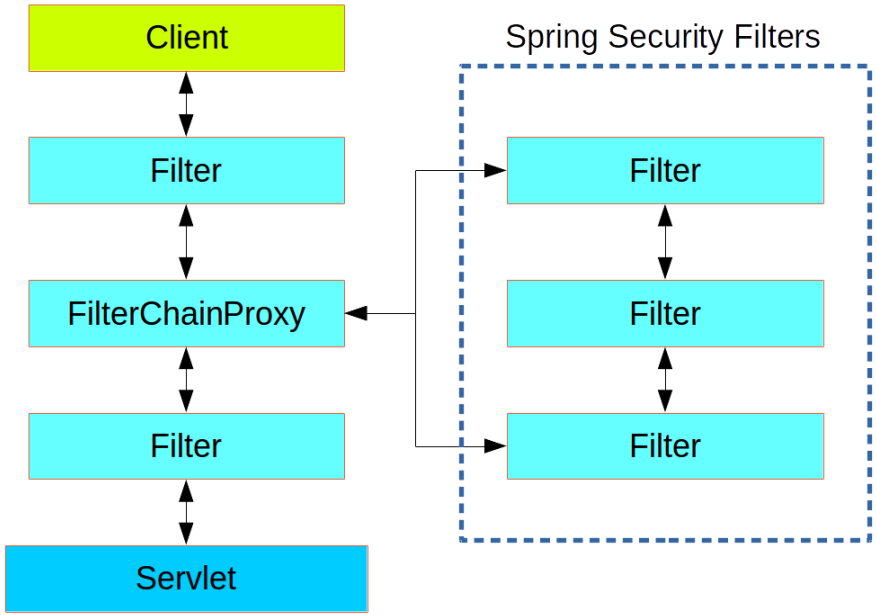


AXXES
IT CONSULTANCY

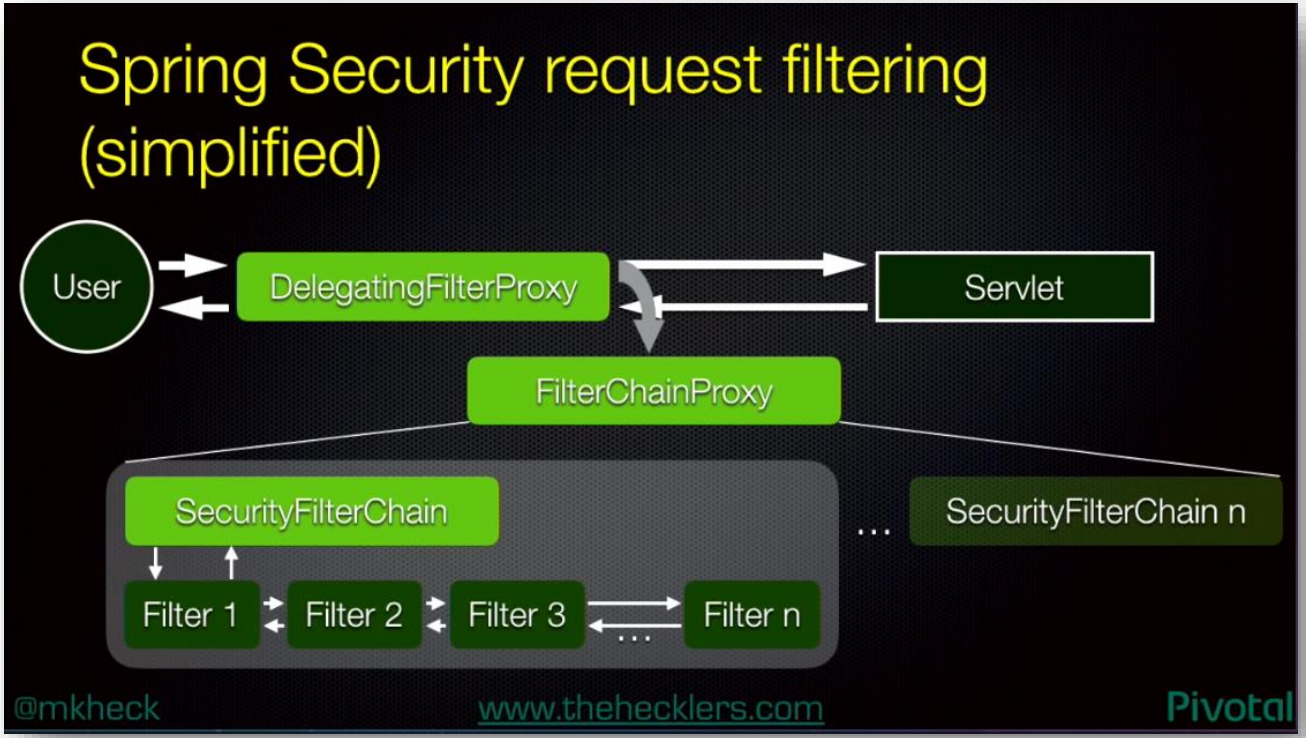
Spring internals



Spring internals

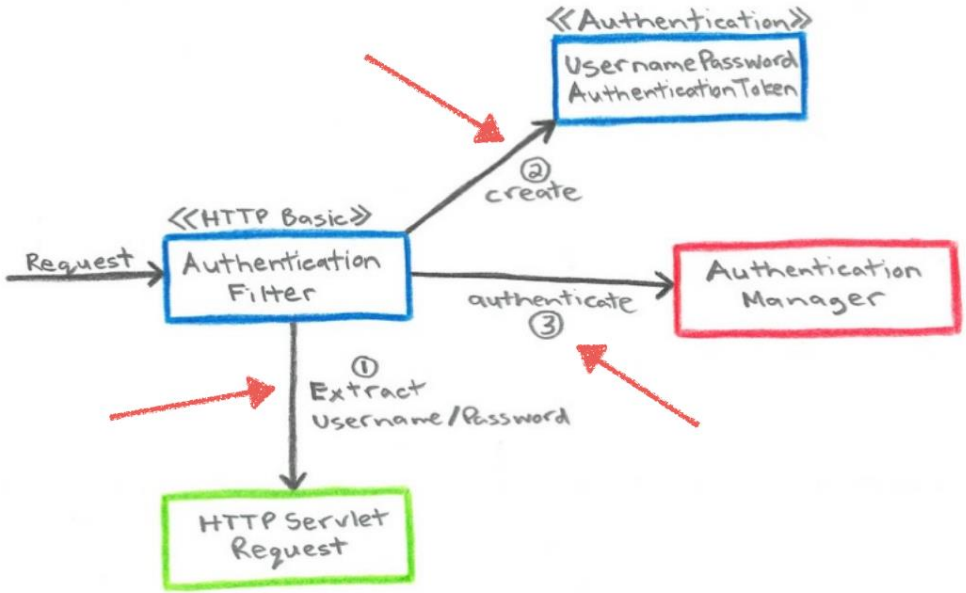


Spring internals



AXXES
IT CONSULTANCY

Spring internals



AXXES IT CONSULTANCY

Spring config

```
@Configuration
@EnableWebSecurity
public class CustomWebSecurityAdapter extends WebSecurityConfigurerAdapter {
```

STEP 1:

Add annotation

STEP 2:

Run

STEP 3:

Spring magic

AXXES IT CONSULTANCY

Spring config

```
@Configuration
@EnableWebSecurity
public class CustomWebSecurityAdapter extends WebSecurityConfigurerAdapter {
```

STEP 1:

Add annotation

STEP 2:

Run

STEP 3:

Spring magic

Creates one security filter chain!

AXXES
IT CONSULTANCY

Spring config

Global configuration

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
            .authorizeRequests()
            .antMatchers("/resources/**", "/signup", "/about").permitAll()
            .antMatchers("/admin/**").hasRole("ADMIN")
            .antMatchers("/db/**").access("hasRole('ADMIN') and hasRole('DBA')")
            .anyRequest().authenticated()
            .and()
            .httpBasic();
```

AXXES
IT CONSULTANCY

Spring config

Global configuration

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
            .antMatcher("/admin/**")
            .authorizeRequests()
            .antMatchers("/resources/**", "/signup", "/about").permitAll()
            .antMatchers("/admin/**").hasRole("ADMIN")
            .antMatchers("/db/**").access("hasRole('ADMIN') and hasRole('DBA')")
            .anyRequest().authenticated()
            .and()
            .httpBasic();
```

AXXES
IT CONSULTANCY

Spring config

Local configuration

```
@Secured("USER")
public String secure() {
    return "Hello Security";
}
```

AXXES IT CONSULTANCY

TODO

- Enable spring security
- Enable basic authentication
- Disable session cookie
  - Configure once with HttpSecurity
  - Configure once with method annotations

Url	Secured	Role
/unsecured	no	/
/secured	yes	Any
/secured/admin	yes	ADMIN

AXXES IT CONSULTANCY

How to access authenticated user?

Autowire in controller

```
@GetMapping("/auto-wire")
public ResponseEntity<SimpleDto> securedUser(final Authentication authentication) {
    service.doSomething(authentication);
    ...
}
```

Not that good.. we have to pass the object to every method call.

Controller -> service -> black hole -> dao

AXXES IT CONSULTANCY

How to access authenticated user?

Static method

```
@GetMapping("/auto-wire")
public ResponseEntity<SimpleDto> securedUser() {
    final Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
}
```

Possible to use static method everywhere. (controller, service, dao)

However static methods are not test friendly...
And not extendable

AXXES IT CONSULTANCY

TODO

- Fetch username from security context
  - Once with autowire
  - Once with static method

URL	Response
/secured/username	{ "value": "username: admin" }

- What is returned when you fetch the name for an unsecured endpoint? (without sending authentication headers)

AXXES
IT CONSULTANCY

How to access authenticated user?

Facade pattern + static method + autowire

1. Declare Facade

```
@Component
public class AuthenticationFacade {

    public Authentication getAuthentication() {
        return SecurityContextHolder.getContext().getAuthentication();
    }
```

2. Autowire and use

```
@GetMapping("/facade")
public ResponseEntity<SimpleDto> securedUser() {
    final Authentication authentication = this.authenticationFacade.getAuthentication();
```

AXXES
IT CONSULTANCY

How to access authenticated user?

Facade pattern + static method + autowire

Possible to autowire anywhere (but don't do that)

Easy to test (mock)

Extentable, add any method to the facade

AXXES IT CONSULTANCY

TODO

Fetch username from security context using facade.

URL	Response
/secured/username	{ "value": "username: admin" }

AXXES IT CONSULTANCY

Another step further...

Using facade in annotations

```
@GetMapping("/{summaryId}/preview")
@PreAuthorize("@authenticationFacade.hasPermission('DOCUMENT', 'READ')")
public ResponseEntity<byte[]> downloadPreview(@PathVariable final Long summaryId) {
```

Uses The Spring Expression Language (SpEL) -> whole new topic

https://www.baeldung.com/spring-expression-language

AXXES IT CONSULTANCY

How to access authenticated user?

Using facade in annotations

```
@Configuration
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class GlobalMethodSecurity extends GlobalMethodSecurityConfiguration {

}
```

To use SpEL you have to add another annotation.

~~ Spring magic

AXXES
IT CONSULTANCY

TODO

Create your own method to check the role

```
@GetMapping("/{summaryId}/preview")
@PreAuthorize("@authenticationFacade.hasRole('USER')")
public ResponseEntity<byte[]> downloadPreview(@PathVariable final Long summaryId) {
```

AXXES
IT CONSULTANCY

Security mechanisms

- Basic authentication
- Digest authentication
- Oauth 1
- Oauth 2

AXXES
IT CONSULTANCY

Idea behind Oauth 1&2

OAuth provides a method for users to grant third-party access to their resources without sharing their passwords. It also provides a way to grant limited access (in scope, duration, etc.).

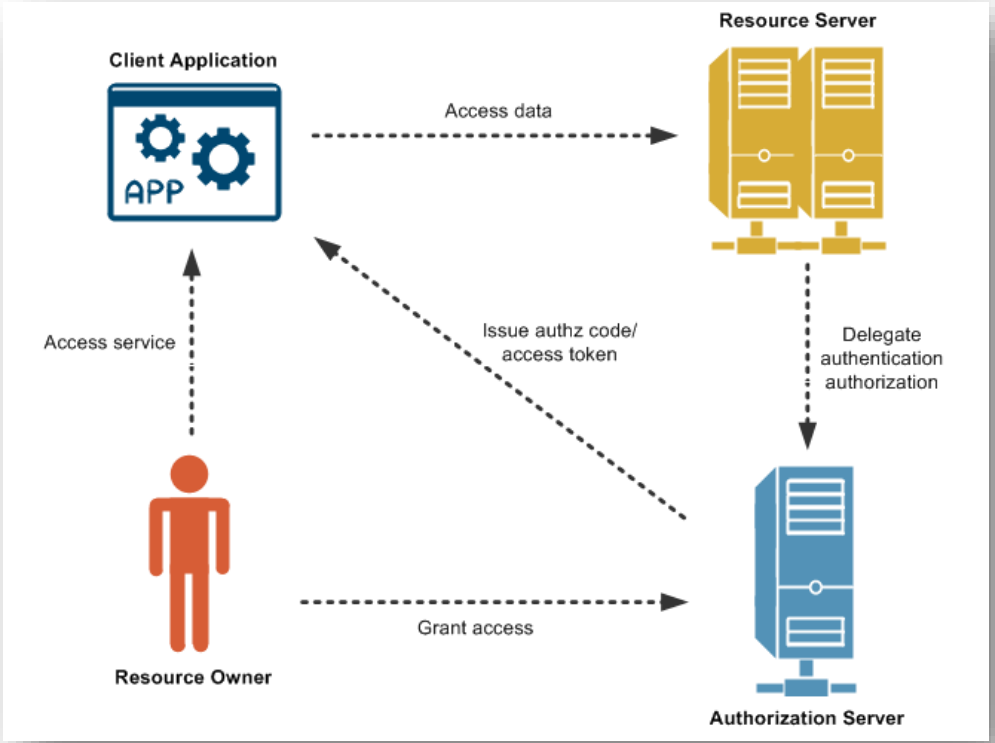
OAuth allows notifying a resource provider (e.g. Facebook) that the resource owner (e.g. you) grants permission to a third-party (e.g. a Facebook Application) access to their information (e.g. the list of your friends).

AXXES
IT CONSULTANCY

Oauth 1 vs Oauth 2

- Oauth 2 is a replacement building on the ideas of Oauth 1
- Better suited for non-browser applications
- Oauth 2 lacks cryptography and trusts on HTTPS
  - Makes it less complicated
- Short lived tokens
  - Is this safe?

Oauth 2 actors



AXXES
IT CONSULTANCY

Oauth 2 tokens

- Gives access to the resources (like session tokens)
- Has to be valid
- Is linked to a user (e.g. in the database)
- Can contain data about the user

AXXES
IT CONSULTANCY

JSON Web Token (JWT)

- Self-contained
- Consists of claims, things that are true
- Published by authorization server (e.g. Google or your own app)
- Signed with PRIVATE key
- Can be verified with PUBLIC key


```
{
  "iss": "Google",
  "exp": 1536779503,
  "jti": "vPrU7-mAEj2Lh9k7ubPpBg",
  "iat": 1536776503,
  "nbf": 1536776383,
  "firstName": "Gert-Jan",
  "lastName": "Heireman",
  "City": "Lille",
  "Role": "Admin"
}
```

AXXES
IT CONSULTANCY

JSON Web Token (JWT)

- 1. Go to www.jwt.io and paste following token:

eyJraWQiOiJUaGlzIGlzIHRoZSBrZXkiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJHb29nbGUiLCJleHAiOjE1MzY3Nzk1MDMsImp0aSI6InZQclU3LW1BRWoyTGg5azd1YlBwQmciLCJpYXQiOjE1MzY3NzY1MDMsIm5iZiI6MTUzNjc3NjM4MywiZmlyc3ROYW1lIjoiR2VydC1KYW4iLCJsYXN0TmFtZSI6IkhlaXJlbWFuIiwiQ2l0eSI6IkxpbGxlIiwiUm9sZSI6IkFkbWluIn0.O_7_R-kpn6wRodXv_ru_9WT8OgJK97B9RpUIahfHMvRbEn0wHp-A4_e6sCgbwhDWqFHz50pB8gxPw_RgA0zX-Ue9tVAO_tJtzcPfn8OymorC81_QR1gtf_y_9gkFGvCcUyZfpPGhOUN0Wvi90iUY8fu0IMoqoT9KfXYF_IR7thKI-t2M0-egwx-hZyf74uq9O0-NdwwSvpJ2GdlZDE8Zntz5J6xDzGwRuRoDmF_PvHIw10zB7VNBJLjnuWy7u4n797F6_QBtonvgQIryprCazUcZMSZ2ZMfmTovNuXdkIt6CqdS-97q1JBEqjfUhFjkTRCAUhu0JHHnyqcl-6NO8GA

- 2. Use PUBLIC key to verify JWT token

-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAnkMJizKC1Io6zAozs3cq4nT3PCxr0cWbVgK+Nb2mIHtCkwjxEYctFBlqesmy96+oDAKayMiDdwdxd9WvtwN6juWOtQuRxN96JM5lN0V2sOUmpUObs7QfDhCJFjaSaxM81zbcXmMKQ4j1NsOFIG6sPgg3lNn3vdcIBLPMZxIFqKoEYqMEIoq+h2kvH5f2y2VvJCNML8GMNBvcU6v8u1qCPcenVhJesliGrGE2mcQ15/r3rq9W/EmeKOhbatI8plsdWd+H+nj5NsdP/kEWaC5Vkp2yXCdCU6AmzjjnU1PP+z5S5+hX887/I2sqLh+KzxIIOPvE7DK7EjinUGZMa3DcewIDAQAB-----END PUBLIC KEY-----

IT IS ABOUT PEOPLE

The End